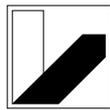


Structuring Descriptive Data of Organisms — Requirement Analysis and Information Models

**(Strukturierung organismischer Beschreibungsdaten
– Anforderungsanalyse und Informationsmodelle)**



**UNIVERSITÄT
BAYREUTH**

Dissertation zur Erlangung des Doktorgrades
der Fakultät Biologie, Chemie und Geowissenschaften
der Universität Bayreuth

vorgelegt von

Gregor Hagedorn

Institute for Plant Virology, Microbiology and Biosafety,
Federal Biological Research Center for Agriculture and Forestry,
Königin-Luise Str. 19, 14195 Berlin, Germany

Bayreuth, Juni 2007

Vollständiger Abdruck der von der Fakultät für Biologie, Chemie und Geowissenschaften der Universität Bayreuth genehmigten Dissertation zur Erlangung des akademischen Grades eines Doktors der Naturwissenschaften (Dr. rer. nat.).

Schlagwörter: Bioinformatik; Biodiversität; Identifikation; Taxonomie;
SDD; TDWG; DELTA; DeltaAccess; DiversityDescriptions.
Keywords: bioinformatics; biodiversity; identification; taxonomy;
SDD; TDWG; DELTA; DeltaAccess; DiversityDescriptions.
ACM Computing Classification System:
J.3 LIFE AND MEDICAL SCIENCES

Die vorliegende Arbeit wurde unter der Leitung von Herrn Prof. Dr. G. Rambold (Lehrstuhl Pflanzensystematik, Abteilung Mykologie) angefertigt.

Einreichung der Dissertation (*Date of submission*): 11. Juni 2007

Tag des wissenschaftlichen Kolloquiums (*Date of examination*): 28. November 2007

Prüfungsausschuss (*Thesis Committee*):

Prof. Dr. Gerhard Rambold (Erstgutachter, *advisor and first promotor*)

Prof. Dr. Prof. Stefan Jablonski (Zweitgutachter, *second promotor*)

Prof. Dr. Ingolf Steffan-Dewenter (Vorsitzender, *chairperson*)

Prof. Dr. Sigrid Liede-Schumann

Prof. Dr. John Tenhunen

This is an Open Access publication distributed under the terms of the Creative Commons Attribution-Noncommercial-Share Alike 3.0 License (<http://creativecommons.org/licenses/by-nc-sa/3.0/>).

Summary

Data that describe organisms in a structured form are indispensable not only for taxonomic and identification purposes, but also many phylogenetic, genetic, or ecological analyses. By analyzing existing information models and performing selected fundamental requirement analyses, the present work contributes to a broadening of the understanding of these forms of data. It falls into an interdisciplinary area between biology and information science.

The term “descriptive data” is understood here in a broad sense: As descriptions of individuals, populations, or taxa, intended for various purposes (e. g., genetic, phylogenetic, diagnostic, taxonomic, or ecological), and covering a wide array of observation methods and data types (e. g., morphological, anatomical, genetic, physiological, molecular, or behavioral data). The position of descriptive data in the context of biodiversity framework concepts (covering, e. g., nomenclatural data, specimen collection data, or resource management) is discussed.

A number of fundamental problems arise when modeling biological descriptive data. The ways in which existing data exchange formats, information models, and software applications address them are studied and future possible solutions are outlined.

One such solution, the information model for the software “DiversityDescriptions (Delta-Access)” is one of the results of this thesis and fully documented (Ch. 7). This entity relationship model fully supports the concepts of the traditional DELTA data exchange format (Description Language for Taxonomy; TDWG standard since 1986). It further improves on DELTA by introducing “modifiers” as a new terminology class, by introducing a more flexible system of handling statistical measures, by improving the handling of multilingual data sets, by supporting subset and filter features for concurrent collaborative editing (instead of supporting these for report-generation purposes alone), by supporting improved character attributes to create natural language descriptions from structured descriptions, and by adding metadata for a data set to improve the ability of data exchange without external documentation.

In preparation of a future improved information model for descriptive data, the results of three requirement analyses are presented: a data-centric analysis of general concepts, a process-centric analysis of identification tools, and a high-level use case analysis.

The first analysis (Ch. 4) is a structured inventory of fundamental approaches and problems involved in collecting and summarizing scientific descriptions of organisms. It is informed in part by current practices in information science, comparative data analysis, statistical, descriptive or phylogenetic software applications, and data exchange formats in biodiversity informatics. At the end three topics are discussed in particular detail (“Federation and modularization of terminology”, “Modifiers”, and “Secondary classification resulting in description scopes”).

Except for phylogenetic analyses, identification is the most common usage of descriptive data. The second analysis (Ch. 5) therefore studies the processes, data structures, presentational and user interface requirements for printable and computer-aided identification tools (“keys”).

Finally, a general use case analysis is performed with the goal of creating a framework of high-level use cases into which present as well as future requirements may be integrated (Ch. 6).

All three requirement analyses are explorative and do not fulfill formal criteria of software engineering. They identify many requirements not addressed by the relational DiversityDescriptions model. Some of these could only be explored and await future solutions. For others solutions are proposed (some of which could already be incorporated into the design of SDD, an xml-based TDWG standard since 2005): The traditional data types are changed into an extensible character type model. The importance of data aggregation concepts was recognized to be fundamental. Complementary to data aggregation, the present and potentially future use of data inheritance along the lines of the taxonomic hierarchy is briefly studied. The concept of calculated characters could be addressed only insofar as the mapping between values can potentially be generalized. Character decomposition models are studied, but ultimately the traditional character concept, supplemented with a forest of ontologies for compositional and generalization concept hierar-

chies, is preferred as a more general concept. Both the traditional character subset and character applicability models can be integrated into concept hierarchies.

Zusammenfassung (German summary)

Strukturierte Beschreibungsdaten von Organismen sind nicht nur für Taxonomie und Bestimmung, sondern auch viele phylogenetische, genetische oder ökologische Analysen unentbehrlich. Durch Analysen existierender Informationsmodelle und durch fundamentale Anforderungsanalysen leistet die vorliegende Arbeit einen Beitrag zum Verständnis dieser Daten. Sie ist interdisziplinär zwischen Biologie und Informatik angelegt.

Der Begriff „Beschreibungsdaten“ wird in einem weiten Sinn definiert, nämlich als Beschreibungen von Individuen, Populationen oder Taxa, gesammelt z. B. für genetische, phylogenetische, diagnostische, taxonomische oder ökologische Zwecke, und unter Einschluss diverser Datentypen (z. B., morphologische, anatomische, genetische, physiologische, molekulare oder Verhaltensdaten). Die Abgrenzung von Beschreibungsdaten zu anderen Biodiversitätsdaten (z. B. Nomenklatur, Sammlungsdaten, oder Medien- und Literaturressourcen), und das Konzept übergreifender Rahmenkonzepte für Biodiversitätsdaten wird erläutert.

In der Arbeit werden grundlegende bei der Modellierung von Beschreibungsdaten auftretende Probleme besprochen, vorhandene Lösungsansätze in Datenaustauschformaten, Modellen und Programmen untersucht, und zukünftige Lösungen aufgezeigt.

Eine solche Lösung, das relationale Informationsmodell für die Software „DiversityDescriptions (DeltaAccess)“, ist ein Ergebnis dieser Arbeit und wird im Detail dokumentiert (Kap. 7). Dieses Modell deckt die Konzepte des traditionellen DELTA-Datenaustauschformats (Description Language for Taxonomy; TDWG Standard seit 1986) vollständig ab. Darüber hinaus erweitert es DELTA erheblich. Es führt eine neue Form von Beschreibungsvokabular („Modifizierer“), ein flexibleres System für statistische Maße und erweiterte Merkmalsattribute zur Erzeugung natürlichsprachlicher Beschreibungen aus strukturierten Daten ein. Weiterhin verbessert es die Behandlung mehrsprachiger Datensammlungen, nutzt Filter auch für gemeinschaftliches Redigieren (anstatt diese nur zur Berichterzeugung zu nutzen), und unterstützt Metadaten für Projekte.

Ein weiteres wesentliches Ergebnis dieser Arbeit sind die Resultate von drei Anforderungsstudien, die eine solide Basis für künftige Weiterentwicklungen darstellen: Eine datenorientierte Studie allgemeiner Konzepte, eine prozessorientierte Analyse von Bestimmungsmethoden, sowie eine allgemeine „Use-Case“-Analyse.

Die erste Studie (Kap. 4) ist eine strukturierte Aufzählung grundlegender Probleme, welche bei der Beschreibung und Charakterisierung von Organismen auftreten. Die Informationen dazu basieren auf Datenverwaltungs- und statistischen Analysemethoden, wie sie in allgemein-statistischer, phylogenetischer und taxonomischer Software (bzw. Datenaustauschformaten) vorkommen. Der allgemeine Teil wird ergänzt durch drei ausgewählte vertiefende Analysen: „Verteilte und modularisierte Terminologie“, „Modifizierer“ und „Sekundäre Klassifikationen in Beschreibungen“.

Die zweite Analyse (Kap. 5) untersucht Bestimmungsmethoden, welche die – neben phylogenetischen Analysen – wohl wichtigste Anwendung von Beschreibungsdaten sind. Die Prozesse, Daten, Darstellungsformen und Benutzeroberflächen von gedruckten oder Computer-gestützten Bestimmungshilfsmitteln werden detailliert in Hinsicht auf Anforderungen an das Informationsmodell untersucht.

Schließlich wird in der „Use-Case“-Analyse (Kap. 6) der allgemeine Gebrauch von Beschreibungsdaten untersucht. Dabei wird eine Gliederung erstellt in welche gegenwärtige und künftige Anforderungen integriert werden können.

Alle drei Anforderungsanalysen sind explorativ und erfüllen keine formalen Kriterien der Softwareentwicklung. In ihnen werden viele Punkte erfasst die nicht durch DiversityDescriptions

abgedeckt werden. Etliche Anforderungen und Probleme können nur herausgearbeitet werden und müssen auf zukünftige Lösungen warten. Zum Teil können aber bereits mögliche Lösungen präsentiert oder skizziert werden. Einige sind bereits in das Design von SDD, dem neuen xml-basierten TDWG Standard für Beschreibungsdaten seit 2005, eingeflossen: Die traditionellen Datentypen werden als erweiterbares Typsystem neu konzipiert; die Bedeutung von Datensummierung und Synthese wird neu bewertet; die umgekehrte Bedeutung von Datenvererbung entlang der taxonomischen Hierarchie wird kurz studiert. Berechnete Merkmale werden insoweit abgedeckt, als sie eine einfache Abbildung zwischen zwei Merkmalen sind („mapping“). Merkmals-Dekompositionsmodelle werden untersucht, das traditionelle Merkmalskonzept jedoch als das allgemeinere Konzept bevorzugt. Dieses wird durch mehrfache strukturelle und generalisierende Ontologien (Konzepthierarchien) ergänzt. Sowohl traditionelle Untermengen („Subsets“) als auch Merkmalsabhängigkeiten können hier integriert werden.

Acknowledgements

Firstly, I am very grateful to my advisor Prof. Dr. G. Rambold (Bayreuth) for his interest in the topic, discussions and advice, as well as great encouragement, support, and patience during the long period of working on this thesis. He as well as others, especially Prof. R. Morris (Boston, USA), but also Prof. Dr. G. Deml, Dr. D. Triebel, Prof. Dr. W. Gams, and Dr. O. Hering never failed to encourage me to continue and finish the work.

I am very grateful to Prof. Dr. G. Rambold, Prof. R. Morris (Boston, USA), Prof. Dr. W. Berendsohn (Berlin), Dr. M. Dallwitz (Giralang, Australia), Prof. Dr. W. Gams (Baarn, NL), Dr. F. Bungartz (Galapagos Islands, Ecuador) and Dr. R. Pankhurst (Edinburgh, UK) for many enlightening discussions and constructive criticism of the entirety or parts of this work. I am further deeply indebted to the many other colleagues and friends who also have discussed problems, read parts of the thesis, or answered questions. I would especially like to thank: J. Asiedu (Boston, USA), Dr. N. Bailly (Paris, France), D. Barnier (Queensland, Australia), M. Choo (Perth/Kensington, Australia), N. Cross (†, USA), A. Ekrut (Berlin), C. Gallut (France), Dr. C. Germeier (Quedlinburg), Dr. E. Gibaja Galindo, Prof. P. B. Heidorn (Urbana-Champaign, USA), D. Hobern (Copenhagen, Denmark), J. Ingenhaag (München), Prof. J. Kennedy (Edinburgh, UK), E. Kolster (New Zealand), Prof. D. Maltais (Québec, Canada), D. Neubacher (München), Dr. T. Paterson (Edinburgh, UK), Dr. G. Rousse (France), Dr. A. Rubner (Karlsruhe), Dr. M. Scholler (Karlsruhe), Dr. S. Shattuck (Canberra, Australia), Dr. K. Thiele (Perth/Kensington, Australia), J.-M. Vanel (France), R. Vignes Lebbe (Paris, France), G. Whitbread (Canberra, Australia), and Zhimin Wang (Boston, USA).

While this thesis was in preparation, an international working group created an XML-based data exchange standard for descriptive data (TDWG SDD, compare p. 20). The discussions in this group strongly influenced the ideas in this thesis and the author is indebted to everybody contributing to this working group during personal meetings or online discussions. Over 60 people contributed to the SDD discussions, both by participating and organizing them. It is impossible to list them all, but I want to thank them all. Much of the travel required for the international discussions was supported through the following project grants (in chronological order): BioCase (Biological Collection Access Service for Europe, EU funding), BIOLOG-GLOPP and GBIF-D-Myk (both BMBF funding), GBIF, and the TDWG infrastructure project (funded by The Gordon and Betty Moore Foundation). The help is greatly appreciated.

Similarly, I thank all my work colleagues at the BBA and in projects for their support that enabled me to undertake this work, particularly A. Hansen, Prof. Dr. G. Deml, Prof. Dr. C. Reichmuth, V. Ristau, Dr. O. Hering, Dr. H. Nirenberg, C. Hild, Dr. D. Triebel, Dr. M. Weiss, A. Kohlbecker, J. Ingenhaag, and Prof. Dr. M. Piepenbring.

Finally, I would like to thank my wife Almut, my son Jakob, my mother and stepmother and all other members of my family for the love and support they gave me throughout this work.

Table of contents

1. Introduction	12
1.1. Biodiversity informatics	12
1.2. Descriptive data for identification and phylogeny	12
1.3. Other uses of descriptive data	15
1.4. Scope, motivation, and constraints of the current work	15
2. Methods	18
2.1. Explorative requirement analyses	18
2.2. Survey of information models and software	18
2.3. UML use cases	23
2.4. UML static class diagrams and ER models	24
2.5. Abbreviations	26
3. Selected definitions	27
3.1. Descriptive data in the context of biodiversity data	27
– Definition of ‘descriptive data’	27
– Biodiversity ‘framework concepts’	28
– Ambiguous or border-line cases of ‘descriptive data’	30
3.2. The term ‘character’	31
3.3. Terms for ‘object parts’	33
3.4. Comparison of current usage of terms	34
4. Fundamental aspects of description models	36
4.1. Introduction	36
4.2. Context, recognition, and language	36
4.3. Natural language descriptions	39
4.4. Structured descriptions and the concept of terminology	42
– Level of abstraction of descriptive information models	42
– Generalization and terminology	44
– Static versus dynamic terminology models	45
– Reaching terminological stability	47
– Relation between terminology and software implementations	48
4.5. Data types	49
– Measurement scales	49
– Continuous versus discrete variables	51
– Categorical versus quantitative (measurement) data	52
– Singularity, extension and connectedness of categories	53
– Data types in computer programming	55
– Unconstrained text	56
– Molecular sequence data	57
– Complex quantitative data types	59
– Media data	60
– Implemented data type systems	61
– Basic property types	62
4.6. Mapping between data types	66
– Mapping univariate continuous measurements to categories	66
– Mappings within categorical data	68
– Mapping complex quantitative data to categorical data	69
– Mappings and definition of categories	70

– Mapping unconstrained text to structured data.....	71
– Mappings involving more than two characters.....	71
– Calculated characters.....	72
4.7. Coding status.....	74
4.8. Character dependency.....	76
– Character dependency in general.....	76
– Character applicability rules.....	76
– Convertibility of applicability rules.....	79
– Coexistence of character applicability rules.....	81
– Cascading character applicability rules.....	82
– Current support in some applications and data standards.....	82
4.9. Raw data and data aggregation.....	83
– Introduction.....	83
– Standard aggregation methods.....	85
– Inappropriate aggregation results.....	87
– Aggregating aggregated data.....	88
– Data recording levels (sample data).....	89
– Linked observations.....	90
– Special aggregation cases.....	92
– Aggregation within individuals.....	93
– Boolean operators between states of categorical characters.....	95
– Boolean operators between characters.....	98
4.10. Inheriting data.....	99
– Data compilation versus data inheritance.....	99
– Inductive inheritance (upwards).....	99
– Deductive inheritance (downwards).....	100
– Current models.....	101
– Implicit data.....	102
– Compatibility testing as a quality control measure.....	103
4.11. Description storage models.....	104
– Introduction.....	104
– Categorical data: Character matrix vs. character state matrix.....	104
– Quantitative data and statistical measures.....	110
– Value order in character data.....	113
– Character decomposition models.....	116
– Concept hierarchies.....	125
4.12. Descriptive ontologies.....	131
– Object composition.....	131
– Multiplicity of objects in compositions.....	141
– Spatial arrangement of objects in compositions.....	147
– Generalization of object parts (compositional concepts).....	153
– Change of object concepts through temporal development.....	162
– Properties.....	164
– Methods.....	169
– Relations between properties and methods.....	176
4.13. Federation and modularization of terminology.....	180
– Introduction.....	180
– Managed federations.....	180

– Terminology modules.....	181
– Extending shared terminology definitions.....	182
– Terminology modules and class hierarchy.....	183
– Models to support multiple distributed terminologies.....	185
– Conclusions.....	188
4.14. Modifiers.....	189
– Introduction.....	189
– Definition.....	191
– Current usage of modifier-related concepts.....	192
– Modifier sets and sequences.....	199
– Modifier combinations.....	199
– Modifiers as an alternative to character proliferation.....	201
– Modifier classes.....	203
– Character- versus value-modifiers.....	214
4.15. Secondary classification resulting in description scopes.....	215
– Introduction.....	215
– Mating type and sex.....	217
– Generations, life cycle, and developmental stages.....	218
– Other classifier concepts.....	219
– Generalized term for sex, generation, life cycle stages, etc.....	220
– Context of secondary classifier data.....	221
– Classifier-related characters.....	221
– Existing models of handling secondary classifiers.....	222
– Summary and conclusions.....	227
5. Identification methods.....	229
5.1. Introduction.....	229
5.2. Classification of identification methods.....	230
– Kind of data used for identification.....	230
– Levels of interaction.....	230
– Phases of interactive identification.....	231
– Structural classification of identification keys.....	233
– Propositional versus object matching metaphors.....	238
– “Promorph” and “looks like” metaphors.....	238
– Radford's classification.....	240
– Other classification criteria for identification keys.....	240
5.3. Presentation styles of identification keys.....	242
– Printable branching keys.....	242
– Computer-aided branching keys.....	247
– Printable multi-access keys.....	249
– Computer-aided multi-access keys.....	251
– Tabular keys.....	256
5.4. Requirement summary.....	257
5.5. Linking multiple keys.....	259
– Transferring progress information between multi-access keys.....	260
– Transferring progress information between branching and multi-access keys.....	262
5.6. Equality criteria and error tolerance.....	264
5.7. Character ranking and guidance.....	267
– Authored character guidance.....	267

–	Algorithmic character guidance.....	270
–	Combining algorithmic with authored character guidance.....	276
–	Alternative algorithms.....	276
–	Presentation of character guidance in multi-access keys.....	277
6.	Use case analysis.....	277
6.1.	Introduction.....	277
6.2.	Roles and agents (use case actors).....	278
6.3.	Information acquisition.....	280
–	Project management.....	280
–	Definition of terminology.....	281
–	Descriptions.....	288
6.4.	Information retrieval.....	297
–	Selection of language and audience representations.....	298
–	Selection of branching keys.....	298
–	Querying container level metadata.....	300
–	Querying natural language description data.....	300
–	Querying coded description data.....	300
6.5.	Information review and interpretation.....	301
–	Analysis of data quality and completeness.....	301
–	Analysis of character correlation.....	303
–	Analysis of character applicability.....	304
–	Aggregating descriptions.....	305
–	Creation of class hierarchies.....	306
–	Analysis of character evolution.....	306
–	Creation of diagnostic subsets.....	307
6.6.	Identification.....	308
–	Identification keys.....	308
–	Switching between branching and multi-access keys.....	309
–	Confirmation of identification.....	309
–	Failure of identification.....	310
–	Identification of potential taxon concepts.....	311
–	Creation of branching keys.....	312
–	Dynamic character recommendations for identification purposes.....	313
–	Character recommendations for identification purposes based on the phylogeny.....	314
6.7.	Information application.....	315
–	Report generation.....	315
–	Taxon pages.....	319
–	Data exchange and archival exports.....	320
6.8.	Open aspects.....	322
7.	Information model for DiversityDescriptions 1.9.....	322
7.1.	Introduction.....	322
7.2.	Logical model for DiversityDescriptions 1.9.....	324
–	Packages and subsystems.....	324
–	Package: Terminology.....	325
–	Package: Descriptions.....	329
–	Package: Resources.....	331
7.3.	Physical model for DiversityDescriptions 1.9.....	332

– Implementation constraints.....	333
– Entity relationship diagrams	335
– Data dictionary.....	339
– Project Properties.....	352
– Statistical measures in DiversityDescriptions.....	356
8. Final Discussions.....	356
8.1. A progression of information models.....	357
– Background of DELTA and NEXUS	357
– DiversityDescriptions	358
– SDD	360
8.2. Results of requirement analyses	361
8.3. Description logic and unified systems.....	367
8.4. Future relevance: A proposal to record identification data	369
9. References	371
10. Appendix	388
10.1. Brief history of SDD	388
10.2. Code fragments for evaluating character applicability rules	388
– Procedural pseudo-code for character applicability.....	389
– SQL code for character applicability in relational databases.....	390
10.3. Preferred, alternative, and rejected terms for identification keys.....	394
– “Interactive identification”	394
– Branching keys	396
– Couplet.....	397
– Multi-access keys.....	397
– “Synoptic key”, a confused term	398
10.4. SQL code for DiversityDescriptions 1.9	400
10.5. Index of figures	404
10.6. Index of tables	408
10.7. Overview of collected requirements.....	409

1. Introduction

1.1. Biodiversity informatics

In parallel with the development of computer and information science, the application of these disciplines in biology has increased. Unfortunately, the general term *bioinformatics* has in practice become a synonym for the application of informatics in molecular biology – being the largest application of computer science in biology. Although good arguments exist to reclaim the general term and develop a comprehensive discipline of *Biological Informatics* (Heidorn & al. 2007), in recent years the perception of bioinformatics as a branch of molecular biology has led to problems like inappropriate reviews and representation. This forced the introduction of new names for other specialized subdisciplines: *neuroinformatics* (applications in neurobiology), *phyloinformatics* (use of informatics to infer phylogenetic relationships, e. g., Cracraft 2002; Page 2004), *ecoinformatics* (applications in ecology; e. g., www.ecoinformatics.org), and *biodiversity informatics* (applications in biodiversity studies; e. g., Berendsohn 2001a, Berendsohn 2001b, and <http://jbi.nhm.ku.edu/index.php/jbi> – a recently established eponymous online journal).

All disciplines of biological informatics may be viewed either as biology or as applied information science. A classification as information science is clearly appropriate, where the biological subject catalyzes the development of new techniques. To a large extent, however, the disciplines apply existing information and computer science techniques to biological subjects. They can therefore aptly be considered biological research – just like the application of microscopes in biology is considered biology rather than optics.

The scientific and computing infrastructure of molecular bioinformatics is well established (e. g., through EMBL, NCBI, etc.), but the other branches of bioinformatics have also improved their infrastructure in recent years (for phyloinformatics see, e. g., the “Cyberinfrastructure for Phylogenetic Research, CIPRES” at <http://www.phylo.org>). The most important recent development in biodiversity informatics was probably the establishment of the Global Biodiversity Information Facility (GBIF, Lane & Edwards 2007) established 2002 in Copenhagen on the basis of an international “memorandum of understanding”. GBIF's mission is to facilitate digitization and global dissemination of biodiversity data, so that people from all countries can benefit from the use of this information. Importantly, GBIF, through its nationally funded nodes, has also spurred many national research activities.

Recently, the implementation of a new “Encyclopedia of Life” (EOL) has been announced (EOL.org 2007). This project, based on plans by Wilson (2003), will be a new infrastructure project aiming specifically at species descriptions. The relations between GBIF, the US-funded EOL, and the – phylogenetically oriented – Tree of Life web project (2007) will have to be worked out in the coming months.

The current work falls into the area of biodiversity informatics. It is an interdisciplinary approach, describing current solutions, requirements, and problems encountered from the perspective of a biologist. The subject is approached from different angles: a fundamental analysis of aspects of descriptive data, an inventory and generalization of the processes in which they are used in biology (use case analysis), and a detailed documentation of a tested information model (DiversityDescriptions). The latter should be considered as a basis for future development rather than as a final product. To aid in the communication across barriers of language and discipline, the language of information science is used as far as possible, together with examples and illustrations.

1.2. Descriptive data for identification and phylogeny

“*Descriptive data*” is understood here in a wide sense as data on intrinsic properties of organisms, i. e., data by which individuals, populations, or taxa may potentially be recognized regard-

less of context. This wide definition shall neither be limited to certain purposes (e. g., genetic, phylogenetic, diagnostic, or taxonomic) nor to specific observation methods and data types (e. g., morphological, anatomical, genetic, physiological, molecular, or behavioral). Not included are descriptions of *ownership* and *events* (e. g., collection, identification, conservation, or nomenclatural events). The present discussion is often also applicable to descriptions of other object classes that biologists work with (places, ecosystems, rocks, soils, climate), but these classes are not the primary focus of this work. A more detailed discussion of this will follow (p. 27).

The driving force behind much of the interest in descriptive data is the identification of organisms. A service offering identification based on descriptions is a primary information portal to biodiversity knowledge whenever the name is not yet known. Identification tools are the “query mechanism” for names and descriptive data of objects.

The name obtained in the identification can then be used to obtain further information. The majority of biodiversity information is indexed by taxonomic names. Names are the key to biodiversity (Thompson 1996), the call numbers to the books in the library of life (Janzen 1991). Supported by name synonymies, they summarize our knowledge on identity, similarity, and relatedness of organisms, simplify communication, allow the association of independently gained knowledge fragments, and enable management and sustainable use (Janzen 1991). However, with a newly found specimen in hand, keys to the keys and indexes to the call numbers are needed, i. e., identification tools. And one needs to read in these books: how they look like, how they behave, how they interact with other organisms. These are descriptive data.

Identification and descriptive data are not only relevant for an initial identification, but are required to interpret the “concept” of a taxon name associated with a knowledge fragment. However, name identity in the current taxonomic name system is usually determined in a “fuzzy” mode. Two organisms with the same name are assumed to share all taxon-specific properties, unless it is specifically known that the name is misapplied or variably applied (historically, in different geographic regions, taxonomic schools, etc.). The development of taxon concepts or “potential taxa” (see, e. g., Berendsohn 1995, Zhong & al. 1996, Berendsohn 1997, Pullan & al. 2000, Ytow & al. 2001, Berendsohn & al. 2003, Geoffroy 2003, Geoffroy & Berendsohn 2003, Berendsohn & Geoffroy 2007) tries to operationalize the knowledge that multiple taxon concepts may have the same name by adding usage and publication references to names.

Ultimately, however, ecological, descriptive, or socio-cultural information is not defined by a taxon name plus literature reference, but by the identification process in which an individual organism is associated with the name. The only exception to this rule is the situation where taxonomists are studying type material. However, only selected descriptive characteristics will be studied on the type material itself. Together with additional information defining the variability (width) of the taxon concept, these characteristics are then used by everybody else as the basis of identification.

Misinterpretations that have led to separate potential taxon concepts can be traced if the correlation between descriptions, specifically between characters leading to the identification, is sufficiently well understood. In the current thesis a system of “identification accessions” is proposed to create an operational system for storing and managing the descriptive information collected in the identification process (see pp. 295 and 369).

Identification is not limited to the species level. The concept of a species is indeed special in biology, at least in taxonomic groups where the biological species concept is applicable. No similarly objective concepts exist for infraspecific (e. g., variety, or subspecies) and supraspecific taxa (e. g., genus, family, or order). The hierarchical arrangement may be an artificial, operational classification or a phylogenetic classification (i. e., a classification mirroring an inferred evolutionary history). In the latter case the topology of the phylogenetic tree can be inferred by scientific methods, but a substantial amount of arbitrariness remains in choosing between alternative trees and in selecting *which* nodes are named and at which rank.

A phylogenetic classification is generally preferred. It maximizes the average *predictive* value of the classification; i. e., studying the similarity of some characters (e. g., molecular) enables pre-

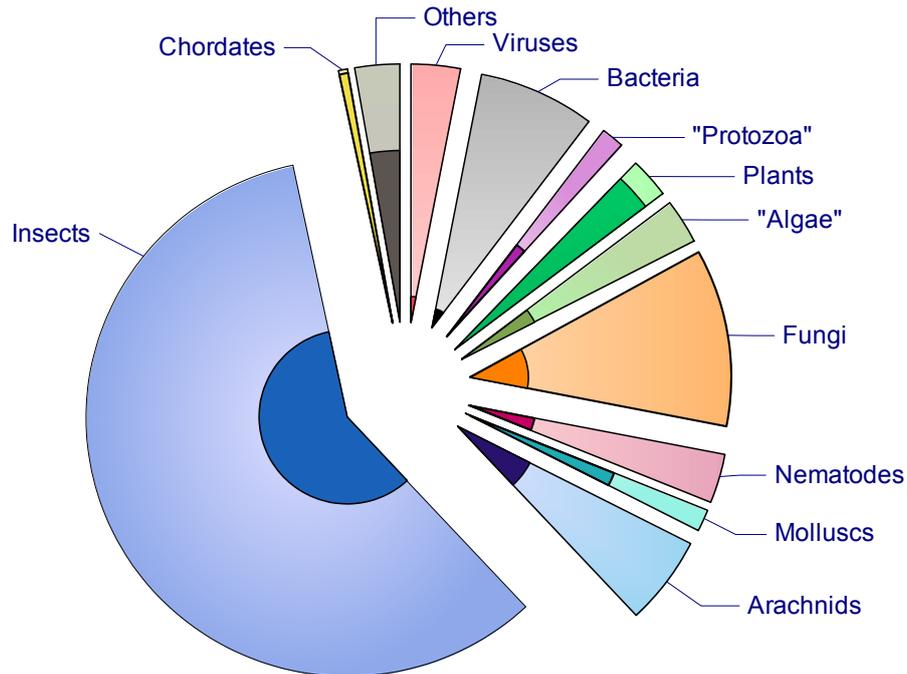


Figure 1. Estimated species diversity of major groups and proportion of known taxa. The total area of the segments indicates the estimated total number of species living on earth. The area of the darker colored inner segments indicates the number of species for which names and descriptions have been published. For example, only ca. 5% of fungi, 10% of algae, 11.5% of insects are already known, but ca. 96% of chordates and 84% of (higher) plants (after Purvis & Hector 2000 and Hawksworth & Kalin-Arroyo 1995).

dictions about other characters. Popular algorithms for producing a phylogenetic tree maximize the likelihood that organisms classified together have common character states. In contrast, artificial classification systems are most useful if a single characteristic is of central importance (like pathogenicity, toxicity) or especially obvious and well founded in vernacular classification (e. g., growth forms like “tree, shrub, herb”). Both phylogenetic and artificial classifications rely strongly on descriptive data (the only other data source for phylogeny being the fossil record, which is informative only in very few taxonomic groups).

Phylogenetic research is not limited to creating phylogenetic classifications. Once this has been achieved, descriptive data or against geographical distributions may be mapped against the classification and used to test causal hypotheses about evolutionary processes that developed the current and historic diversity of life.

It must be stressed that neither current knowledge of taxonomic names, nor the associated descriptions of life on earth are anywhere close to satisfactory. The proportion of known to estimated species is very small for most groups (Fig. 1). The failure of biology to identify and name the majority of organisms living on earth is at least partly due to the difficulty in managing descriptive data.

This can be seen, e. g., when descriptions of new species are compared with later revisions which recognize the majority of these purported new species as synonyms of existing species (Fig. 2). If descriptions had been appropriately searchable, this would not have happened.

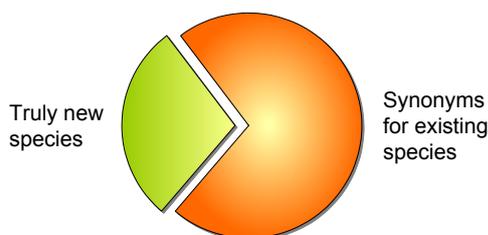


Figure 2. “... mycologists inadvertently redescribe already known species at the rate of about 2.5:1” (Hawksworth 1991).

Several paths are explored in contemporary biodiversity research to improve the efficiency of identification processes. The most important ones are the use

of molecular methods to generate more accurate and partly automated identifications and the use of informatics methods to manage descriptive (morphological, anatomical, chemical, or molecular) data.

A major difficulty with descriptive data is that they are concerned with *two* kinds of diversity: the number and variety of organisms that exist on earth and the number and variety of properties that characterizes these organisms. These are reflected in a diversity of taxonomic names and descriptive terminology, respectively. The process of identification – or rejection of identification and recognition of a new species – typically requires two persons to communicate and reach the same conclusions: one person who created a description associated with a taxon name and another person studying an unidentified object. An exact descriptive terminology is a prerequisite for this. Current terminology, however, has arisen over several centuries, is often specific to its taxonomic domain, and may even be specific to geographic regions or scientific schools. Although every effort should be made to clarify and harmonize terminology, the development of scientific terminology is necessarily a scientific process in itself. In contrast to other fields of biodiversity informatics, the terminology that is used to express descriptive data needs to be defined as data and not as part of the information model. This requires a substantially higher abstraction level for information models that represent terminology and object descriptions compared with other biodiversity information models.

1.3. Other uses of descriptive data

In addition to their relevance for identification and classification, descriptive data are a fundamental documentation of organismic biodiversity on earth and are studied for other reasons as well:

- Descriptive data often contain information that directly relates to potential uses of organisms in medicine, agriculture, or biotechnology. Examples are data about enzymatic activities of microorganisms, data about wood density or antimicrobial secondary metabolites that correlate with the use of timber for specific purposes.
- Information about organism interactions (e. g., pollinators, host-pathogen, or predator-prey; see also p. 30) or the interaction with the environment (e. g., growth of plants in different soils) is relevant for agriculture and to understand ecological networks.
- Conversely, the same comparative approach can aid in the understanding of the function of morphological, enzymatic, behavioral, or other features. Studying the phylogenetic, geographic, or ecological distribution of character states can elucidate the function of characters (“character evolution”). Similarly, character value correlation, perhaps corrected for phylogenetic correlation, will often indicate related functionality.

Furthermore, the concept of descriptive data is not limited to the realm of biology and medicine. Biological descriptive information models have, in fact, repeatedly been used for other areas such as archeology (pottery identification, Louhivuori 1996). The present discussion, however, concentrates on the use and integration of descriptive data in the framework of biodiversity information. A clear analysis of the requirements in biology will hopefully facilitate the development of even more generalized information models for descriptive data.

1.4. Scope, motivation, and constraints of the current work

The current work attempts to apply entity-relational and object-oriented information modeling techniques to descriptive data management and analysis. For the development of information systems, a sequence:

- system planning and definition
- requirement collection and analysis
- generation of initial information models and user views, followed by
- creation of a global model,
- hierarchization and detection of inheritance patterns, and
- normalization

is recommended (Connolly & Begg 2002). For complex systems an *iterative process* of design, refactoring, and redesign is often necessary (Fowler & Scott 2001, Ambler 2003b). The present work is offered at a point of redesign. Several information models exist, two of which (Diversity-Descriptions and SDD) are authored by the author of this thesis (without and with collaborators, respectively):

The *DiversityDescriptions* model – based on entity relationship modeling – is presented in detail (Ch. 7, p. 322 ff). It has been thoroughly tested and improved for about 12 years in applications written by the author (DeltaAccess/DiversityDescriptions, versions 1.0 to 1.9). At the inception of this model stood the question whether data commonly expressed in the DELTA standard can be handled in a relational database. This was established, but some shortcomings of DELTA became more and more obvious over time. Several of these shortcomings could not be addressed immediately because it was desired to keep the model compatible with DELTA import and export; some further shortcomings had to be maintained because the necessary refactoring in the code base of the application would have been too time-consuming.

As a consequence of the limitations of DELTA and the DiversityDescriptions model, a fundamental redesign was started. The new information model was developed in the context of the TDWG *SDD (Structured Descriptive Data)* working group, led by the author, as an XML schema. In 2005, SDD version 1.0 was approved as an international data exchange standard by TDWG (www.tdwg.org). Development is ongoing and version 1.1 was created in spring 2006 and released after testing in March 2007 (for further information on SDD see p. 20 and 388).

Due to time and space constraints, the present thesis can only present the successful relational DiversityDescriptions model and contribute towards a better understanding of “open problems” and unfulfilled requirements. The vision against which the requirements are collection is a general information model for descriptive data, suitable for a wide spectrum of applications (especially identification, generation of natural language reports for monographic works, phylogenetic or other analyses, as well as general knowledge management). To work towards this desired generality, the analysis was deliberately approached from three different angles:

- A data-centric, fundamental analysis trying to explore the general concepts, to recapitulate and reformulate current practices, and to assess how differently named approaches may in fact be related (Ch. 4, p. 36).
- An analysis focusing on the identification processes, methods, and tools. While trying to discuss all major aspects of identification, this analysis tries to distinguish between aspects requiring underlying data structures and aspects that are purely a question of algorithms and presentation (Ch. 5, p. 229).
- A general and high-level use case analysis. This addresses potentially the same requirements as the fundamental analysis, but is performed with the goal of creating a generalization hierarchy of use cases, into which the present as well as future requirements may be integrated (Ch. 6, p. 277). In contrast, the fundamental analysis provides details and examples on data in biological descriptions, but is structured by the order in which concepts refer to each other, rather than a process or usage-centric view.

The work laid down in these chapters forms an explorative and informal requirement analysis in preparation of the next development cycle. All three analyses are explorative and do not fulfill formal criteria that allow direct use in software development. The numbered requirement statements (starting at p. 41) are intended as references for future work, rather than being used to analyze and compare actual information models.

The first requirement analysis is somewhat heterogeneous in the level of detail with which specific problems are discussed. Parts that are studied in particular detail address specific problems raised in the SDD discussions:

- How abstract should the model be?
- Are different models for individual and class descriptions needed?
- Which data types are required?
- Use a character matrix or character state matrix model?
- How to handle quantitative values and statistical measures?
- How to handle original measurements and sample data?
- How to handle deduction of information from higher taxa?
- How to handle characters that can be calculated based on other characters?
- How to handle the relation between broad and narrow concepts of character states?
- How to handle character dependency (and whether two complementary mechanisms (applicable-if, inapplicable-if) are necessary or desirable)?
- Which DELTA features can be omitted or generalized?
- Should the traditional character concept (employed, e. g., in DELTA and NEXUS) be followed or should a “character decomposition model” be embraced?
- How to federate and modularize terminology as well as descriptions?
- How can the terminology be kept concise, while supporting structured extensions to the expressibility? Can “modifiers” contribute to this?
- How to handle secondary classifiers like sex or life cycle stages?

Some of these questions are answered in the following, others are left exposed for subsequent work, and some aspects have not yet been dealt with at all: Details of natural language generation need to be studied, aspects of ontology in the light of current developments in machine-reasoning discussed, and the chapter on identification processes should be complemented by a similar study of processes in phylogenetic analyses.

Being about system *design*, not observations or testable hypotheses, the explorative requirement analyses cannot possibly produce objective results. By necessity, they include an element of discussion and review of available information. The separation between results and discussion commonly expected for a publication in the biological sciences could therefore not be achieved. The final discussion (p. 356 ff) therefore recapitulates the major achievements of existing models, summarizes problems still open, and discusses ways ahead.

Despite being not a typical biological thesis and in spite of efforts to use the language of information science, this work is not a work of computer science. It aims to be truly interdisciplinary. The focus is to analyze, formulate, and transform domain knowledge about biological descriptive data into the terminology of computer and information science. New research in artificial intelligence and data management may provide solutions to some of the requirements analyzed here. Presently, however, methods of information science that have already shown their usefulness and are widely available are preferred over methods that are at the current front of computer science research.

2. Methods

2.1. Explorative requirement analyses

Corresponding to the desire to collect requirements for broad and general models of descriptive data, the requirement analysis aims to be as inclusive as possible. Requirements were collected in the following contexts:

- Fundamental discussions of the processes involved in data collection, aggregation, and management.
- Current practices in information science, comparative data analysis, and statistical analysis in the biosciences.
- Documented data exchange formats for descriptive data (especially NEXUS, DELTA).
- Conceptual models or proposals (e. g., New DELTA, Nemisys/Genisys, or Prometheus).
- Various software applications for identification and management of descriptive data.
- Personal experience, analysis, and user-feedback during twelve years of developing and distributing the DeltaAccess/DiversityDescriptions application (see Ch. 7, p. 322).
- Several years of intensive discussions in the SDD (Structure of Descriptive Data) subgroup of Biodiversity Information Standards/TDWG, the IUBS taxonomic databases working group, www.tdwg.org. The SDD group created an eponymous successor to the DELTA format (see p. 388 in the appendix for a brief history of SDD).

The cost of not focusing on a specific user group that works on a specific taxonomic group with specific methods is that the resulting requirements are neither formal nor complete enough to be used directly for software development. This was not a goal of the current study. Instead, the desired outcome is a set of testing requirements and examples. Independently developed information models that are desired to be general and broad may be checked against these.

2.2. Survey of information models and software

As mentioned above, the collection of requirements for descriptive information models is guided and informed by existing software applications, exchange formats, and conceptual models. Using computers for descriptive data has a long tradition (see Pankhurst 1991) and only a small selection of models could actually be analyzed and discussed. In sequence of decreasing utility for the purpose of extracting requirements for information models, these may be broadly categorized into:

- documented data exchange formats,
- undocumented data exchange formats,
- documented internal data structures of implemented or modeled software applications, and
- software that could be evaluated based on the user interface or published information alone.

The extent to which different models and applications have contributed to the current analysis differs greatly. It was difficult to decide which software to exclude, especially since the requirements in the chapter “Identification methods” (p. 229) have been informed by a great variety of available software. A complementary list of “Systems that could not be adequately studied” (p. 22) may inform readers acquainted with these systems about limits of the present presentation.

To avoid repeating literature citations to software and models throughout the text, the citations in the following sections will be referred to later.

Documented data exchange formats

NEXUS, currently in version 2 (Maddison & al. 1997), is designed for analytical purposes like phylogenetic inference and has very limited support for free-form text. Originally intended only for categorical data, it can now also handle continuous (quantitative) data. It is widely used to analyze DNA or protein sequence data, e. g., using **Mesquite** (Maddison & Maddison

2006) or PAUP (Swofford 1990, 2000). It is also used occasionally by identification software; examples are:

- **ETI Linnaeus 2/IdentifyIT** (Schalk & Heijman 1996, <http://www.eti.uva.nl/Products/Linnaeus.html>) is a locally running PC application based on an authoring system, producing formatted and hyperlinked (but unstructured) text with images and other media resources, similar to web or Wiki pages. Through a “plugin” it offers an identification key for NEXUS-coded categorical data. The results of the key link to the taxonomic descriptions or other information. Using NEXUS is possible because descriptions and key data are completely independent, i. e., the natural language descriptions are created manually rather than being generated on the basis of structured descriptive data (as in DELTA).
- **Visual Key** (Klimov 2001, OConnor & Klimov 2004a) is a web-application running in internet browsers (e. g., Firefox or Internet Explorer). The client-side generated user interface uses only JavaScript, allowing for considerably greater responsiveness to user actions than the client-server logic of most other web identification interfaces. It supports categorical characters/states, images, and interactive images informing the user about names of organism parts under the mouse cursor. It supports a subset of the NEXUS format (including inapplicable, missing, and polymorphic characters) and extends it with an embedded syntax to define figures and interactive image maps inside the character state labels.

DELTA, the “Description Language for Taxonomy” (DELTA) goes back to work by Mike Dallwitz at Canberra University in 1973. DELTA was first published in Dallwitz (1980). The Taxonomic Database Working Group (TDWG, www.tdwg.org) has endorsed the basic directives of DELTA as an international data standard (Dallwitz & Paine 1999, 2005). In contrast to NEXUS, DELTA supports free-form text data and annotations and is suitable for generating natural language descriptions and keys from coded data. More detailed information about the DELTA format may be found in Dallwitz & al. (2000a) and Ch. 5 of Pankhurst (1991). DELTA is probably the most widely used general-purpose format for descriptive data and it is used by several taxonomic software packages. The most well known are:

- The “**CSIRO DELTA package**” (containing Confor, Delfor, and Intkey; Dallwitz 1993a, Dallwitz & al. 2000a). This package has been the focus of DELTA development for many years. Both the exchange format and this software package are often simply called “DELTA”, which may lead to confusion. In this thesis, *DELTA* always refers to the standard; *CSIRO DELTA package* or *software* being used otherwise. Where applicable, the individual applications in the package (Confor, Delfor, and Intkey) are named directly. A new Windows-based editor was published in 2000, called Delta.exe.
- **Pandora** (Pankhurst 1993b, Pankhurst & Pullan 1996, Pankhurst & Pullan 1998, Pankhurst 1998). Pandora is a DOS-based general taxonomic database (including nomenclature, distribution, etc.) that stores descriptive data in a DELTA-like format and interacts with DELTA-based software (especially Pankey, see Pankhurst & Pullan 1996) for data analysis and creation of keys or descriptions.
- **Pankey** (Pankhurst 1998, Pankhurst 2003) is a DOS-based suite of programs for editing (the DEdit DELTA editor), creating printed or online keys, and character analysis.
- **TAXASOFT** (Gouda 1996, Gouda 2001), a DOS-based DELTA editor (originally shareware, now free).
- **DeltaAccess/DiversityDescriptions** (Hagedorn 1997, 2001a, 2005b). A Windows-based database editing and analysis system, in 1997 this was the first graphical Windows editor available for DELTA data. This information model will be documented in detail in this thesis (p. 322).
- **DELIA**, “The DELTA Integrator” (Choo & Spooner 2001, Choo 2002) integrates and manages multiple DELTA data sets (separate projects) within a relational database framework. It provides management and storage operations like basic editing and report generation, backup and restore directly, but uses external DELTA programs like CSIRO DELTA

or CBIT Lucid for most editing and analysis tasks. DELIA itself extends the functionality where necessary, e. g., it supports to move characters and items across DELTA projects and allows new projects to be created from existing ones.

- The NEXUS data editor for Windows (NDE, Page 2001a) has limited support for DELTA data exchange.
- Examples of interactive identification packages importing DELTA data are: **Intkey** (Dallwitz & al. 2000b, local and web, part of CSIRO DELTA, Fig. 132, p. 252), **Online** (only local, part of Pankey), **CBIT Lucid** (see below, local and web), **Navikey 2-2.3** (Bartley & Cross 1999; Java applet using DELTA text files), **Navikey 4** (Neubacher & Rambold 2007a; different source fork from Navikey 3, using DELTA text files, see Fig. 135, p. 253), **WebDelta** (Percudani & al. 2006, Rivetti 1999; Perl script), **PollyClave** (Anonymous 1996, web application in ANSI C), **ActKey** (Brach & Hong Song 2005, server-based Java, Fig. 140, p. 254), **Système d'identification interactive multimédia** (Goujon 2007, Java applet using DELTA text files, only French information available).
- Some identification or editing packages support DELTA indirectly through **DeltaAccess/DiversityDescriptions**. These are: **Identify** (only local, part of DiversityDescriptions, Fig. 131, p. 252), **Navikey 3** (McGillicuddy 2005, web identification, using DeltaAccess databases instead of DELTA text files), **DAP = DeltaAccess Perl** (Cross 1997, Findling 1998a, Fig. 138, p. 254), **DAWI = DeltaAccess Windows** (Findling 1998b), **Diversity-Navigator** (web-based data editor, Neubacher & Rambold 2007b, see Fig. 231, p. 360).
- DELTA is further supported with various restrictions by the general biodiversity management systems **ALICE** (p. 22), **BG-Base** (BG-BASE Inc. 2007), and **BioLink** (p. 22).

New DELTA is a proposed revision of DELTA, addressing criticism and shortcomings of DELTA. The original proposal was Dallwitz & al. (1993), which led to an intensive follow-up discussion (Pankhurst 1993c, Dallwitz 1993b, Kirkbride & Dallwitz 1993, Gouda 1993, and Dallwitz 1993c). The proposal was later updated with minor corrections in Dallwitz & al. (2005) and is complemented by a general requirements survey including information about New DELTA (Dallwitz 2005c).

SDD (Structured Descriptive Data) is a TDWG standard since 2005. SDD is developed in the form of an object-oriented w3c XML-schema (Fallside 2001, Thompson & al. 2001 and Biron & Malhotra 2001) by the international TDWG SDD group. Two versions appeared so far: SDD 1.0 (Hagedorn & al. 2005) and SDD 1.1 (Hagedorn & al. 2006). An older primer and introduction to SDD (Thiele 2003) has recently been updated for version 1.1 (Thiele & Sharp 2006). See p. 388 for a brief history of SDD. SDD is currently supported by:

- **Lucid3** (see above)
- **Electronic Field Guide** (EFG, <http://efg.cs.umb.edu/>) tools; the EFG Key Rendering Suite (Morris & al. 2007) imports and exports SDD.
- **XKey** as described in Gibaja Galindo (2004) is based on the early version 0.5 of SDD.

MEKA (Version 3.1 for Windows, 2003, Meacham 2007) is a dedicated multi-access identification program that is using a state-only model (without a character concept).

IDnature guides are on-line identification tools available at **DiscoverLife** (Pickering 2007). The identification software is called “20q” and uses a proprietary xml format (Ballew & Pickering 2003). DiscoverLife is run by a non-profit organization. The xml format is inadequately studied here because the existence of documentation for this format was discovered only shortly before ending this work.

SLIKS (Guala 2006) and **SAIKS** (Alexander 2006a, 2006b, 2006c, Figs. 143 ff, p. 255 ff), both JavaScript-based web identification keys form a special kind of application because their data exchange format is JavaScript code itself. The deliberately simple format is documented and has been studied. It is a small subset of DELTA, but interesting in its prioritization.

Undocumented open databases and data exchange formats

LIF, the “Lucid Interchange Format” is used by **CBIT Lucid** (Thiele & al. 1998, CBIT 2007a), a software suite with authoring (“builder”) and player software for interactive multi-access keys (see Figs. 133-134, p. 252). Lucid up to version 2 used a text-based LIF format, Lucid3 a new xml-based format (Lucid3 also supports SDD). No documentation seems to be publicly available; an example of LIF 1.1 may be found in Dallwitz (2005c).

- **X: ID** (UBio 2004, Fig. 139, p. 139), an XML/XSLT-based identification system combining a simple character/state model with extensive use of images; uses text-based LIF format (i. e. Lucid version 2) for data interchange (compare Leary & Hagedorn 2004).

XPER (Lebbe 1984, Lebbe & Vignes 1989, Lebbe & al. 1989, Lebbe 1991, Lebbe & Vignes 1998, 2003). The original XPER has a text-based format for the storage of the different object names (taxa, characters, ...) and uses a binary file for the taxonomic descriptions (R. Vignes Lebbe, pers. comm.), the new XPER² replaces the text-based format by an xml-based exchange format. The original XPER format might have been documented to some degree, but the lack of French language skills prevented the author from evaluating the format. The new XPER² xml format is – to the author's knowledge – yet undocumented.

3I interactive key (Internet-accessible Interactive Identification, Dmitriev 2006, 2007) is a package that covers on-line interactive keys and taxonomic revisions (including production of printed monographs). It is distributed under a proprietary open source license. 3I has no known exportable data exchange format, but uses open MS “mdb”-database structures. 3I provides many special features for organism stages, host-parasite interaction, etc.

AditKey (Adit 2004) is a package supporting both dichotomous and multi-access keys. Data exchange is possible through unencrypted MS mdb-database files, which could be studied.

TAXIS 3.5 (Meyke 2004) is an all-encompassing biodiversity information management system. To keep the complexity manageable, it searches for simplicity by providing only the minimal required functionality for all parts. Taxis supports a very simple character + state terminology (optionally with multiple images), a simple state-taxon association without free-form notes or modifiers, and single values or ranges (but neither both, nor range plus extremes) for quantitative values.

PHPKey for lichen identification (Lindh 2003, Lindh & Thor 2004); both the tables and the PHP source code for web identification are reported. The user interface in Swedish is available at <http://www2.artdata.slu.se/Nycklar/knappnal/welcome.html>; an English version linked from there was not functional as of 2007-05-10.

Documented internal data structures of implemented or proposed software applications

- **Data model for the evaluation and characterization of plant genetic resources** (Germeier & Frese 2001).
- **Nemisys** and **Genisys** are conceptual character decomposition models developed by Diederich, Fortuner and Milton (Diederich & al. 1989, Diederich & Milton 1989, 1993a, 1993b, Fortuner 1993, Diederich 1997, Diederich & al. 1997, 1998, 1999, 2000a, 2000b, Fortuner 2002), building on earlier studies by Lebbe (1991). The model was originally developed for nematodes and called “NEMISYS, **N**ematode **I**dentification **S**ystem”); later the name “GENISYS, **G**eneral **I**dentification **S**ystem” was preferred. The ideas of the authors developed over time so that not all publications describe exactly the same concepts. A broad general concept will here be referred to as the Nemisys/Genisys model. For details on basic properties see p. 62, for specifics of the decomposition of characters into a “structure” (i. e. part, physical component) and “property” dimension see p. 117, for modifiers see p. 195.
- The **Prometheus description model** (= “Prometheus II project”) is described, e. g., in Cannon & McDonald (2001), Paterson & al. (2004) and Pullan & al. (2005). The publications describe mainly high level features of the model; no actual implementation and data structures could be

studied by the present author. For details on character decomposition see p. 118, for modifiers see p. 196.

Software that could be studied based on publications or the user interface alone

BAOBAB (conceptual model) and **ALICE** (derived implemented system; Allkin & White 1988, Allkin 1989a, Allkin 1989b, Allkin 1996, White & al. 1993) are relational information models (including taxonomic and descriptive information) that could not be studied in detail.

BIKEY, a software package (Lobanov & al. 1996, Lobanov & Dianov 1999) containing especially **PICKEY** (e. g., Schilowa (undated)), a picture-centric multi-access key. Only the user interface of **PICKEY** was studied.

FRIDA (Nimis 2007); **FRIDA** stands for “FRiendly IdentificAtion” (pers. comm. P. L. Nimis) and is a software package using an Oracle™ database to produce web-based biodiversity identification keys and eLearning modules. It is patented in 2002 by University of Trieste (www.dryades.eu) and yet undocumented (pers. comm. S. Martellos).

Mycokey (www.mycokoy.com), **LichenLand** (multi-access key at http://mgd.nacse.org/cgi-bin/hyperSQL_gateway?/hyperSQL/lichenland/hsq/nu3col.soph.hsql), and many other keys on the internet.

USDA SBML keys may be studied in action (Castlebury & Farr 2002, Fig. 127 on p. 248 and Hernandez & al. 2004) and the design criteria are described in Farr (2006), but the software and information model could not be studied.

WebKey-x (Kirejtshuk & al. 2005) is a new project or program, partly by the authors of **BIKEY**. The web project page is entirely in Russian, but some information for non Russian speakers is available at Lobanov & al. (2005, partially in English).

XKey (Gibaja Galindo 2004, Delgado Calvo-Flores & al. 2005) is a new Spanish system. Some of the approaches and data structures could be inferred from the publications, but further analysis is desirable.

Several commercial digital publications (using protected, proprietary information models) of plant (Lauber & Wagner 2001, Seybold & al. 2001 & 2004, Götz 2003) and fungal (Böhmer & Wohanka 2002) floras for Central Europe.

Systems that could not be adequately studied

The following list of systems not studied completes the attempt to survey available software. It is intended to document the limits of the current analysis by highlighting software that potentially could increase the understanding of descriptive information models if it had been studied. Future studies by those who have access to these systems are most welcome.

BioLink is a general, software-component-based system Australian biodiversity data management system (Shattuck & Fitzsimmons 2000). **BioLink** is probably the only system so far that has implemented inheritance and compilation of descriptive data up and down the taxonomic hierarchy. Originally the system was sold, then freely available for several years. As of 2007-04-15 most web pages at www.biolink.csiro.au have been removed, the front page informing that “**BioLink** is currently unavailable”. According to S. Shattuck (pers. comm. 2007), the system remains in use internally and for a limited number of previous users, but development is limited to bug fixes and it is no longer actively advertised.

BioLoMICS (Biological Data Manager for Identification Classification & Statistics, BioAware 2007) is especially interesting because it considers descriptive data in a very general scope, dealing with morphological, physiological, biochemical, chemical, chromatographic, electrophoretic, as well as molecular sequence data. It uses similarity methods for identification, and seems to have a very flexible system of performing combined analyses on widely different data types. It is commercial, for MS Windows, and data can be published to CD-ROM or Internet.

CABIKey (some information in White & Scott 1994 and White & Sandlant 1998; current status unknown, all earlier web links disappeared) and **TAXAKEY** (e. g., Blackman & al. 1997, current status of software unknown). According to White & Scott (1994), CABIKey is a multi-access key based on a proprietary exchange format; the primary difference to Intkey or Onlin7 being that images may stand on their own, replacing rather than supplementing textual statements.

MANTIS (Naskrecki 2007) is primarily a specimen collection and image/sound manager. It supports nomenclatural information, taxonomic hierarchy, literature citations, and loan management, but also has some support for descriptive data beyond the image support. It supports the creation of species pages based on database content (free-form text for diagnosis, description, and natural history, pest control details) and images. It is explicitly designed to allow the recording of organism interactions (host-parasite, predator-prey, etc.). Based on available screenshots it also seems to support numeric measurements. The extent of descriptive support could not fully be gathered from the available documentation.

Platypus (Houston & al. 2002) a broad Australian biodiversity data management system, the degree of support for descriptive data could not be studied.

PollyClave 2 has been announced on <http://www.botany.utoronto.ca/faculty/dickinson/main.html> and is described by Lindh (2003) as being functional. However, as of 2007-04 only broken links and a dysfunctional prototype under <http://www.botany.utoronto.ca/faculty/dickinson/identify.html> could be found.

SYNTAX is an all-encompassing German biodiversity data management system (see, e. g., Boos 1992, Hoppe 1998, Hoppe & al. 1999, SysTax 2004, Hoppe & al. 2004, and Hoppe & al. 2007). The support for descriptive data is relatively recent (J. Hoppe, pers. comm. 2007) and could not yet be studied. An interactive identification component “TaxDet” is in prototype stage.

XID (Exeter Software: undated, XID Services 2007), a commercial software system; featured in Pennisi (1994), but without information usable for the present analysis.

2.3. UML use cases

Use case analysis is a method to capture high-level requirements for information models and implemented systems. Use case modeling is used in different ways and UML does not specify the amount of detail that has to be provided (Fowler & Scott 2001). In software development processes, use cases may be used to define the milestones of development progress. In this case it is essential that the diagrams are accompanied by step-by-step descriptions of the scenario (or “stories”) for each use case, and variations or extensions to the scenario. In this thesis, focusing on high-level requirements for information models, use case analysis (p. 277) is used to create a broad conceptual overview of descriptive data systems, providing only a few detailed scenarios. The analysis relies on a short narrative text accompanying the diagrams, intuitive use case names, a relatively high number of specialized use cases combined by «include» and «extend» relations.

To counter the high number of specialized use cases, a use case generalization hierarchy is developed (Fig. 3), often starting at *abstract* use cases. An abstract use case cannot actually be used, but summarizes the common behavior of more specialized use cases. Its name is formatted in italic letters (other UML dialects use “{abstract}” after the name). The goal is to formalize a decomposition of a high level task into rigorously defined subtasks for which software may be more easily written.

Use cases are presented in the form of UML use case diagrams (UML version 1.4, OMG 2001). Use case diagrams changed substantially after UML version 1.2 (which used «uses» and «extends» relations). The available UML modeling tool (Microsoft Visio 10) supports only UML 1.2 and several shapes had to be reprogrammed to function as UML 1.4 shapes. As a consequence, however, it was not possible to use UML syntax checking for use cases.

The following references were studied for current usage recommendations of UML (especially in the context of database design): Fowler & Scott (2001), Naiburg & Maksimchuk (2001), and Connolly & Begg (2002). In addition, most of the guidelines laid out in Ambler (2003a) were followed (e. g., the UML “system boundary” is omitted from use case diagrams).

Two important concepts are the *use case* and the *actor*. Use cases are coherent units of functionality provided by a system or a part of it, and actors are human or software agents which execute it. A simple example is: “taxonomist identifies specimen”.

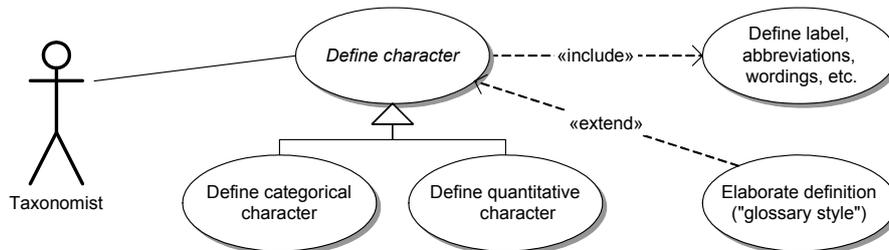


Figure 3. Elements of UML use case diagrams. Actors (e. g., “Taxonomist”) perform use case actions (“Define character”). Relations between use case symbols may be generalizations (solid lines), connecting a general use case (to which the triangle points) with a specialized one, or dependency relations (dashed lines), expressing included and extending use cases.

Besides generalizations, two further relationships are expressed in the use case diagrams: «include» and «extend». To «include» another use case means that the entire functionality of the other use case is always part of the including use case. The case that the included use case is an optional elaboration is expressed through «extend» (reversing the direction of the arrow to indicate the dependency).

Many UML dialects (e. g., Naiburg & Maksimchuk 2001) distinguish between system and business use cases (Fig. 4). Use cases in the present analysis are always general (or conceptual). Unfortunately, the general and system use case icons are indistinguishable so that such a usage note is required.

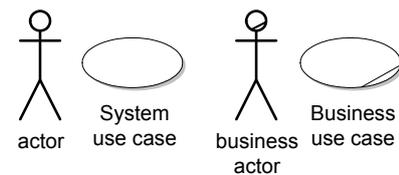


Figure 4. Some UML use case dialects may use different symbols for system and business actors. In the present work the first icon always signifies a general, undifferentiated use case.

2.4. UML static class diagrams and ER models

UML static class diagrams are used during many general and abstract discussions to illustrate issues of generalization, composition, and dependency. Classes are represented by rectangles and connected by **associations** and **composition relationships** (Fig. 5). The role of the relationship is presented on the branch that attaches to the entity with which the role starts. Different modeling traditions exist for the role naming pattern at the end of association and composition relationships (UML does not define a standard here). Fowler & Scott (2001) recommend to use a noun phrase and to omit the role name when possible. In this discussion, however, the more conventional verb-phrases for roles (“is applicable to”, “controls”, etc.) have been used. The **cardinality** (or “multiplicity”, Table 1) of association and composition relationships is given at each end next to the role name. **Generalization relationships** are indicated with a white triangle (Fig. 5 left, identical to those used in the use case diagrams, Fig. 3 ff).

UML class representations may have up to three sections separated by horizontal lines: class name, attribute list, and operations (or “methods”). In the thesis the operations section is almost universally suppressed; the attributes section is suppressed where irrelevant to the discussion.

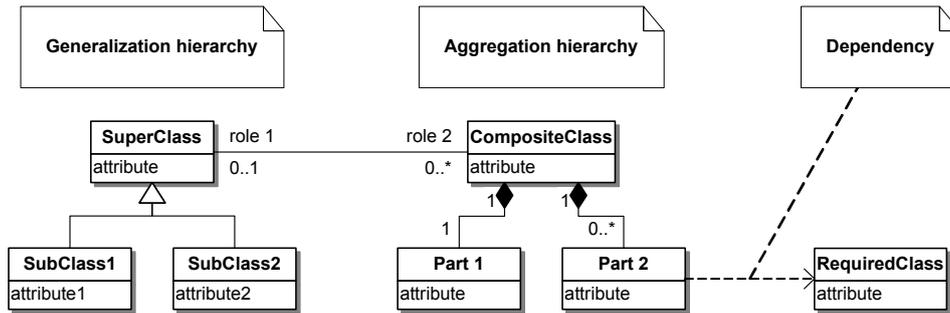


Figure 5. Relationships in UML static class diagrams. On the left side two subclasses are derived from a superclass. The triangle has the form of an arrow pointing from specialized to general class. Subclasses inherit attributes and methods of the superclass. – In the middle, a class is connected with two component classes (Part 1, Part 2). Composition associations are indicated by black diamonds. Part 1 must occur (1:1 relationship), whereas Part 2 may be missing, occurring once or multiple times (1:0..* relationship; compare Table 1). Component classes may not belong to other classes. – A class association with multiplicities and role names is shown between SuperClass and CompositeClass in the center. Role names for association ends are used only in the database UML models. – On the right side an additional class is connected with a dashed arrow indicating a dependency relationship (i. e., Part 2 depends on RequiredClass).

Table 1. Common multiplicity indicators (i. e., ranges of allowable cardinalities) used in UML class associations.

Multiplicity code	Description	Association is:
0..1	Zero or one	Optional
1	Exactly one	Required
0..* (or "*" alone)	Zero or more	Optional
1..*	At least one	Required
n (for example: 6)	Specific number	Required

The entity relationship model for DiversityDescriptions (p. 322) is also presented using UML static class diagrams. In parallel to the general acceptance of UML for general software design, this has become common practice in database modeling (compare Connolly & Begg 2002 and Naiburg & Maksimchuk 2001). **Entity types** are represented by class diagrams (limited to persistent classes, showing only persistent attributes, and having the “operations section” of the UML class icon suppressed).

Both a logical model (using UML) and a physical model (using a tabular documentation) are presented. The ER diagrams for the logical model are generated with Microsoft Visio for Enterprise Architects; the tabular documentation with DiversityModelDocumenter version 2.6 (Hagedorn 2004a). Because of the extensive discussion above, which ends in requirements beyond those fulfilled by DiversityDescriptions, version 1.9, no conceptual model is presented.

In the logical model, no explicit assumptions on referential integrity are made. Implicitly, all associations and compositions are assumed to be protected by referential integrity. Cascading updates are implied where primary key values are natural keys that may change during edits (i. e., they are not system keys). Furthermore, composition (symbolized by a black diamond at the parent class) indicates that instances of the component classes can only exist as part of instances of the parent class. For these relationships cascading deletes may be assumed.

In the logical model of DiversityDescriptions the data types for attributes are indicated using a small selection of generic data types (Naiburg & Maksimchuk 2001, Table 2).

Table 2. Generic data types used in ER-/UML-class diagrams.

Generic data type	Description
Boolean	Logical values "true" and "false"; may be Null if not defined as required.
Byte	Positive integer numbers in the range 0-255
Date	Date, time, or data-with-time values
Double	Real (floating-point) numbers with 15-16 digits of precision
Integer	Integer numbers with 32 bit precision (= "Long" in older programming languages which consider int/integer as 16 bit).
Single	Real (floating-point) numbers with 7 digits of precision
String	An unlimited number of Unicode characters

2.5. Abbreviations

ABCD	Access to Biological Collections Data (TDWG)	NEXUS	(not an abbreviation: phylogenetic data exchange standard, p. 18)
AFLP	Amplified Fragment Length Polymorphism (a molecular identification technique)	NLP	Natural Language Processing (machine reasoning from human-written text)
ANOVA	Analysis of Variance	OCR	Optical character recognition
CBIT	Centre for Biological Information Technology, Australia	OMG	Object Management Group
Ch.	Chapter	OOP	Object-oriented programming
CDEFD	Common Data structure for European Floristic Databases, a concluded European concerted action project	PAUP	Phylogenetic Analysis Using Parsimony, a NEXUS-based software package.
CSIRO	Commonwealth Scientific and Industrial Research Organisation, Australia	PDF	Portable Document Format (proprietary format by Adobe)
DBMS	Database Management System	RAPD	Random Amplified Polymorphic DNA (a molecular identification technique)
DDBJ	DNA Data Bank of Japan	RDF	Resource Description Framework; a w3c-standard for resource metadata suitable for ontological reasoners.
DELTA	Descriptive Language for Taxonomy (p. 19)	RFLP	Restriction Fragment Length Polymorphism (a molecular identification technique)
DNA	Deoxyribonucleic Acid	SDD	Structure of Descriptive Data (a TDWG interest group) and the resulting standard (Structured Descriptive Data, p. 20)
EMBL	European Molecular Biology Laboratory	SQL	Structured Query Language, a language for defining, manipulating, and querying relational databases
ER	Entity-relationship (... model, ... diagram, etc.)	STR	Short Tandem Repeat (a molecular identification technique)
Fig.	Figure	TDWG	Taxonomic Databases Working Group (www.tdwg.org)
GBIF	Global Biodiversity Information Facility; www.gbif.org	UML	Unified Modeling Language, a modeling standard of the OMG
GLOPP	Global Plant Pathogen Index	URI	Universal Resource Identifier (including universal resource names, URNs)
GUID	Globally Unique Identifier	URL	Uniform Resource Locator (the most common type of URI, e. g., "http://x.net")
ICBN	International Code of Botanical Nomenclature	W3C	Worldwide Web Consortium ("www" = "w3")
ICZN	International Code of Zoological Nomenclature / International Commission on Zoological Nomenclature	WIKI	(not an abbreviation: name of a class of internet collaboration tools, Hawaiian for "simple")
ID	Identifier, i. e. a number or code uniquely identifying an object	XML	Extensible Markup Language
IPR	Intellectual Property Rights	XPS	XML Paper Specification
IT	Information Technology		
IUBS	International Union of Biological Sciences		
JET	Joint Engine Technology, a database technology developed by Microsoft		
LIF	Lucid Interchange format, used by CBIT Lucid programs (p. 21)		
NCBI	National Center for Biotechnology Information, USA		

3. Selected definitions

A major problem with existing generalized or specific description models is that many terms are highly overloaded with multiple meanings and various models often define terms in new ways. This section tries to clarify how terms like *descriptive data*, *character*, or *object parts/structures* are used in this thesis. It also provides an overview of the current usage of additional terms in information science and descriptive information models (Table 3, p. 34).

Some further definitions are provided throughout the text, especially for *character applicability rules* (p. 76), *modifiers* (p. 191), *secondary classifiers* (p. 220), and *ontological* versus *operational terminology* (p. 281). A review of general dictionary definitions of “feature”, “attribute”, and “property” is provided in Table 44 (p. 164), of “identification” in Table 54 (p. 229). Fundamental terms of information models and UML usage have already been discussed in the sections “UML use cases”, p. 23 and “UML static class diagrams and ER models”, p. 24.

3.1. Descriptive data in the context of biodiversity data

Definition of ‘descriptive data’

The term “descriptive data” is used in various disciplines, including geographic information systems and medicine. In biodiversity informatics and taxonomy it has been more or less informally used by various authors since at least 1980. No actual definition could be found and, although the term is generally intuitively understood, it is worth attempting to define:

Descriptive data are data which inform about repeatably observable, intrinsic properties of an entity (components, individuals, populations, or classes).

The entities described in biology are *individual* organisms, *components* or *parts* of them (especially in paleontology), or *classes* (e. g., populations, species, or other taxa). *Individuals* may be organisms observed in the field, or living or dead specimens in natural history collections. These may be identified (have been assigned a taxon name) or not. Observations on living or dead collection objects can normally be repeated, observations on free-living organisms can be repeated if they have been labeled by some means.

A *class description* consists of generalized statements that apply to all individuals in the class. Although a fundamental philosophical difference exists between individuals and classes, the difference in the case of descriptions is remarkably small. Often within an individual organism, components (e. g., leaves) are variable and occur many times. As a result, generalizations, sampling, and statistical techniques are required in single organisms as well as in classes.

Repeatability of observations should be understood as “potential repeatability”. Observations may not be repeatable in individual objects:

- if the object is unlikely to be encountered again (example: birds observed during flight),
- if the observation method destroys the feature (or even the entire specimen) that is being observed (example: measuring total carbon content),
- if the observation depends on a point in development (example: heartbeat rate after birth),
- if the feature depends on the preservation stage of preserved objects (examples: flower of herbarium specimens may loose or change their color with drying; molecular features may deteriorate with time, e. g., enzymes fast, DNA slower).

Data are *intrinsic* (or “*inherent*”) *properties* if they are observable directly on the object, at least at a certain point in time and under certain conditions. Data that are attached to the object only by the method of handling or processing it (collection data, history data, management data on preparation, transfer, etc.), or potential uses (in agriculture, medicine, etc.) are not observable on the object itself and considered “*extrinsic data*” here. Such data may, e. g., be considered manage-

ment, history, event, or metadata and belong to other components in a biodiversity framework concept (see below).

The values of properties may occur only with a given probability, reflecting population polymorphism, environmental factors, or methodological problems; the frequency information reflecting this is part of the description itself.

Descriptive data as defined here are neither limited to morpho-anatomical data nor to observations that can be made on preserved, dead objects. However, in many groups of organisms, these two subsets are of special importance. The visually observable morphological or anatomical data are important because they:

- are often easily observable (without technical help, or using widely available instruments like a light microscope),
- are easily processed and memorized by human beings (dogs would certainly prefer olfactory identification keys ...),
- were more suitable to paper-based publishing techniques (for example, although bird or insect songs are easily recognized by most humans, and recording techniques were widely available, the lack of cost-efficient publishing techniques drastically limited their use in biology prior to the digital age),
- have a long history of use in biology, reflected in a huge knowledge base.

Similarly, descriptive data that can be studied in preserved dead objects are of special relevance to taxonomy due to the nature of the taxonomic type specimen concept (roughly, this is the principle that any concept of a taxon name must contain a designated particular specimen).

An information model for descriptive data should not be limited by a preference for morpho-anatomical data. Chemical, physiological, genetical, or molecular descriptive data (biosequences, AFLP or STR patterns, etc.) as well as behavioral data (including sound, video) may be equally important in certain groups of organisms. Techniques in biodiversity research are currently undergoing drastic changes. For example, in the future, hand-held olfactory or DNA-based analyzers may become generally available and usable even in the field, just as simple recording and analysis methods for digital audio and video streams already are today.

Biodiversity ‘framework concepts’

Implicit in the attempt to study descriptive data separately from other kinds of biodiversity-related data is an understanding that biodiversity data can be structured into multiple, relatively independent knowledge domains. Examples of these are taxonomic nomenclature and concept synonymy, taxon collection or observation data, collection storage and management data, or referencing and indexing information. Each domain may be represented by its own information model and by specialized software components, which remain relatively isolated and interact only through defined interfaces. The categorization of information into knowledge domains may be viewed as a high-level ontology. If the dependencies are analyzed further and abstract interfaces are formally defined, a component framework is created.

Some information models for biodiversity data do not take this approach. They view the entire area of biodiversity research as a single knowledge domain covered by a coherent, global information model (compare “Survey of information models and software”, p. 18 below). The choice between a coherent information model and a component framework is based on a trade-off. A coherent model optimizes the sharing of data and program code at the expense of increased complexity (or, alternatively, oversimplification of the components of the model).

Contemporary software engineering methodology (e. g., Bruegge & Dutoit 2004) has shown that a component framework model reduces the complexity, simplifies design and – perhaps even more importantly – system reanalysis and rearrangement (“refactoring”). Each software component (and the corresponding knowledge domain) may be analyzed independently, treating other

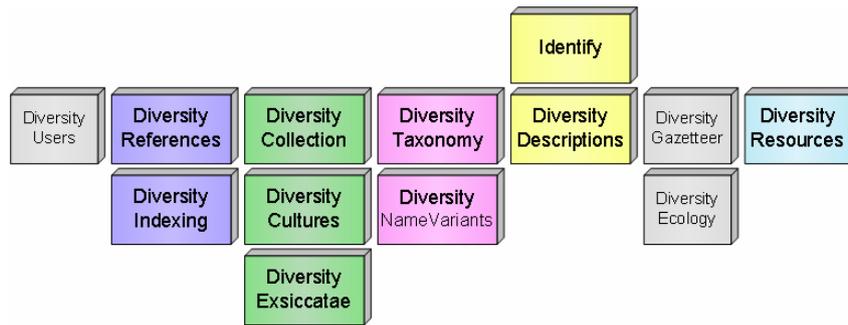


Figure 6. Potential DiversityWorkbench components grouped by similarity. Except for Diversity-Cultures, work on all components has started. The models are, however, of different quality and revision status (from Hagedorn 2002d).

components as a “black box” that is represented through well-defined interfaces. Software components that observe the interface definitions are not affected by internal changes in other components they depend upon. This isolation may come at the expense of duplicating a limited amount of secondary data and program code, which may be mitigated by careful reuse of class libraries containing common functionality. The problem of finding the best possible software decomposition is a current research topic in software engineering, called “Aspect Oriented Design (AOD)” (e. g., Noda & Kishi 1999).

Berendsohn & al. (1996b) were among the first to call for a framework for biological objects. Framework concepts are implicit to several biodiversity software developments, but one of the most extensive attempts to develop such a framework is the “DiversityWorkbench” component framework (Hagedorn 2002d, Hagedorn & al. 2002, Rambold & al. 2003; Figs. 6-7). The present discussion of descriptive data will often assume a component framework like this. It does not, however, depend on any specific one.

A major advantage of a component framework is that components may be developed independently, simplifying international collaborations. A prerequisite for this is, however, that framework and interface definitions are well architected and formally defined. No existing framework concept has achieved this so far. As a step, TDWG has recently formed an “Architecture Group (TAG)” that aims to develop a high-level ontology (see wiki.tdwg.org/twiki/bin/view/TAG/).

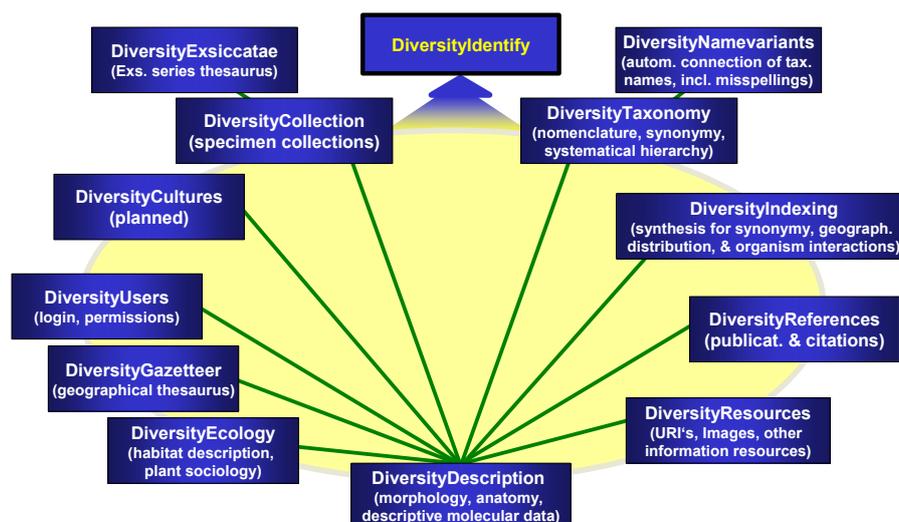


Figure 7. Relation of DiversityWorkbench components with DiversityDescriptions. Diversity-Identify identifies organisms based on features, geographical distribution, and organism interactions (e. g., host plants).

Ambiguous or border-line cases of 'descriptive data'

In the light of the general definition of descriptive data given above, some data are problematic in that they may be considered belonging to descriptions, to another major framework component like taxonomy or specimen collections, or they may even be handled best by a new, specialized component. This is an area for future research. Specific examples of such cases are:

- Spatial data (e. g., geographical distribution)
- Temporal data (e. g., phenological descriptions)
- Organism interactions (pollination, host-pathogen relations, predator-prey, etc.).

If such data are collected in a descriptive data application together with truly descriptive data, these data will often be recorded for pragmatic reasons as if they were descriptive data. They are therefore briefly discussed here.

Spatial data are on the borderline between collection data and descriptive data. They can theoretically be observed directly and synthetic distribution ranges based on the observation of many individuals are useful in identification (at least to identify specimens not suspected to be neobiota). However, in most organism groups the usual workflow involves a separation of specimen collection and specimen study. As a result, at the time when most descriptive data are being studied, spatial information is available only as metadata of the collection process. It is therefore, conventionally handled in specimen collection management systems.

On the other hand, these collection management systems lack the ability to express synthetic, generalized statements summarizing the geographical distribution ("chorology") of higher taxa. Such knowledge may be handled as hierarchical categories with the features generally available in descriptive data systems, or through a specialized component for "taxonomic checklists".

The current lack of integration between individual information and synthetic, generalized statements is worrisome. However, it also seems unwise to replicate in descriptive systems geographical abilities already present in collection or observation management databases (including interoperability with gazetteers or GIS applications). To facilitate global information exchange GBIF and TDWG currently jointly develop XML-based (Bray & al. 1998, Bray & al. 2004a, Bray & al. 2004b) exchange formats to automatically query the geographic (and other) information from collection and observation databases (DiGIR, DiGIR 2005, ABCD, Berendsohn 2005). Once experience has been gained with these methods, the question of integration into descriptive systems should be asked again.

A very similar case is that of **temporal data** like the point in time and duration of plant flowering or insect pupation (phenological data). Any observation is a point in time, but the features of interest are durations and cyclical temporal information (daily, yearly). A large body of point observations is required to allow these generalizations. For some features the information can be derived indirectly through observing the state of collected material and combining this with the collection date. The statistics of such observations are weak, however, since the data collection is non-random (e. g., plants with only immature early flowers are rarely collected).

The inclusion into descriptive data sets is slightly more complicated for temporal than for spatial data. Firstly, whereas geographical distribution is based on the entire organism, many temporal (phenological) data are related to specific features, parts, or life-form stages of organisms. This creates a specific dependence on the terminology for features, parts, or stages used in the descriptive database. Secondly, interest in temporal development is not as widespread among taxonomists as interest in geographical distribution, and consequently, existing collection or observation systems usually have no support for temporal (phenological) data. This will often make it desirable to include such data in a descriptive data system for pragmatic reasons. Finally, handling temporal information satisfactorily requires "date/time" data types not found in any current descriptive data application.

Organism interactions (parasitism, predation, mutualistic symbiosis like mycorrhizae, lichen formation, pollination, seed dispersal, etc.) or ecological habitat or abiotic substrates preferences are a third problematic area. In general, the data pertaining to the surroundings of a collected

specimen may be considered descriptive behavioral information. However, some information is routinely stored in collection or observation databases. Examples are the preference of herbivorous insects for certain plants (which are not collected with the insect), the host plant identification of a pathogenic fungus (the identification of which may no longer be possible using the fragments collected together with the fungus), the soil, climate, and other habitat data. Other information (for example, pollination interactions) is present only in specialized observation databases. Thus, a substantial amount of interpretation is necessary to aggregate and generalize these data and remove chance effects or scientifically false reports. In the case of homogeneous, randomly collected data, statistical methods can automate this process. However, normally the amount of available data is limited so that published literature data without voucher specimens or historical collections have to be used in addition to information with known circumstances of data collection. This requires human intervention and experience.

The synthesis and interpretation of all available organism interaction data into a coherent picture for a class of organisms is a separate piece of scientific knowledge that is on another level than the original observation. The outline of a model for plant-animal-interactions covering data capture both on the original data and synthesis level was published in Théry & al. (1998). Modeling is complicated by the fact that interactions themselves have a geographical distribution (an interaction may occur in one region, but not in another). In the GLOPP project (Global Plant Pathogen Index, Hagedorn & al. 2000, Hagedorn 2000a, Hagedorn & al. 2001, Piepenbring & al. 2001, Hagedorn 2002a, Hagedorn & al. 2003a, 2003b, Piepenbring & al. 2003) organism interactions have been modeled on two levels (original data and interpretation) in the DiversityIndexing (Hagedorn 2001c) component of the DiversityWorkbench. DiversityIndexing supports collecting and interpreting data for any combination of two organisms and a geographical area. Supported data sources are printed or digital publications, physical specimens, and digital specimens or observation metadata. The concept of DiversityIndexing needs further development. Its integration into a more general descriptive information model is probably desirable (not only organism interactions, but also certain morphological or phenological data depend on geography!) and the current thesis may provide some groundwork for this task.

Note that conversely, for practical reasons specimen collection databases will often contain data like flower color or tree height that in the framework model unambiguously belongs to descriptive data. This is a result of the workflow in most biodiversity disciplines that depends on descriptive data being observable on dead, preserved specimens. Features that are either not observable because only part of the material has been collected (e. g., “height of trees”) or that are perishable in preserved voucher specimens (e. g., many blue flower colors) are often recorded together with true collection data on the label text of specimens or as fieldbook notes. Specimen collection management systems will often provide for the capture of such collection-related descriptive information (see ABCD schema, UnitDataType/UnitMeasurements and UnitDataType/UnitFacts, Berendsohn 2005). In contrast to the spatial, temporal, or organism interaction cases mentioned above, this is not a fundamental problem. It is possible to use a descriptive data component within the collection system for “morphological field descriptors” and “ecological descriptors” as proposed, e. g., in the European reference model for collections (Berendsohn & al. 1996a; Berendsohn & al. 1999). However, until both the specimen collection and descriptive data components are sufficiently developed and tested to attempt such a tight integration, it will be a pragmatic choice to duplicate selected data in several components.

3.2. The term ‘character’

Some discussion exists about the correct use of the term *character*. Citing Colless (1985), the authors of the Prometheus description model (p. 21), prefer the term *description element* over *character* (Table 3, p. 34). An earlier occurrence of this term is Lebbe & Vignes (1998). Colless distinguishes three primary uses of “character”:

- Denoting a variable definition (e. g., “wing color”),
- denoting variable data (e. g., something “has brown wings”; i. e. “attribute” in DELTA), and
- denoting a part (“wings”).

The first two uses distinguish whether the term refers to an abstract variable concept, or also to the case where the variable is filled (instantiated) with a value. These uses are clearly related and it is customary to use the same term for these in other disciplines as well. In computer science a variable or class attribute primarily denotes the abstract concept (e. g., symbolic variable, storage location), but the same terms will be used when referring to these concepts in the context of an instantiated object that has values. For example, in UML the “class attributes” are still called “attribute” in an object (i. e., an instance of a class). The attribute values (or data) are distinguished, but no separate terms are introduced for class and instance attributes.

The third usage noted by Colless seems to be more worrisome. However, the examples given by Colless resolve to natural language expressions implying that the character is the *presence* of a part, not the part itself. A statement like “petiole is an important character” implies the property values *present* or *absent*; it is similar to a statement like “diameter is an important character”, where the part (e. g., trunk) and method (e. g., measured using circumference in breast-height) may be implied.

Another analysis of the term “character” was performed by Inglis (1991). Coming from a cladistic perspective, he is not concerned with the distinction between variable-concept and variable-data, but argues that distinctions between usage in systematics and taxonomy need to be made. Inglis restricts character to a very narrow “original” diagnostic sense, essentially as a diagnostic character of a specific taxon. The term thus has meaning only relative to a specific taxon and specific homology assumptions. Inglis introduces replacement terms for other senses (including “homology avatar” for a broader sense of character, and “homolostratum” for a broader sense of character state). The usefulness of these terms when studying the theory of taxonomy and systematics cannot be assessed here, but it is clear that these terms have not found broader acceptance so far (as of 2006-05-20, Google reports no occurrences for both terms mentioned).

The argument that because of such definitional problems the term character “has lost most of its meaning and value” and needs to be replaced with “description element” (Pullan & al. 2005) is not accepted here. The authors of all other description models studied (DELTA, p. 19; SDD, p. 20, Nemisys/Genisys, p. 21, DiversityDescriptions, p. 322, CBIT Lucid3, p. 21, XPER & XPER², p. 21, and NEXUS, p. 18) use “character” in an almost identical sense, i. e., as a defined variable, expressing an object feature that can be recorded in one of the data types supported by the information model (e. g., text, categorical, quantitative, or perhaps complex data). Colless (1985) himself states that the definitions of character are related and that differences cause little difficulty in practical studies. The distinctions made by Colless are relevant and fully accepted. However, it is disputed that terms that were ever involved in ambiguous usage have to be replaced. Colless himself proposes “*character variable*” for the first sense, which (in addition to “*character definition*”, having a slightly different perspective) is accepted here. The proposal by Colless to reserve the term “attribute” for the character data case (a usage that DELTA enacts) appears undesirable in the light of the use of the term in software engineering (compare Table 3, p. 34) and is not followed here. Instead the terms “*character data*”, “*character value*” or, where referring to a single element, “*character data element*” is used. In most cases, it was found sufficient to use the unqualified “character”, because the argument would apply both to the abstract and the concrete concept. The present study does not pretend to study character theory, but to clarify the pragmatic and operational perspective used here, the following definitions are offered:

Character variable or definition: *a concept for which an observation or measurement method has been defined such that repeatable results can be obtained in a defined format. The observation or measurement method includes issues of object constraints and conditions like stage, object parts to study, instrumentation, operating instructions, and data conversion instructions (see p. 123 for details). The repeatability of the results does not imply identity: the*

variation of repeated quantitative numeric values and repeated free-form text statements by different observers require different comparison problems, but do not differ in principle.

Character data or value: *The results of applying the measurement method defined for a character variable. A character value may be a single measurement or the results of an aggregation method. Examples for the latter case are a set of statistical measure or a set of measurement values.*

Character state: *A special term used for values (categories) of categorical variable. Character states must be defined (character state definition) and are used in character data (character state data).*

Inglis (1991) argues that in English a distinction exists between feature and character, and criticizes the English translation of Ax (1984) for its use of “character”, noticing that Ax used the German term “Merkmal”. Interestingly, in German no such distinction between feature and character exists, “Merkmal” being used for both (but property = “Eigenschaft” in German exists). The current practice of the use of “character” suggests that the distinction in actual use of current English is perhaps weaker than assumed by Inglis. However, the term “feature” may be a good substitute for “character” when used with operational definitions as given above. Doing so has been proposed by K. Thiele in the SDD discussions (see Table 3, p. 34). See also Rohlf (1993) for the preference of “feature” (and “operational homologies”) over “character” in automated computer vision.

3.3. Terms for ‘object parts’

Physical parts of objects, which can be distinguished and named individually, are a major foundation for morphological descriptive data. These parts often form a *compositional* or *structural* hierarchy, where parts consist of further parts. The preferred terms for parts vary strongly (Table 3, p. 34).

The preferred term in character decomposition models (Nemisys/Genisys, p. 21, or Prometheus description model, p. 21) is “structure”. While it is undoubtedly appropriate to refer to a “structural hierarchy”, the term “structure” in the understanding of the author is related to composition, not to part. According to the Collins English Dictionary (CED 1992), structure is defined as: “1. a complex construction or entity. 2. the arrangement and interrelationship of parts in a construction, such as a building. 3. the manner of construction or organization: the structure of society. 4. Biology: morphology; form. 5. Chemistry: the arrangement of atoms in a molecule of a chemical compound. 6. Geology: the way in which a mineral, rock, [...] etc., is made up of its component parts [...].”

Clearly, when allowing going down to atomic particles, all relevant parts in biology are in themselves structures again. However, this results in using the same term both for a composition and a component. In the context of the present discussions, the perspective implied by using “structure” seems generally to be wrong. Instead of viewing the relation of object parts with a larger entity, the fact that the parts in themselves are structured composition is highlighted. For this reason, the term “structure” has been avoided in this thesis.

Inglis (1991) prefers the term “component” over “structure”. In this thesis, both “object part” and “component” are used interchangeably.

3.4. Comparison of current usage of terms

Table 3. Overview of usage of terms in descriptive information models and computer science.

	Class ¹ of objects	Individual object	Measurable concept	Part in composition	Variables to store class or object property values		Variable type	Value domain for categorical variable	Univariate statistical measure	Actual value in property of object	
					Single variable	Set of variables				Categorical value	Quantitative value
Biology/ICBN	Taxon	Specimen	[...]	[...]	[...]	[...]	n/a	n/a	n/a	n/a	n/a
DELTA	Item	Item	n/a	n/a	Character	Keyword ² , Include/Excl. characters	Character type	(Character) States	"Numeric values"?	State value; Attribute ³ = Char.+ set of state values	Numeric value; Attribute = Char.+ all num. values
DeltaAccess/ Divers.Descr.	Item	Item	n/a	n/a	Character	Char. group, char. subset	Character type	(Character) States	Statistical attributes	States	Value
Lucid3⁴	?	?	n/a	n/a	Character state	?	n/a?	States	n/a	?	?
Thiele, K.⁵	Entity	Entity	Feature ⁶	Feature	Feature	?	?	Value	?	Value	Value
SDD⁷	Taxon	Specimen, Taxon- Occurrence	Property (using Desc.- Concept)	Part (using Descriptive- Concept)	Character ⁸	Character tree	Character type	States	Univariate statistical Measure	Categorical state	Quantitative value
Diederich/ Fortuner⁹	Taxon	Organism	Property ⁶	Structure, substruct.	Character	?	Basic property	States (descriptive)	n/a?	Qualitative data? ¹⁰	Quantitative data, Value
Lebbe/ Vignes¹¹	Taxon, concept	Individual ("indivi- dus"), instance	Quality	Subject	Variable, Descriptor ("descripteur")	Descriptive system= vari- able set + de- pendencies	Type	Possible value, state ("modali- tés", "états")	Statistical summary	Qualitative attri- bute (= set of all values)	n/a
XPER², Vignes Lebbe /Chalubert¹²	Biol. entity	Case	Ontology: quality ¹³	Ontology: subject, structure	Descriptor or descriptor element	Descriptive system= vari- able set + de- pendencies	Character type ¹⁴	Single =qual. state or Bool., domain = "ref- erential set"	Numerical state, Integer/float state	Qualitative attri- bute (= set of all values)	n/a
Promethe- us descript. model¹⁵	Taxon	Specimen	Defined property term ⁶	Defined structure term	Conceptual description element ("DE")	Description unit (?)	n/a?	State group?	Generalized measure	Scored qualitative description element	Scored quanti- tative descrip- tion element
NEXUS	Taxon	n/a?	n/a	n/a	Character	Charset	"Format DataType" (?)	(Character) States	Item (multiple items in each char. × tax. cell)	"Data matrix"	"Continuous"

	Class ¹ of ob- jects	Individual object	Measurable concept	Part in composition	Variables to store class or object property values		Variable type	Value domain for categorical variable	Univariate statistical measure	Actual value in property of object	
					Single variable	Set of variables				Categorical value	Quantitative value
UML 1.x	Class	Object, Class instance	n/a	Class in composition, Component	Attribute ⁶	?	Type	«enumeration» Class	[...]	?	?
Entity- Relationship	Entity type	Entity	n/a	n/a	Attribute ⁶	[...]	Type (incl. user-def.)	value domain, lookup table	Aggregate functions	Value	Value
Java	Class	Object	n/a	Class, Object	Field; or set- /get-methods	[...]	Type	Enumeration (Enum)	[...]	Enum. object (Java 1.5)	Numeric type value
.NET/Mono	Class	Object	n/a	Class, Object	Property ⁶	[...]	Type	Enumeration	[...]	Enum. object	?

“[...]” indicates variable, unsettled usage, “?” unknown usage, and “n/a” (not applicable) indicates that a concept appears to be not supported.

¹ “Class” also has a special meaning as a taxonomic rank in biological nomenclature. Throughout this thesis, the term class is used in the general sense, not as a specific biological rank.

² Intkey “Keywords” (using the “Define Characters” directive) are the only facility in DELTA where a character may be a member of multiple character sets. Other DELTA facilities (Character headings, Item subheadings, and Link characters) partition characters uniquely into sets. Views can further be defined with Include/exclude characters.

³ DELTA uses the term “attribute” both for categorical and quantitative data (Dallwitz & al. 2000a) and defines it as “character number” (i. e., reference to a character definition) “together with the character values (state numbers or quantitative values) which apply to the taxon being described”. In addition to state numbers or quantitative values also special symbols ‘V’, ‘U’, ‘-’ are supported (compare “Coding status”, p. 74). Note that Sokal & Rohlf (1981) and Zar (1984) define the term “attribute” as exclusively referring to data on the nominal scale.

⁴ CBIT Lucid3: information by K. Thiele (pers. comm.)

⁵ Simplification of terms proposed by K. Thiele in SDD discussions (pers. comm.)

⁶ Dictionary definition of *attribute*, *feature*, and *property* may also be found in Table 44 (p. 164). Note that in XML the terms “attribute” and “entity” have completely different semantics than any of those listed above.

⁷ See p. 20 for references for SDD.

⁸ “Character” in terminology is used as the abstract concept (“character variable”, “character definition”). In the context of a description (i. e. an instance) the term “character data” is used where necessary to clarify the context (compare p. 31).

⁹ Diederich/Fortuner: according to Diederich (1997), Diederich & al. (1997), Diederich & al. (1998).

¹⁰ Diederich & al. (1997) define qualitative data as “descriptive (textual)” data, opposed to quantitative data define as “real numbers, integers” (but without statistics).

¹¹ Lebbe/Vignes: according to Lebbe (1991), Lebbe & Vignes (1998), and Diederich & al. (1997). The author of this thesis regrets that due to his lack of knowledge of French he could not do justice to Lebbe (1991), probably one of the most thorough treatments of descriptive data in recent years. Readers able to read French are encouraged to read this beyond the places where it could be cited in the current publication.

¹² According to Chalubert & Vignes Lebbe (2006) and pers. comm. R. Vignes Lebbe.

¹³ A ‘quality’ is what could be measured on something: a length, a color, a shape, etc. Then a descriptor is the combination of a quality on a designed part (one or several subjects): “color of the anterior wings”, “length of antennae”, “ratio of the length petals/sepals” etc.” (R. Vignes Lebbe, pers. comm.). Thus, the term here includes both categorical (qualitative) and quantitative measurements.

¹⁴ XPER and XPER² currently support only discrete values (qualitative, Boolean, or interval range).

¹⁵ “Prometheus Description model” (p. 21): according to Pullan & al. (2005).

4. Fundamental aspects of description models

4.1. Introduction

Describing objects seems to be a common and intuitive task. Modeling this activity in a precise way is, however, anything but trivial. The following sections try to introduce basic concepts and discuss the various problems and modeling approaches encountered in object descriptions.

Some of the problems involved in language usage and the recognition of parts and properties of objects are introduced by examples in the following section. Unconstrained *natural language descriptions* are an intuitive, but difficult-to-analyze form of descriptions that is briefly discussed on p. 39ff. The following sections of this chapter then try to lay groundwork for more structured descriptions by discussing levels of abstraction and the use of terminology (p. 42), data types (p. 49), mapping between data types (p. 66), coding status (including manual coding for inapplicability, p. 74), rule-based character dependency and applicability (p. 76), the problem of information aggregation associated with the description of both classes and individuals (p. 83), and inheriting data along the taxonomic hierarchy (p. 99). Finally, the complex topics of description storage models (character or state matrix, character decomposition etc. (p. 104), and descriptive ontologies (composition and generalization hierarchies for parts, properties, methods, etc., p. 131) are discussed.

4.2. Context, recognition, and language

An essential prerequisite to enable the comparison of diverse objects is a generalized conceptual model (or “framework”), within which differences may be compared in a meaningful way. Such a model is essentially a language for object descriptions, defining the concepts and names of object parts, methods, properties, and property values. Unfortunately, a language general enough to be applicable to all objects is often not expressive enough to distinguish similar objects, at least not in an efficient way. This is easily overlooked, since humans experience the context-specific vocabularies (i. e. “linguistic registers”) that are required to communicate about everyday objects as a part of the general language.

For example, many context-specific terms will be used in the description of a car model, defining parts and their arrangements, functional concepts, complex properties and property values: “steering wheel”, “automatic transmission”, “sunroof”, “metallic” paint, etc. Although most readers will understand these terms, it is easily imagined how, without knowledge of the “car vocabulary”, many terms could be misunderstood. Without an appropriate language, describing the composition of objects may become difficult, if not impossible. It is not conceivable to start in a corner of a car and describe the properties of every part encountered in a linear sequence.

The names of object parts (also called “components” or “structural elements”) often imply more than position in a composition. A term like “steering wheel” will imply a general functional and structural description. Often even default (or “most common”) properties like shape and color are implied. The description of the steering wheel of a specific car model will then use the general concept as well as omit the most general properties being already inferred by the consumer of the descriptions (e. g., “the brightly red steering wheel is shaped almost in the form a rectangle ...”, but not: “the steering wheel is circular and painted black”).

Similarly, property names and values may be context-specific. This may even apply to very general concepts like color, which may be overloaded with context-specific terminology like the proprietary marketing terms for color that car manufacturers use.

The problem of recognizing a part as belonging to a certain category (e. g., “rear mirror”, “plant leaf”, “car antenna,” or “insect antenna”) is difficult to address formally. The relationship

between the composition of parts (presence and multiplicity) and object properties is one of reciprocal dependencies and consequently difficult to model (Figs. 8-9). Creating part definitions from general properties alone (color, shape, etc.) is often very difficult. Many properties will be specific to the context of the described part (e. g., “venation”, the arrangement of vascular bundles of the leaf), or conversely imply the presence for further child parts (e. g., “hairiness”). Similarly, the relation to neighboring or parent parts will often be part of the definition of a part (e. g., “sepals” as opposed to “bracts”).

This can best be studied when considering out-of-context identifications of components. For instance, during a criminal investigation it may be necessary to identify car parts without knowing their context. A classification system of car parts that is truly general and may be applied by someone who has never previously studied the parts is severely constrained in the properties it can use. In general a much more successful method would be to have someone study all car parts in their context (placement within the car, function, manufacturer, and manufacturing period) and use the human associative memory. This memory could be supported by a classification system that is partly based on compositional context (e. g., “parts of light bulbs”) and supported by general properties (e. g., “metallic”). Clearly, knowing that a piece of glass is a part of the light bulb greatly simplifies the task of identifying the car model to which it may belong.

In biology, complete out-of-context identification of parts (e. g., air-borne spores or animal parts when studying predator diet) is rare. However, this is offset with a huge diversity of form and properties of life on earth. The missing context is often not the placement in a part composition, but the placement in a taxonomic group (virus, bacteria, annelids, fungi, nematodes, etc.). Independent terminologies have been developed for most taxonomic groups, sometimes introducing superfluously synonymous concepts (similar to car colors), but most often introducing terms and concepts for object parts and properties that are necessary, efficient and convenient. Again, returning to the car example, a similar situation may be found in the criminal investigation: recognition that a machine part is from a racing car, a truck, or a building machine may be vital to solve the case.

Some other problems involved in the reciprocal dependencies of recognition and analysis are:

- Individual variability: This problem is typically smaller for manufactured objects like cars than for biological object that may show high within-class variation (due to genetic polymorphisms and phenological plasticity).

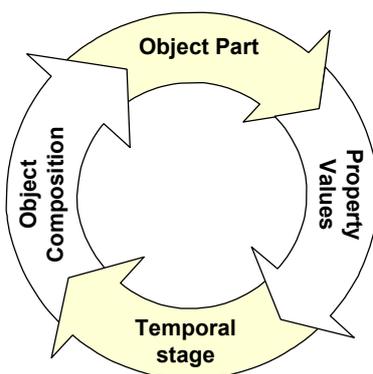


Figure 8. Recognition of object parts is a cyclic identification process based on recognition of object properties, life-cycle stage, and composition of previously recognized object parts.

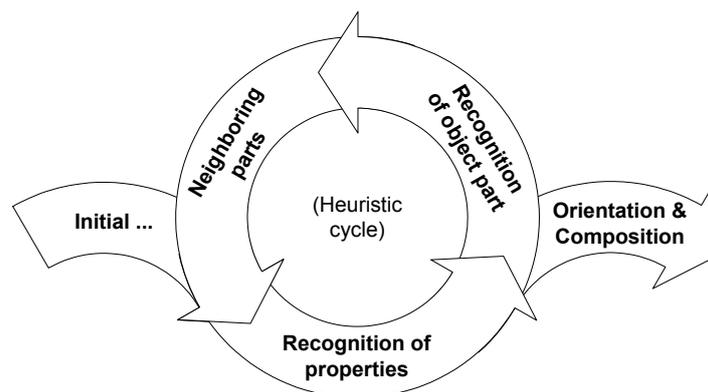


Figure 9. Interpretation of the recognition of object parts as a heuristic cycle. Each recognition step is a hypothesis tested through evaluation of properties and information about neighboring parts. Ultimately, the general orientation (front/back, top/bottom) and the object composition are resolved. Additional steps may be required to recognize life stage or taxonomic context.

- Aging and life-cycle dependency: A new brake may be as different from a worn-out one as a young plant from an old (“senescent”) one. Recognition of the life-cycle stage is an essential part of the heuristic process of recognizing parts and properties of objects (Fig. 8).
- Descriptions are strongly influenced by the methods used to observe properties. A property value without knowledge of the method used to obtain it may be of limited value (Fig. 10). The relation between properties, methods, and part-composition is a central problem when modeling descriptive data and will be discussed repeatedly in the following sections.
- The taxonomic diversity leads to increased diversity of components, properties, and methods. The greater the taxonomic diversity, the more difficult it becomes to establish a common framework within which comparisons remain meaningful (Fig. 10).

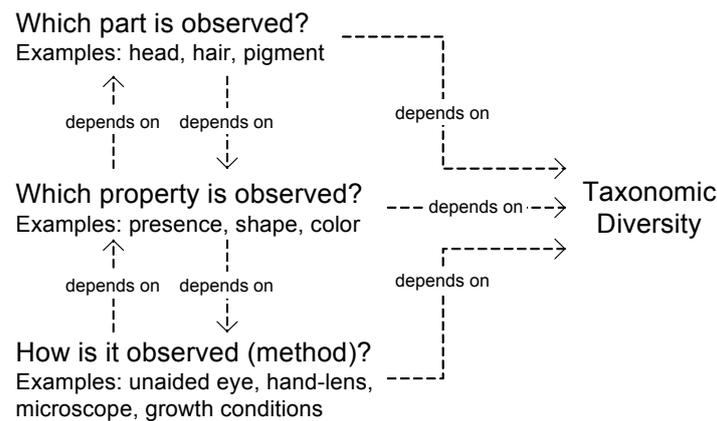


Figure 10. Additional dependencies on observation methodology and taxonomic diversity, extending Figs. 8-9. The diversity of methods, properties, and components (i. e. structural diversity) is increased with taxonomic diversity.

One solution to this problem is to choose analytical methods that are as context-independent as possible. In the car-part-example, if small machine parts have to be identified out-of-context, forensic practice may resort to expensive, but highly general and feature-rich properties like isotope composition to enable out-of-context identification. Similarly, biological identification uses more and more context-free identification methods based on molecular patterns or sequences (including “DNA bar-coding”). Although still comparatively expensive, these methods typically provide sufficient information to obtain an identification result in a single step, are usually applicable to a large taxonomic domain, and are usually independent of the compositional complexity of biological objects (the latter applying to DNA-based, but often not to RNA- or protein-based methods).

The examples above illustrate how much associative thinking biologists normally use when they, for example, generalize strongly different structures as a “leaf” – while recognizing other leaf-like structures as “stipules” or “cladodes”. Language can incorporate many of these intuitions, and descriptions addressing humans will be able to make use of these. However, the specialized language used by taxonomists is often specific for a given taxonomic group and depends on intensive training and experience. With an increasing diversity of objects, and with an increased diversity of expertise of the users, finding an appropriate common description language becomes more and more difficult.

No complete solution is known to the problems of descriptive data, i. e., combining the desirable properties of allowing associative thinking while retaining analytical properties of data sets and adequate manageability by biologists. Although it is probably possible to build a fully axiomatic descriptive terminology for biology, where each term is defined from first principles or based on other terms previously described, the author knows of no such attempt. Also, at least for the purpose of identification, the value of such an enterprise is questionable. Even many “nor-

mal” biological definitions that require anatomical, ultrastructural or other properties that are very difficult or expensive to observe, may, in the practice of biological identification, be replaced by terms appealing to intuition and associative memory.

The following sections discuss the advantages and trade-offs of various description models. The first section introduces the most traditional and perhaps intuitive method of describing biological objects by free-form text.

4.3. Natural language descriptions

The most intuitive way to describe an object is to use “normal” or “natural” language in an unconstrained free-form text element. This method relies on terms whose definitions are either specific to the subject (a “register” or “terminology” within the language) or are assumed to be commonly known. As usual in natural language, terms are highly context-sensitive and at least those terms borrowed from common language are often ambiguous. Natural language descriptions in biology normally use only a restricted code consisting mostly of enumerations. They are commonly used in most taxonomic monographs and may look like the following (hypothetical) example:

Stromata small, superficial, orbicular to elongate, at first gray, later black, 1-3 × 1-2 mm and 0.5 mm high; perithecia semiglobose, 0.5-0.6 mm diam. with papillate ostiola; asci cylindric, 80-90 × 7-11 μm; ascospores uniseriate, ellipsoidal, with obtuse ends, dark brown, often with large oil drop in center, 15-18 × 5-8 μm; paraphyses present.

The compact, relatively readable format provided by such descriptions is an efficient way to verify an identification reached primarily by other means (e. g., browsing illustrations or identification keys). Some identification methods may leave several taxa as potential, but doubtful, identifications and require comparison with illustrations or natural language descriptions in a final phase. However, even where the identification results in a single taxon name, an explicit confirmation phase (see p. 232) is highly desirable to catch errors that may have been made during the earlier parts of the identification procedure.

For this purpose most natural language descriptions focus on characteristics considered to be diagnostically relevant, i. e., necessary or useful to distinguish the individual taxa within the intended scope of a taxon set (e. g., all species of a family in Europe). Such descriptions are called “**diagnostic descriptions**” (or just “diagnosis”, which may, however, be confused with the “protologue”, i. e., the diagnostic description accompanying the first publication of a name). They omit information that is:

- constant within the set (and usually implied through context and the knowledge of the observers, e. g., that most plants are somewhat green),
- too variable within instances of the classes of the set to be useful,
- considered too difficult to observe, or
- considered redundant.

Strictly diagnostic descriptions contain only a *minimal* set of characters required to diagnose all members within a taxon scope (Pankhurst 1991).

Due to these omissions most natural language descriptions are only an incomplete documentation of the morphological, anatomical, ecological, etc., properties of a taxon. Descriptions that attempt to explicitly record any descriptive data quickly become inefficient when comparing them with actual objects. This trade-off between completeness and usability becomes highly relevant when descriptions are used out of the originally intended context (e. g., when new species have been discovered but no up-to-date monograph exists, or when monographs must be used in geographic region they were not intended for).

It is possible to find compromises and create natural language descriptions that are both concise and provide sufficient redundancy to offer a chance of raising doubt when encountering ob-

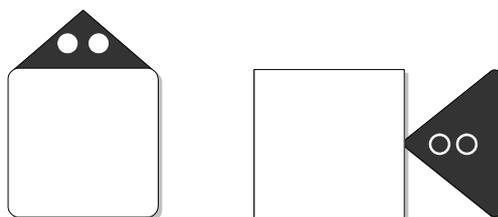
jects not covered by the diagnostic set. In digital formats it is possible to add markup information to enable hiding redundant or static parts of complete, documentary descriptions. However, the creation of concise diagnostic descriptions or diagnostic markup is not a trivial task. It requires excellent knowledge of a set of taxa plus experience with errors commonly made by users with less knowledge. In practice, such high quality natural language descriptions are thus achieved only for a tiny fraction of total biodiversity (e. g., birds, mammals, vascular plants from temperate regions, some well-studied insect groups like butterflies).

An advantage of natural language descriptions is that they often enable readers acquainted with the taxon group, to visualize the morphology from the description. Such visualization is aided by reliance on a well known “standard model”, from which only the deviations are recorded. In the experience of many biologists, it is much more difficult, to achieve a similar visualization based on an exhaustive, perhaps tabular documentation of all object properties. Due to this, natural language descriptions may be useful even as a temporary report format to facilitate proof-reading of data.

However, these visualizations largely depend on associative memory. Ambiguities, almost unavoidable when using natural language, cause direct visualizations without prior knowledge of the taxon to be often unsuccessful. Fig. 11 illustrates subtle ways in which both explicit omissions (rounded corners, absolute and relative orientation) and slight differences and ambiguities in terminology (white versus bright, circle as filled disc or outline, color as bright or white) may mean that a description may or may not fit strongly differing objects. Natural language descriptions are an effective tool supporting the associative memory of trained taxonomists, but less suited for identification or when used by less experienced users lacking these associations.

Also, it may be noted that future attempts at creating machine-aided “virtual reality” visualizations will be much more likely to succeed on the basis of exact and exhaustive descriptions, than on the basis of natural language descriptions.

Perhaps the most relevant shortcoming of natural language descriptions is that, both for humans and machines, descriptions are often very difficult to compare with each other (see, e. g., Allkin & Bisby 1988). Fig. 11 illustrates that this is not a question of limited ability of machine processing: whether the two example descriptions refer to the same object class or not remains ambiguous to humans as well. Performing such comparisons (and perhaps calculating similarity measures on them) is highly desirable, e. g., during a confirmation phase of an identification process (see p. 232), to search for potential taxonomic synonyms based on the similarity of descriptions, or to verify that newly described taxa are indeed new (compare “Failure of identification”, p. 310). Unfortunately, this also severely limits the use of computers to search for descriptions



Description 1:
Body rectangular, head symmetrical, narrowing evenly to a tip, with two bright eyespots

Description 2:
Body rectangular, head triangular with two white circles

Figure 11. Natural language descriptions are useful when comparing an existing object with a description, but often a poor method to visualize objects or compare descriptions with each other. The left description matches only the left object, the right description fits both objects.

matching the object in hand. Whereas direct comparisons between an object with each description are possible, this is often impractical if too many descriptions are in the scope. However, using a computer to search for a matching description (“leaves lanceolate”) essentially involves the creation of an ad-hoc description and has the same problems of comparability.

Similarly, the lack of comparability also severely limits the options for software-aided error checking and analysis of hand-generated natural language descriptions. In practice, even in taxonomic monographs covering only a few hundred species, proof-reading becomes an almost impossible task. Each description consists of dozens of individual statements that are extremely time-consuming to systematically verify against actual specimens. Taxonomists reading a description for consistency and cor-

respondence with memorized character states will, however, easily overlook erroneous statements. This is similar to the tendency to overlook spelling errors in a text as long as it remains comprehensible.

Finally, the lack of comparability prevents tailoring data for different purposes, such as documentation of taxonomic knowledge, for phylogenetic analysis, or for identification purposes (Allkin & Bisby 1988). For each purpose, data have to be created and maintained separately, and error corrections or new information have to be synchronized manually.

It is important to distinguish between natural language as original data storage, and natural language as a report format automatically generated from other storage forms. Several current applications (CSIRO Confor with “tonat” directive, Dallwitz & al. (2000a), and DiversityDescriptions, p. 322) are capable of producing such reports (Fig. 12). Doing so requires additional information. The approaches to do so differ substantially between DELTA (p. 19), DiversityDescriptions, and SDD. Further information regarding natural language generation in DELTA may be found in Pankhurst (1991) and Dallwitz (2000a, 2005c). For DiversityDescriptions see the user’s guide (Hagedorn 1999a) and p. 327 in the documentation of the information model.

With regard to original data storage, however, the problems mentioned make it desirable to develop more stringent models that facilitate machine processing and provide better knowledge management and error checking options. Such models are discussed in the following sections.

Erysiphe betae (Vanha) Weltzien – **Input Form**

Ascospores ▲

526. Ascospores number (categorical) 1. 1-2 per ascus
 2. c. 4 per ascus 3. c. 8 per ascus 4. 12-16 per ascus
 5. 16-32 per ascus 6. more than 32 per ascus
 (Select a char. state: to add this modifier:)
 (Select a char. state: to add this note/comment:)

528. Ascospores shape 1. globose 2. subglobose
 3. broadly ellipsoidal 4. ellipsoidal 5. oval
 6. fusiform-elongate 7. bifusiform 8. filiform
 9. acerose 10. cylindrical 11. oblong-obtuse
 12. oblong-truncate 13. discoid (in surface view)
 14. discoid (in side view) 15. lenticular (in surface view)
 16. lenticular (in side view) 17. sigmoid 18. reniform
 19. allantoid 20. lunate 21. falcate 22. ovoid
 23. obovoid 24. pyriform 25. obpyriform 26. clavate
 (Select a char. state: to add this modifier:)
 (Select a char. state: to add this note/comment:)

529. Ascospores length (numerical)

Min:	Lower range:	Mean:	Upper range:	Max:	Unit:
<input type="text" value="16"/>	<input type="text" value="18"/>	<input type="text" value="21.3"/>	<input type="text" value="26"/>	<input type="text"/>	<input type="text" value="µm long"/>

530. Ascospores width (numerical)

Min:	Lower range:	Mean:	Upper range:	Max:	Unit:
<input type="text"/>	<input type="text" value="11"/>	<input type="text" value="13.7"/>	<input type="text" value="16"/>	<input type="text"/>	<input type="text" value="µm wide"/>

Erysiphe betae (Vanha) Weltzien – **Description**

Data maintenance: Data compiled and standard item. Data record authors: Kainz C. (01-07-20). Record not revised (01-08-28).

Nomenclature: Taxonomic status: accepted name. Taxonomic level: species. Basionym: *Microsphaera betae* Vanha. Family: Erysiphaceae Tul. & C. Tul. Order: Erysiphales. References: Braun U., Beih. Nova Hedwigia 89: 1-700 [217-218] (1987).

Geography and Ecology: Global distribution: Africa, Asia-Temperate, Europe, Northern America, Southern America, and Asia-Tropical. Life habit: phytopathogenic.

Ascoma morphology: Ascomata present, cleistothecoid, gregarious or subgregarious, on thallus, orbicular, (0.075)-0.1-0.12-(0.135) mm in diam.; external filaments present, mycelioid, 3.5-9 µm in diam., hyaline, numerous, at lower half of ascomata, wall thin, not ramified, septate.

Asci: Asci unitunicate, 3-8, at basis rarely not or indistinctly stipitate, 45-75 µm long, 30-45-(50) µm wide. **Ascospores:** Ascospores c. 4 per ascus, ellipsoidal or slightly ovoid, (16)-18-21.3-26 µm long, 11-13.7-16 µm wide, aseptate; wall thin and smooth, hyaline, remaining hyaline.

Conidiomata: Present, hyphomycetous. **Conidiophores and conidia:** Conidiophores hyphae ramified. Hyphae foot cells 18-35-(40) µm long, 7-11 µm wide; conidia macroconidial, single, ellipsoidal, cylindrical, or ovoid, 30-50 µm long, (11)-14-22.5 µm wide, aseptate, not ramified; fibrosin bodies invisible.

Host taxa: Host plant / phorophyte family(-ies): Chenopodiaceae.

Figure 12. An example of an editing form (left) and the corresponding natural language description (right), both generated by DiversityDescriptions. Corresponding data are highlighted in red (from Hagedorn 2002a).

(Throughout this thesis a list of consecutively numbered statements expressing required or desirable features for a descriptive information model are presented in boxes like the following:)

1. The provision of free-form text natural language descriptions (full or diagnostic) is desirable, for example, during the identification process.

2. These may either be authored (including digitized legacy descriptions) or automatically generated from more structured descriptive data forms. A number or data items is required for natural language generation – these are not discussed further in this thesis, but are present in the models presented.
3. For many purposes, natural language descriptions are inadequate. Supplying more structured forms of descriptions is highly desirable.

4.4. Structured descriptions and the concept of terminology

Level of abstraction of descriptive information models

Information models express a static view of how information about a specific knowledge domain may be subdivided and structured. A model may limit users to a highly constrained vocabulary, it may allow users to extend the vocabulary, or it may provide for unconstrained (e. g., free-form text) information. After deciding that free-form natural language text alone is unsatisfactory, a deliberation is necessary, which level of structure and which level of semantic abstraction is appropriate for descriptive data.

Given the task to model plant descriptions, a naïve information designer might create data elements like plant height (type: double), number of petals (type: integer), leaf presence (type: Boolean), leaf shape (type: enumeration). In object-oriented terminology this describes an object through instances of a class with several properties to which values are assigned. Each property has a name and a type (text, categorical enumerations, or quantitative types).

Objects and classes are often shown as UML (Unified Modeling Language, see p. 24) diagrams. UML lists the attributes of a class in the diagrams below the name of the class or object. Fig. 13 provides an example both for an instance object (an actual object with values) and a class (a generalized concept for a set of objects). In both examples the methods (which may be shown below the attributes) are suppressed. Where appropriate, in some diagrams the attribute box itself may also be suppressed (leaving only the top part with the class name). For more information about UML class and association icons see Fig. 5 (p. 25) in the introduction.

UML as well as most ER (Entity-Relationship) models prefer the term “attribute” over “property” (compare Table 3, p. 34). In the present discussion, attribute and property are used interchangeably, with a preference for “property” in conceptual thinking, and “attribute” when explicitly referring to class attributes.

In biology, a property or attribute of an object is often called a “character” (compare p. 31 for

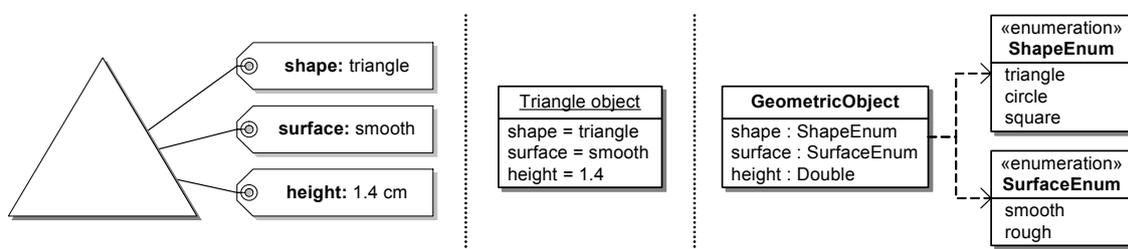


Figure 13. Comparison of informal presentation, UML object diagram and a corresponding class diagram. An object (the triangle) described through associated property-value pairs. On the left side the object properties are informally symbolized graphically by tags attached to the object. In the center, the same situation is depicted as a UML object diagram (name of object instance underlined, attributes with assigned values) and on the right side it is shown as a UML class diagram (together with the enumeration data types the shape and surface properties depend on).

distinction between “character variable” and “character data”). In the case of properties with a categorical data type the property value is commonly called a “character state”. Terms like character, state, attribute, and property are also used in various models, and usually differ strongly from the definitions used in OOP (object-oriented programming), UML, or ER notation (compare Table 3, p. 34). The term “property” is used here in a general sense as used in OOP or by Lebbe (1991) and Lebbe & Vignes (1998), not in the sense of the “basic properties” (Diederich 1997, Diederich & al. 1997, Diederich & al. 1998). Similarly, the term “attribute” is used in the general UML sense, not as used in DELTA (p. 19; compare Table 3).

A design like the one shown in Fig. 13 remains close to the terminology used in the knowledge domain. For the sake of simplicity and intuitiveness, this is recommended if the primary purpose of the model is to partition information into free-form text data elements. Example for schemata using this direct approach are OpenKey (2003) and the one cited in Cui & Heidorn (2007). However, as will be shown further below, substantially more structure and ontological information is required for machine-based reasoning (e. g., by identification programs). Also, in most cases the data are more complex than initially assumed by simple systems. Plant height is often not recorded as a single value, but a range or a set of statistical measures (such as mean, standard deviation, and sample size). The leaves of a single plant may vary strongly and require leaf shape to be expressed as a set of categories rather than a single value (which requires a collection class in OOP and a separate entity in ER models), and shape may be accompanied with frequency information (e. g., “usually ovate, rarely linear”). These examples show that when trying to reach a model of better quality, the “simple” models may soon become highly complex, given that a typical biological object may be described through several hundred properties. Substantial resources are required to develop software handling descriptions of such complexity.

Biology deals with an enormous diversity of organisms: viroids, viruses, prokaryotes, phylogenetically diverse groups classified as protists, fungi, and algae, plants, and animals. It is obvious that with the exception of a few molecular characters, no property is applicable to all organisms. Software solutions directly reflecting the property-approach outlined in Fig. 13 result in taxon-group-specific models. Although financial resources may be found to develop taxon-group-specific applications for popular groups like birds, butterflies, flowering plants, or fungi (see, e. g., Lauber & Wagner 2001, Seybold & al. 2001, Götz 2003, Böhmer & Wohanka 2002), this has not been and is unlikely to become the case for the smaller or less popular groups of organisms. Unsurprisingly, however, knowledge about biodiversity is particularly lacking of in exactly these groups (see Fig. 1, p. 14).

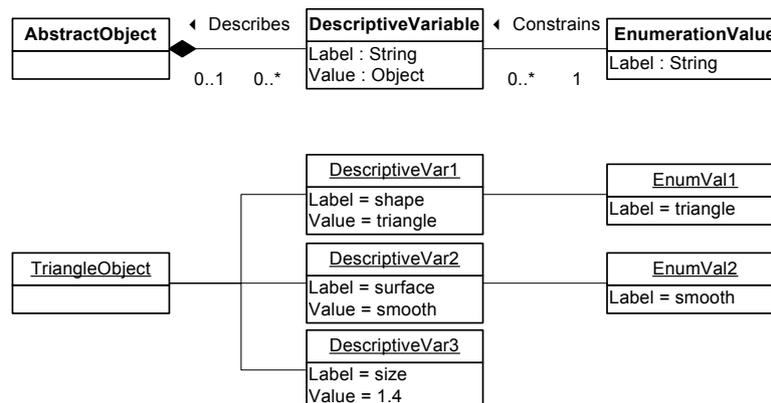


Figure 14. An example of an abstract descriptive data information model corresponding to the specific model shown in Fig. 13. The specific variables and enumerations have been dissolved and replaced with a collection of label/value pairs. The upper UML class diagram shows the general model, the lower UML object diagram an example instance.

Most existing models for structured descriptive data therefore attempt to avoid a close correspondence between the model and the described object and use more basic “concepts” of descriptions. As a consequence, the level of abstraction of most descriptive information models is considerably higher than in many other information models. Instead of specific properties like “plant height”, general quantitative data and univariate statistics are handled, instead of “leaf shape”, categorical data on object-parts are recorded (see section “Data types”, p. 49 for further information). Fig. 14 shows a very simple way in which the specific model from Fig. 13 might be converted into an abstract model.

4. Although simple, taxon group and observation methodology-specific information models for structured descriptive information do have a place, a more abstract and generalized information model is desirable.

Generalization and terminology

Abstract models provide only a framework to record the descriptions. Additional information (a “specification”) is required to adapt the general model to a specific organism group. In the SDD model this information is called “terminology”, in DELTA “character list”. In the simplest case the terminology is not explicitly defined and consists simply of label strings (e. g., “Descriptive-Variable.Label” in Fig. 14). However, this means that the advantage over the natural language lies primarily in a predictable segmentation of information. Whether descriptive variables (e. g., labeled “shape” and “outline”) are comparable or not is not directly accessible to machine processing. Similarly, the relation between “growth rate” and “growth diameter” is difficult to assess for humans as well unless further definitions are available. For this reason the terminology is usually explicitly defined and some part of the full descriptive information model is responsible for defining terms. For example, for each character the DELTA model supports a concise label, a natural language wording, an image, and extensive definitional text providing additional semantics (“character notes”).

With the introduction of explicit terminology, the information model to describe a specific taxonomic group is split into a fundamental information model, and a domain-specific extension. The fundamental information model is collaboratively designed by information scientists and biologists and subsequently implemented. However, defining the terminology is not a trivial task either, and normally has to be carried out by biological domain experts (taxonomists, plant pathologists, ecologists, behavioral scientists, etc.), who usually have little experience in information modeling.

Unfortunately, conventional biological terminology (as used in natural language descriptions) is often contradictory and poorly defined. Definitions of terms are often learned through years of experience, rather than by reading dictionaries or glossaries. As a consequence, biological terminology has a tendency to be fragmented by organism group, country, and “school of thought”. Many such differences between definitions are relatively minor and the consequences may not be obvious until a specific organism with an unusual combination or expression of characters is encountered. It is therefore difficult to “prove” that a terminological definition is satisfactory. Any terminological concept should hence be defined through text or media resources enabling human users to compare their own concepts with a definition (Pullan & al. 2005). An important aspect of this is whether the definitions are sufficient to enable identifications with little training or using computer-aided self-training. This aspect often has to be addressed separately from the formal and complete definitions of terms.

Terminology also addresses software agents. Descriptive information models differ in how much of the definition of terminology is a formal ontology intelligible to software agents, and how much is exclusively directed towards a human user. The older DELTA format addressed primarily humans, projects like Prometheus (p. 21) or “plantontology.org” (e. g., Pujar & al.

2006, Ilic & al. 2006) concentrate on machine-readable definitions, and SDD (p. 20) attempts to find a compromise by adding rich machine-readable semantics, that can still be easily defined by the domain expert creating a specific terminology.

5. A descriptive terminology is required as mediator between the abstract information model and the concrete properties and observation methods in a taxonomic group.

Static versus dynamic terminology models

As more objects are described, property concepts often have to be extended or reformulated. Fig. 15 continues the example started in Fig. 13 (p. 42) by adding new objects. Some objects may be described by extending the list of previously used shape concepts (circle, square with rounded corners, hexagon, square with sharp corners, cross, rectangle, ellipse), whereas others call for additional differentiating properties: triangle with unequal or equal sides, right-angled triangle.

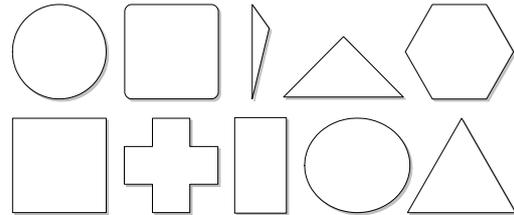


Figure 15. A collection of objects that would require a refinement of the descriptive properties used in Fig. 13.

Such a “schema evolution” (or “character evolution”) is a major problem in the design of all systems managing descriptive data. It is probably unavoidable if the objects that shall be described are not fully known in advance. Biology shares this problem with all scientific endeavors dealing with yet unexplored terrain.

Table 4. Some consequences of changing the terminology in an extended character/state model (as used, e. g., by DiversityDescriptions).

Terminology	Operation	Consequences on descriptions
Character (incl. states)	Add	Neutral (frequent operation)
	Remove	Normally only immediate effects occur: Data using this char. must be deleted, but other data are not affected. Exceptions are controlling characters in a character dependency/applicability relation (p. 76) and characters involved in mapping and calculations.
	Change definition	The consequences are difficult to judge. Note that even clarifying previously ambiguous definitions implies that some existing data may be erroneous as a result of previous misunderstandings.
	Change arrangement (tree or sequence)	Normally neutral (although it may lead to or resolve misunderstandings)
Char. state (discrete)	Add/Remove/Change	See character. Example: “antennae segments with 3 states: 1, 2-3, 4-5”. Adding “6-7” is a neutral operation and does not affect existing data
	Change arrangement	Considerable interpretation change for categorical data on ordinal scale, neutral if scale is nominal.
	Move to different character	This may be neutral: Independent states that have been recorded in ill-defined characters combining several properties may be moved to more appropriate character, assuming the observation methods used and the parts to which the characters apply are appropriate. If the state is a controlling state in a character dependency/applicability relation, only the dependency relation needs to be updated, but no character data in descriptions.
Char. state (classifying continuous variation)	Add/Remove	In addition to the effects described above, adding or removing states that partition a continuously varying feature will normally influence the borders of existing states. The data referring to these states will have to be revised. It may be necessary to maintain the changed states (marked as deprecated), introduce new states for all changed concepts, and migrate all existing data until the deprecated states are no longer used and may be deleted.
Univariate statistical measure	Add/Remove	Making a new measure (e. g., standard error) available is neutral; removal requires deletion of data but has only immediate effects. A change of the definition is not possible; the semantics are fully defined and refer to standard statistical practices.

Changing the terminology affects descriptive data based on the terminology and terminological changes must be carefully considered (Table 4). However, the problem is somewhat smaller in practice than in theory, because in practice it is unlikely that terms are applied precisely according to their definition. Term definitions may be misunderstood or even ignored: Normally, recording scientists will assume that previously learned concepts are also applicable to the current data set with defined terminology. Explicit definitions will be considered only when problems or inconsistencies are noted (and most likely displaying all definitions all the time will be perceived as “information noise” and ignored...).

Thus, the application of terminology may differ substantially from the one intended in the definitions. These errors will usually be indistinguishable from the errors made because the delimitation of a term has slightly changed. A measure whether a change of terminology may be considered acceptable or not may be achieved by comparing:

- an estimate of the number of errors introduced in the data set by the terminology change,
- an estimate of the number of errors present in the data due to imprecise application of the terminology, and
- an estimate of the number of effective coding errors and omissions present in the data set due to incomplete proofing.

In general, experience with systems allowing the ad-hoc definition of new characters and states (and thus an evolutionary development of terminology) is positive. Ad-hoc definition may even be considered an indispensable feature (pers. comm. of users of DiversityDescriptions, which strongly supports reorganization of terminology). Similarly the Genisys model (Diederich 1997) outlines some explicit mechanisms to support schema evolution while minimizing the impact on existing data.

Designing a descriptive terminology is not much different from software development. In software engineering, the attempt to first design a near-perfect information and object model is sometimes called “Big design up front (BDUF)” (see, e. g., Ambler 2002, Ambler 2003b). This classical model of software engineering may still be preferable in certain situations. However, development processes that explicitly contain iterative elements (e. g., the “Rational Unified Process” of inception, elaboration, iterative cycles of design, implementation, testing, and refactoring during construction, and final transition to finished product, Jacobson & al. 1999) are often more successful. The rate of change is even greater in development processes like “extreme programming (XP)” (see, e. g., Beck 2000).

Unlike in software development, the usefulness of explicit prototyping phases is often limited when developing a terminology for biological descriptions. Unfortunately, a small sample of existing published descriptions is not an adequate representation of the problem domain. The contradictory nature of conventional terminology often surfaces only during the development of a structured terminology. Furthermore, the diversity of organisms is often so large that it is very difficult to assess the validity of a terminological model until a substantial proportion of the organisms have been studied. However, studying perhaps up to 50% of taxa in a taxonomic group in a first prototype phase and re-recording all data in a second pass is clearly inefficient.

Prototyping being limited should not be understood as it being useless. Starting with a badly designed terminology, and then having technical personnel record a large amount of data is almost guaranteed to either fail or produce low-quality data. Before attempting a large project, developing the terminology in smaller prototype projects is highly advisable. However, in the prototype projects as well as in the large project, continuous further change may have to be anticipated. Similar to software engineering, the best development process for descriptive terminology depends on the circumstances:

- Some large projects may fare best with extensive prototyping and then settling on a fixed terminology for a prolonged period of time. This will especially be the case if a large amount of resources are available and a large number of personnel have to be trained.

- Most large collaborative projects will, however, fare best with a development process similar to the “Rational Unified Process”, where change is explicitly expected, but occurs in a controlled way.
- Small projects may appreciate an “extreme describing” approach (analogous to “extreme programming”) and a software allowing rapid evolutionary changes of terminology even when data already have been entered. The advantages of iterative or evolutionary development processes can best be reaped if the designers of the terminology themselves record descriptive data and use the experience to improve the terminology.

Many terminological changes are nearly neutral in their effect on other data items (Table 4). Where adverse effects exist, the most relevant ones are likely to be detected relatively early. Changing a bad terminology, even if reducing the accuracy of a few descriptions, is usually preferable to sticking to a bad terminology for the remaining majority of data still to be recorded. Obviously, changes detected at a later time affect increasingly rarely used characters, where manual revision of all affected data typically becomes feasible.

6. A dynamically evolving terminology that may be changed while recording descriptive data is possible and desirable.

Reaching terminological stability

Although researchers need to be able to define their own “proprietary” terminology, it is evident that standardization is very desirable. This has a technical aspect and a semantic aspect:

Technical standardization: To compare and integrate descriptive data sets, the identity of terms from different terminologies has to be assessed. This is not possible based on labels or names: For example, two data sets defining a character with the same name “leaf structure” may have completely incongruent definitions (e. g., rough or smooth surface vs. anatomical features). Ontology languages such as OWL (McGuinness & van Harmelen 2004) enable the expression of semantic information in a machine-readable way, but the expressiveness of current ontological techniques remains limited. Such information can greatly help finding similar terms, but it is unlikely that a question whether the similarity is large enough to warrant data integration can be answered based on such information alone. Terminological concepts thus should carry unambiguous, preferably globally unique, identifiers. Different terminologies can then make assertions about comparability of their concept with a standard concept. See the section “Federation and modularization of terminology” on p. 180 for an in-depth discussion of this.

Semantic standardization: Improving the consistency of terminology and reaching convergence of usage is an important goal for biodiversity research, which in principle is independent of the use of computer programs for descriptions. However, attempting to define a structured terminology to be used by software programs, aids in finding problems that may otherwise be overlooked.

Although many groups attempt to construct terminologies (or “ontologies”), these efforts still remain isolated. Older attempts, e. g., by the Taxonomic Databases Working Group (TDWG) to standardize on plant descriptors have remained unsuccessful (R. Pankhurst, pers. comm.), perhaps in part because the need to standardize was not perceived widely enough yet. Nowadays computer science and world-wide data exchange create technical requirements for standardization to which science has to react. Pullan & al. (2005) believe that their model is able to reach a consensus within large, “natural groups” like angiosperms. Acceptance of the terminology still has to be proven. However, their effort is focused on a specific audience: taxonomists working on groups where most characters are morphological and easily observable (i. e., description does not depend critically on methods). Conversely, as discussed later on p. 159, the Plant Structure Ontology

(PSO, Ilic & al. 2006) has deliberately ignored the need of morphological descriptions in favor of simplifying comparisons of molecular gene-annotations.

Different opinions exist as to whether the current phase of terminological instability is transient and will soon be replaced by a period of terminological stability – or not. It is arguable that the assumption that a group of eminent scientists should form a standards consortium to decide on a fixed terminology is contradictory to the fact that science:

- is constantly discovering new features or characteristics,
- develops new methods,
- that the results of these new methods also modify the interpretation of features studied previously by older methods (e. g., SEM studies influence how morphology is interpreted in the light microscope),
- reinterprets features because of improved causal and functional understanding,
- but also is an opinionated, often very personal enterprise, where schools of thought arise that identify themselves through use of proprietary terminology and concepts, and which are replaced by an evolutionary process of survival of the fittest ideas rather than through logical argument (Kuhn 1970, Hull 1988, Hull & Ruse 1998).

The latter point is especially relevant. Terminological instability is most likely a part of the process of science, i. e., changes in terminology are a method to achieve progress.

7. Standardizing terminology is desirable and the information model should support this.
8. A stable and global identifier method for objects of the descriptive terminology is required.
9. Rigid standardization is no alternative to providing support for a freely evolving terminological schema evolution.

Relation between terminology and software implementations

Several approaches exist to implement software based on a combination of a general and abstract base model and domain-specific terminology:

- **Run-time concept:** The implementation is based directly on the abstract information model. The terminology is defined as data (expressed in an abstract model itself) and is interpreted at run-time. The user interface is based only on the abstract model but may provide run-time customization based on definitions from terminology.
- **Compile-time concept:** The software in which the terminology is defined is an application generator that creates terminology-specific implementations based on the modeling definitions. An example of this may be the Prometheus data entry tools said to be automatically generated specifically for data recording projects (Paterson & al. 2004). This is similar to a model-driven architecture where software applications are generated with high-end UML tools. The persistent storage of data will be specific to the terminology as it was defined when the application was generated.
- **Hybrid concept:** The user interface part of the application is based on a fixed terminology (compile-time concept), but all persistent storage remains constrained by the general abstract information model. Already the DELTA data compilation into a binary Intkey file may follow this model. JavaScript-based web identification tools that embed data in code (e. g., SLIKS, SAIKS, p. 20) will certainly benefit from such a model. These tools are optimized for user experience but suffer data management problems unless the key data can in the future be automatically generated from more flexible persistent storage.

The implementation concepts differ in their potential quality of user interface and the ease with which the terminology can be revised (Table 5). Most existing DELTA and NEXUS-based applications (e. g., CSIRO DELTA editor and Intkey, Pankey, DiversityDescriptions and CBIT Lucid/Lucid3) use the run-time concept. The generation of model-driven software application code is a relatively new development in computer science, and only limited experience (e. g., the

Castor framework is used for generating schema-driven parts of software in the EFG project, R. Morris, pers. comm.) exists in applying it to generating software for biological descriptions. It offers more flexibility in customizing the user interface. However, if the domain-specific terminology-part of the information model has to be changed frequently, the run-time concept may still be preferable.

The development effort required to implement the various concepts is difficult to assess. Whether it is simpler to create a fairly good user-interaction based on a highly abstract model, or to write a code-generator that generates specific user-interface code, depends largely on the available tools. A new generation of development tools may greatly simplify model-driven application development. One advantage of the run-time concept is certainly that even with standard database forms (as in DiversityDescriptions) an acceptable level of user interface quality may be achieved.

Table 5. Different implementation concepts for software based on an abstract model plus domain-specific terminology.

	Potential for quality of user interface	Revising and improving terminology:	
		— Ease —	— Involved agent —
Run-time-concepts	+	+	user
Compile-time-concepts	++	--	developer
Hybrid concepts	++	-	developer

10. Different concepts exist to implement software based on a combination of a generalized, abstract base model plus domain-specific terminology. The choice depends on available development tools and is *not* part of the model requirements.

4.5. Data types

Any attempt to generalize the handling of descriptive data creates a system of data types. Since most of the semantics of the data have been delegated to terminological data, these data types are primarily concerned with providing information for query and analysis purposes. The following sections discuss the most frequently used data type categorizations.

Measurement scales

Variables in statistical data analysis are usually classified (e. g., Zar 1984) as follows:

- **Nominal scale:** Data that may be classified into discrete categories but cannot be arranged into an order of relatedness. In biology this usually corresponds to the existence of polymorphic alleles in a feature determined by one to few genes. Examples are male/female, eye color states.
- **Ordinal scale:** Data that may be classified into discrete categories and further the values may be arranged in a ranked order, i. e., expressions like “A is greater than B” and “B is greater than C” are meaningful. Transitivity applies (i. e., $A > B$ and $B > C \rightarrow A > C$), but it is not possible to quantify the distance between categories and compare the intervals between categories.
- **Interval scale:** Data values are ordered and the interval between values may be quantified and compared. However, the position of zero is arbitrarily defined, so that the comparison of ratios of these intervals is not meaningful. The examples in biology fall into two subcategories:
 - **Linear interval scale:** The outstanding example are the common temperature scales °Celsius or °Fahrenheit. Since zero is an arbitrary point it is not meaningful to say that 20 °C is twice as hot as 10 °C (but both temperature scales are convertible to Kelvin (K), where zero is a logical point, and which is on the ratio scale, see below).

- **Cyclical interval scale:** Data that are based on time of day, season within year, compass directions, etc. are not uncommon in biology and require special statistical methods for analysis.
- **Ratio scale:** Data fulfill all properties of the interval scale, but further ratios of two values are meaningful to compare. This includes integer counts and the majority of continuous measurements used in biology.

These scales have a natural order in that each scale makes additional assumptions about the data. As a consequence, data on a higher scale can always be analysed using methods devised for a lower scale. Doing so will reduce the analytical power (error of not rejecting a false null-hypothesis) but if a result is significant on a lower scale it should always be significant on a higher scale as well. “Downgrading” of data is frequently used in statistical analysis. For example, even if the data are on the ratio scale, an analysis of variance (ANOVA) requires normal distribution and homogeneity of variances. If these assumptions are violated, it is possible to analyse the data using “non-parametric” methods, essentially reducing the measurement scale from ratio to ordinal.

It may be noted that Boolean values (i. e., categories with only two possible states, such as “true/false” or “absent/present”) can usually be equally well treated on the nominal and ordinal scale. Whether it makes sense to say “absent is less than present” or not may be a philosophical question, but the properties of data on the ordinal scale (transitivity, rank) that are used in ordinal statistics become applicable only for > 2 states.

Note that although in the interval scale the ratio of two values has no meaning, the multiplication and division operators are defined and used, e. g., to calculate averages. Thus an information model cannot simply prevent the use of all ratio operations on interval-scaled data.

In contrast to Zar (1984), Sokal & Rohlf (1981) do not discuss the distinction between the interval and ratio scale (presumably because the only relevant example of a linear interval scale in biology may be solved by conversion to degree K) and prefer the term “attributes” for nominal variables, and “ranked variables” for variables on the ordinal scale.

The use of the term “ordinal scale” may be somewhat problematic, because in statistical analysis ordinal data are usually assumed to be linearly ordered. However, categorical data in biology quite frequently are ordered in more complex ways (e. g., into a tree of plesiomorphous/ancestral and apomorphous/derived states in phylogenetic analysis, Fig. 16).

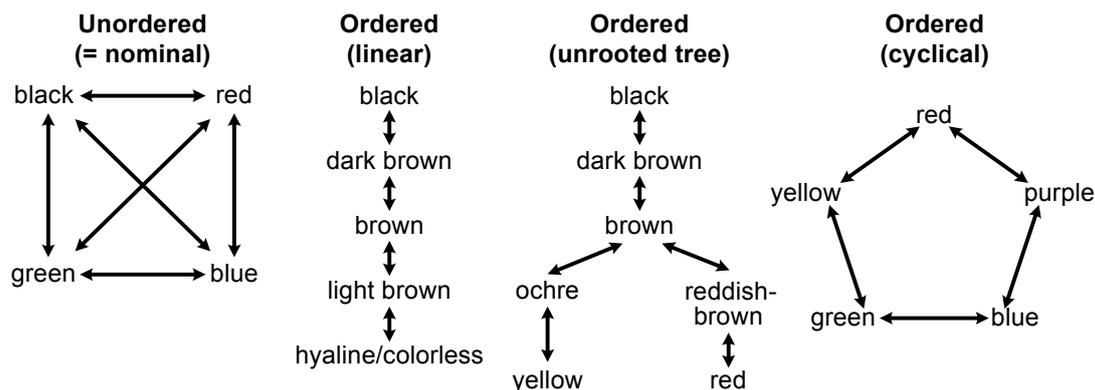


Figure 16. Illustration of categorical measurement scales. In the examples, color states are modeled using different assumptions, resulting in data on different scales. The arrows indicate possible transitions: In an unordered categorical character each state may be directly transformed into any other state, whereas this may require several steps where states are ordered.

11. The concept of measurement scales is an important concept in statistical and phylogenetic data analysis and should be reflected in the descriptive data type system.

Continuous versus discrete variables

Another central distinction in statistical analysis is that between continuous and discrete (also called “discontinuous” or “meristic”) variables (e. g., Zar 1984, Sokal & Rohlf 1981). Different perspectives exist when using the terms continuous and discrete and these are often difficult to separate. Here the following perspectives are proposed to clarify the discussion:

- The **statistical analysis perspective** is concerned with assumptions that statistical methods make about continuous variation. Under this perspective, it is relevant whether the actual data values are from a finite set of discrete values or whether they are from an infinite set of values. In principle, categorical (nominal or ordinal) and counting values are always discrete and values expressed through real numbers continuous. In practice, however, this can often not be guessed from the data directly. The numbers “26, 28, 30” may either represent the number of leaves, or the height of plants measured to the nearest centimeter. Even more confusingly, values like “1.7, 5.1” may be discrete. For example, they may represent the “bacteria per liter”, and are calculated based on counts of 1 or 3 colonies per agar plate, multiplied with a scaling factor. Furthermore, since all numeric computation is based on fixed-precision floating-point numbers from a finite set of values, it is obvious that a practical rather than a philosophical distinction between continuous and discrete is called for. The deviation of calculated and true error values (p) depends on the robustness of the testing method in respect to violations of its assumptions. For ANOVA, if discrete data contain sufficiently many values (20-100) and a testing for normal distribution is successful, the ANOVA will usually result in p -values of acceptable accuracy. On the other hand, an ANOVA analyzing values that appear to be discrete (but are not, like “1.7, 3.4, 5.1”) may falsely report a significant deviation from the null-hypothesis.
- The **functional (or causal) perspective** is concerned with processes in the object that is being studied. Variables may be known to be potentially continuous, but are treated as (usually ordinal) categorical variables due to a lack of efficient measurement methods. For example, animal behavior may be scored as “very aggressive”, “aggressive”, “neutral”, “submissive”, and “very submissive” (Sokal & Rohlf 1981). In biology, discrete characters are controlled by one or relatively few genes. An example of a discrete character is “determinate versus indeterminate habit of growth” in plants. Continuous (= “quantitative”) characters (as far as they are hereditary) are controlled by many genes. Each gene has a small effect that interacts (usually more or less linearly) with the effects of other genes. The distinction between continuous and discontinuous characters is therefore not absolute. In practice a character will be considered continuous if phenotypic variability and measurement error will be in the same order of magnitude as the differences between genotypes.
- Finally, an **operational categorization perspective** exists, if some property values are theoretically continuous, but this continuum is in fact not observed. Thus, in a given taxonomic group, certain ranges of property values may never occur, creating gaps that are highly useful when searching for borders of categorical classes (Thiele 1993). This is further discussed in “Singularity, extension and connectedness of categories”, p. 53.

Table 6. Interaction between continuous/discrete variation and measurement scales.

Measurement scale	Statistical analysis perspective	Functional/causal or operational perspective
Nominal	discrete	continuous or discrete
Ordinal	usually discrete	continuous or discrete
Interval	continuous or discrete	continuous or discrete
Ratio	continuous or discrete	continuous or discrete

Table 6 shows that, depending on the perspective, measurement scales and the classification into continuous versus discrete variables are not completely dependent. The discreteness under different perspectives is thus a separate part of the definition of a descriptive variable.

12. The distinction between continuous and discrete is relevant, but different perspectives exist when using it for descriptive data. Depending on the perspective, the distinction does or does not interact with the concept of measurement scales. It is, however, never nested within measurement scales.

Categorical versus quantitative (measurement) data

A frequently used simplification of the system of measurement scales plus discreteness is a fundamental distinction between *categorical* (including nominal and ordinal) and *quantitative* variables (including interval, ratio, both continuous as well as discrete). This dichotomy is useful because it reflects the differences in the data recording structure of categorical and quantitative variables. Whereas both types of data require general definition, including the measurement methodology (which part, which property, which instruments to use, etc.), only categorical data require definitions of the category values (“states”) that are used. Essentially, the values of a categorical variable are “pointers” to definitions present elsewhere. In contrast, the values of quantitative variables are standard numerical values. However, to capture measurements of *m*, *mm*, and μm in a single data set, it may be desirable to allow measurement unit scaling factors on individual measurements rather than define them in the general variable definition; both being unique to quantitative data.

For many analytical purposes, the information about measurement scale or discreteness will also be very desirable. However, this information may be given nested in each data type, either as separate subtypes or simply as metadata of characters.

Categorical variables may also be called “*qualitative data*” (e. g., Thiele 1993, the Genisys model, Diederich & al. 1998, or the Prometheus description model, cited on p. 21). For various reasons this may be less desirable:

- Some statistical texts restrict “qualitative” to categories on the nominal scale (e. g., Sokal & Rohlf (1981), while avoiding “qualitative” in definitions, use the term in explanations for unordered, unranked attributes).
- As noted by Thiele (1993), “qualitative” is often erroneously confounded with discrete and quantitative with continuous. In plant breeding “qualitative” seems to be used as a synonym for discontinuous characters and traits (e. g., EB 2001: “Plant breeding/Evaluation of plants”).
- Whether the term “qualitative” is fundamentally appropriate may depend on the (philosophical) definition of what a *quality* is, and whether or how it can be measured. According to CED (1992), the major senses of “quality” are “1. a distinguishing characteristic, property, or attribute. 2. the basic character or nature of something. 3. a trait or feature of personality. 4. degree or standard of excellence, esp. a high standard. [13th century: from Old French *qualité*, from Latin *qualitas* state, nature, from *qualis* of what sort]”. All senses given include quantitatively measurable traits (e. g., height, aggressiveness of a person, durability, or consumer satisfaction with a product).

Thus, although *quantitative/qualitative* form a more intuitive pair than *categorical/quantitative*, the term “categorical” is preferred in this treatment.

Similarly, *quantitative* variables may also be called “*measurement variables*”. Again, this may be slightly confusing since – depending on the measurement instrument – the result of a measurement may be categorical (e. g., an instrument measuring some toxic gas concentrations may report only “OK” or “Danger”).

It may further be noted that the term “*numerical data*” is no synonym for quantitative data and should be avoided. Both nominal and categorical data may be represented through numbers.

13. The distinction between categorical and quantitative data is a fundamental that should be reflected in the descriptive data type system.

Singularity, extension and connectedness of categories

The usual concept of “categorical data” informs about data representation (“statistical analysis perspective”, see “Continuous versus discrete variables”, p. 51), but does not necessarily inform about the world that is being represented by the data (see “functional (or causal) perspective”, p. 51). By definition, categorical data are always discrete in their data representation, but the underlying real-world concepts may be continuous. In this case categorical data are the result of a categorization (or “digitization”) process.

Knowledge whether categories are naturally distinct or a categorized continuous variable is important when analyzing the possible statements involving categorical data. For example, if color (continuous) is represented through color names (distinct categories), the natural language statements: “color intermediate between yellow and orange” or “color somewhere between yellow and orange” will normally be meaningful to humans (or machines that are informed about the underlying categorization process). Without this information, such a statement would appear to the computer just as absurd, as “the mother had 2.5 children” is to humans.

With respect to ranges, ordered categories provided additional need for interpretation. Many readers will interpret both “leaf 3 to 5 *something*” and “leaf between 3 and 5 *something*” neither as “3 or 5” nor as “exactly 4” (i. e., excluding the borders), but as “3, 4, or 5”. Clearly, this is guided by *something*: e. g., leaflets (discrete) or cm (continuous). Experienced botanists, knowing that in the genus under consideration leaflets occur only in odd number (e. g., 3, 5, 7, 9) may even include this knowledge into their interpretation of a range statement!

Thus, if non-adjacent categories are connected with “to” or “somewhere between”, the interpretation of an inclusive range may be dominant, and if adjacent categories are expressed the same way, depending on background knowledge about the character and measurement unit, interpretation of a single intermediate value may be dominant (without excluding an interpretation as an inclusive range).

Clearly, a substantial amount of knowledge about the real world is implicitly used when interpreting these statements. Furthermore, the existence of the two related natural language statements “*a* to *b*” and “between *a* and *b*” seems to be relevant. Although neither is unambiguously defined in natural language, the first is probably considered simpler than the second. Use of the second is more readily interpreted to exclude the border values of the range than the first. The information model should provide means to unambiguously express the intent that may be hidden in natural language expressions.

The following is an attempt to reconsider some aspects of categorical descriptive data and develop a new terminology for it. The concepts that are being categorized (i. e. classified) may be:

- *singular* (each is unambiguously a single thing) or *extended* (covering a range of variation);
- *connected* or *disjointed*;
- *adjacently connected* or not (in Fig. 17 d, ‘A’ and ‘C’ are connected through ‘B’, but not adjacent, in Fig. 17 e ‘D’ is adjacent and connected to all other categories);
- *ordered* in a natural order, which may be linear, form a tree, or an undirected or directed graph (cyclic or acyclic).

Each category (or class) may contain:

- one or several *singular* concepts (Fig. 17 a, categories ‘A/B/C/E’ and ‘D’, respectively), or
- a range of continuous variation (Fig. 17 b to f). Fig. 17 d represents the most frequent case, a categorized continuous measurement variable such as size.

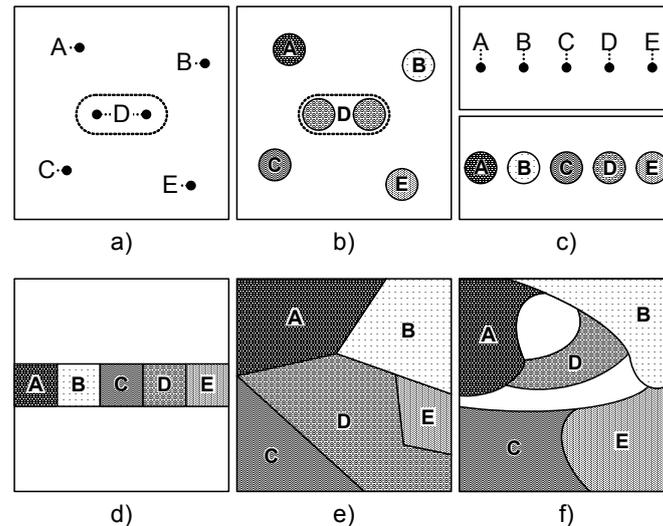


Figure 17. Two-dimensional visualization of potential relations between discrete or continuous reality and categorizations applied to it. Each square represents an example of a value space that is to be classified. The white area represents values that either cannot logically exist, or do not exist in reality. a) Disjointed singular values. Most classes contain a single, 'D' multiple values. b) Disjointed extended (ranges of variation). c) Linearly arranged disjointed categories (top: singular, bottom: extended). d) Connected categories where adjacency creates an exclusive linear order. e) Two-dimensional connected categories where the adjacency may be expressed in a complex graph (i. e., not linear, cyclical, or as a tree). f) Connected categories where adjacency is more complex, either because the relations can no longer be expressed in the two-dimensional space of the visualization, or because – under an operational categorization perspective – parts of the two-dimensional value space are not occupied.

Table 7 gives Examples of the various cases. Whether extended but disjointed categories (Fig. 17b) indeed exist, depends on the perspective. From a functional (or causal) perspective, this is unlikely. It may occur in the case of typologies, where categories are a combination of multiple atomic measurements, some of which may be singular, others extended. However, no good biological example could be found and the result of such a situation is more likely to resemble that of Fig. 17 f, i. e., the categories are partly connected and partly disjointed. If the perspective is, however, to assess continuous or discrete variation based on actual occurrence in nature (see “Operational categorization perspective”, p. 51), it is quite common that no taxon exists in a theoretically possible value space. This results in practically disjointed categories.

Knowledge of this is very useful when devising identification keys, since such a “natural” classification improves the repeatability of identification. In contrast, the categorization of a continuum (Fig. 17 d to f) is much more liable to be interpreted differently by different users of an identification key. A major problem in assessing this is, however, that operational discreteness is often local to taxonomic groups or even geographic scope. Thiele (1993) discusses in detail methods to create categorization of quantitative data based on gaps in the studied data.

Indications of connectedness, ranges, and intermediate values in current standards or software: In **DELTA** (p. 19) the hyphen, when used between immediately adjacent state values, indicates intermediate values (“1-2” instead of “1, 2”, Dallwitz & Paine 2005). In the case of ordinal states, the hyphen primarily indicates a range (i. e., “1-3” = “1, 2, 3”), but it is possible to use “1-2-3” to express a continuum of intermediates. In **SDD** (p. 20) Categorical character data carries a “statemodel” attribute, where one of the values is “Between” (compare Table 22, p. 97). **Prometheus** (p. 21) deliberately supports neither ranges (“1-3”) nor intermediate values (Pullan & al. 2005).

Table 7. Examples of different classes of categorical measurements (compare Fig. 17).

Ordering	Singular (always disjoint)	Extended-Disjointed	Extended-Connected
none (nominal scale)	presence of secondary metabolites	(no example found)	In theory none: connectedness defines order. In practice: categories ordered in a complex graph (e. g., leaf shapes with multiple potential intermediate shapes) will often be treated as nominal to simplify analysis.
linear (ordinal scale)	any character based on counts, e. g., spore septation: aseptate, uni-, bi-, ..., multiseptate	flower color within a genus having red and blue flowers (but not intermediate colors)	immersed, sessile, stipitate
cyclic	phenological events like first leaf or flower in a year	life-cycle stages like egg, caterpillar, butterfly	daily hours, yearly months
complex (n-dimensional, tree or graph)	"types" based on singular elements	"types" involving singular as well as continuous elements, like inflorescence or spore types	most categorical properties!

14. In many object descriptions information about variation underlying the categorization is implied. A better agreed terminology seems to be desirable to accurately express knowledge about ordering, ranges, inclusiveness of border values, and presence of intermediate values in an unambiguous way. This topic requires further study; the terminology presented in Fig. 17, in Table 7 (both above) and 22 (p. 97) is intended only as a first attempt.

Data types in computer programming

Data types have so far been mostly discussed from a data analysis perspective. The following is an inventory of data types that are conveniently available in current high-level programming languages. Without question, computers can handle any conceivable data type and different high-level languages support different types. Nevertheless a broad inventory of "current practices" may help to distinguish between type concepts that may be implemented with little or no effort, and those that require extensive custom programming.

Most computer processors (computer hardware including the microcode embedded in the processor) have native data types and operations for integer and fixed precision floating-point numbers. Usually multiple integer types are supported (e. g., signed or unsigned integers of 8, 16, 32, or 64 bit precision). All have a limited range of supported values (integers with arbitrary precision may be emulated by higher levels of software). Enumerations of categorical values (including Boolean with values 'true' and 'false' and text characters like 'a' to 'z') are represented as bit strings for which comparison and bit-wise logic is defined. Computer processors do not natively handle real (rational and irrational) numbers. In principle, rational numbers could be handled with full precision (and in rare cases this is achieved through software rather than directly in the processor). However, their representation through fixed precision floating-point values is sufficient for most commercial, technical and scientific purposes.

High-level programming languages and DBMS add constraining facets to these two basic types, resulting in type-safe support for, e. g., categorical enumerations, Boolean, positive or negative integers, or floating-point numbers. Interestingly, no computer language known to the author has built-in support for the concepts expressed in the measurement scales. Categorical data are usually supported through enumerations, but no separate types exist for nominal or ordinal enumerations. In many programming languages it is possible to perform a type cast to integer to compare the order of enumerated values (e. g., casting the Unicode letters 'A' and 'B' to byte results in '65' and '66'). The problem with the lack of type support for linearly ordered catego-

ries (ordinal measurement scale) is that the evaluation is not type-safe (i. e., the cast is possible on enumerations on the nominal scale as well). Furthermore, after casting data may also erroneously be used to calculate distances (e. g., “distance between A and B is 1”), which is nonsense because distances are defined only on the interval and ratio scale, but neither for nominal nor ordinal data.

Similarly, no native support for data on the interval versus ratio scale is available in computer languages known to the author. If type-safety is desired, this has to be implemented in a statistical analysis library. Most high-level programming languages offer some support for cyclical data, especially in the form of time and date values. Using the modulo operator (i. e., remainder after integer division) any finite, discrete cyclical data type may be efficiently implemented, and cyclical continuous types (e. g., angular values in degree or radians) require little additional support. Although not normally part of the programming language itself, the angular types may be readily available in mathematical libraries.

Further data types may be represented through sequences of the same (lists, arrays, strings) or different type (structures, records). Complex numbers, vectors, and matrices may be expressed in this way. However, complex numbers have so far not been used in descriptive data in biology, and vectors and matrices are usually assembled dynamically for analysis purposes from their respective dimensions, because in descriptive data the choice of dimensions in a vector usually strongly depends on the needs of data consumers (e. g., for identification purposes, phylogenetic analysis, etc.).

A more relevant example is the sequence of values. It may be desirable to have sequences of all primitive data types to express ordered multiple observations (color, respiration, compass readings). Two kinds of sequences, unconstrained text and molecular sequences, are especially relevant in biological descriptions and are discussed in the following sections.

15. The matching of descriptive data types with those implemented in specific processors, programming languages or DBMS is not a requirement.
16. However, commonly used data types (current practice for other disciplines) may help to guide priorities and are relevant when estimating the cost of implementing a concept.
17. Sequences or arrays of data types may be desirable. Based on current practices (no existing software implements sequences other than free-form text), the requirement seems to be weak.

Unconstrained text

A special data type may be needed to support unconstrained, free-form text created primarily for informing human readers rather than for analytical purposes. Such text may either replace other forms of data (e. g., as in a DELTA “text character”) or it may occur in addition to categorical or quantitative data as an extension mechanism (e. g., as in a DELTA “attribute-comments”).

Although only in relatively few situations free-form text is ultimately the best choice, the presence of free-form text elements greatly enhances the flexibility and acceptance of applications. Free-form text elements allow the gradual introduction of a strict terminology, rather than forcing content authors to solve all terminological problems before starting to enter data. Free-form text is probably the only practical way to express information, if:

- The nature of the descriptive information is such that categorical or quantitative characters seem fundamentally inappropriate. For example, complex arrangements may be difficult to atomize.
- No appropriate terminology exists and the introduction of ad-hoc terminology appears undesirable because it would apply to very few taxa. An example might be the description of relative positions of the inner organs of an organism.
- The terminology is not yet defined in the application and perhaps difficult to define. Before a precise definition is created, a period of testing and discussions with colleagues is planned.

Such text characters may or may not later be converted into categorical, terminology-based characters or character values.

- Current limitations of the exchange format or the application used offer no better alternative. In DELTA it is inconvenient to code host plants or geographical areas as categorical characters because either new states would have to be added continuously, or the characters might have to have several thousand character states. As a result, such data are often treated in free-form text characters. Other examples are molecular sequence or pattern data in the absence or appropriate data types.

Some special properties of text strings are that they may be language-neutral or language-specific (with or without translations/alternative representations) and that often a simple form of text formatting is desired. The topic of free-form text data elements is discussed in more detail in an SDD discussion document (Hagedorn 2003f).

18. A data type for unconstrained, free-form text is desirable. The text may either be an extension of more structured information (such as a character state or quantitative value) or it may replace more structured information.
19. A free-form text extension mechanism (e. g., “comments”, “notes”) may be part of the fundamental information model and always available.
20. Whether free-form text may replace more structured information or not should be controllable by the designers of the terminology.
21. It is desirable to provide support for multilingual free-form text.

Molecular sequence data

Nucleic acid (DNA, RNA) and protein sequences are expressed as a sequence of symbols. Each symbol represents a molecule and the order of the symbols is defined by the chemical bonds between the molecules. The writing direction is standardized by convention and based on the chemical asymmetry of the molecules.

It is common to record molecular sequences as a string of letters, essentially using the data types available for text for molecular sequences as well. Nucleotide sequences are always expressed through single-letter symbols. Protein sequences may either be expressed through one letter or three-letter symbols.

Occasionally, either as a result of ambiguities in original data, or as a result of data aggregation (see “Special aggregation cases”, p. 92) a position in a sequence may be a set of elementary symbols. This situation is usually expressed using “ambiguity symbols” that have the same length as the fundamental symbols. In DNA sequences (with fundamental symbols ‘A’, ‘C’, ‘G’, and ‘T’), Examples of ambiguity symbols are ‘W’ = {A, T}, ‘S’ = {C, G}, or ‘N’ = {A, C, G, T}.

Double-stranded DNA (as found in most organisms) presents a special problem. While molecular sequences are chemically asymmetrical and the reading direction is fixed by convention (5' to 3' for nucleotide sequences), the two complementary DNA strands cannot be distinguished *a priori*, i. e., without recourse to sequence patterns itself (as when using specific PCR primers). For example, instead of the sequence “5'-ACCGTT-3'” the alternative strand with the complementary and reverse sequence “5'-AACGGT-3'” may have been sequenced in the laboratory. This may either require special mechanisms in the information model, or it may be handled as implicit knowledge on the data consumer side.

Molecular sequences are a special data type insofar as the original data are in a form that can only be compared for identity in its entirety (i. e., the entire sequence of categorical states is identical). Additional data processing is required to convert them into an “alignment” that allows assessment of similarity between sequences from different organisms and that allows the comparison of individual sequence positions. An alignment is created by inserting an additional gap-symbol into sequences where insertion or deletion events are assumed to have occurred. The goal

of creating the alignment is, to “homologize” sequence positions, making symbols at the same position comparable throughout the data set.

Various alignment algorithms have been devised (e. g., Clustal, Higgins & Sharp 1989) but the topic remains an active area of research and current algorithms produce only heuristic estimates that often need manual corrections. Unfortunately, the only algorithm available that produces a fully evolutionary correct alignment requires prior knowledge of the phylogeny of organisms (Hein 1989). This is usually not helpful in phylogenetic research, where alignments of sequence data are usually created with the goal of inferring an incompletely known phylogeny. As a result, the alignment remains a special form of data, which is based on the original sequence data, but which cannot simply be modeled as a form of automated data processing prior to analysis.

An additional problem in handling aligned sequence data is that aligned sequences depend on all other sequences in the alignment. Correcting an error in one sequence or adding a new sequence to the set will change the alignment in all other sequences. Some projects attempt to create alignments covering all studied taxa at least for some of the most frequently used genes in phylogenetic analysis (e. g., the ARB project, Ludwig & al. 2004, <http://www.arb-home.de>).

Thus, although in principle the symbols of molecular sequences are categorical data and each position (=column) in the alignment could be treated in a separate categorical variable, the assignment of the position to a variable is stable only under a specific alignment algorithm and for a specific set of aligned sequences. Other forms of categorical data (e. g., morphological data) also require an assessment of phylogenetic comparability to avoid comparing unrelated characters. However, the character identity may be expressed as a set of rules and definitions that can be assessed before recording the descriptive data. This is not possible in the case of molecular sequences. As a result, adding data for a new taxon will only extremely rarely change the interpretation of morphological categorical data from previously studied taxa, whereas it may introduce or move many gap symbols in the aligned sequences already in the data set, thus changing the relation between a base position and the character column significantly.

Whereas this rearrangement is relatively trivial if the alignment is stored as a string of letters, it becomes rather difficult to manage the necessary rearrangements if each base position is already assigned to a specific character. The similarity of a sequence alignment and a character state matrix (e. g., in the NEXUS format) is therefore somewhat deceptive. In essence, NEXUS uses a string for any kind of categorical data, thus solving the specific problem of sequences, but at the trade-off of being incapable to express more than a simple categorical state in descriptive data (no notes, modifiers, etc.). NEXUS is appropriate for the analysis of a defined set of sequences under a specific alignment hypothesis, but not for integrating descriptive data from different sources with the intent of analyzing them in various combinations and with different methods.

In conclusion, when handling molecular sequences (e. g., for identification purposes), it may be necessary to handle both original unaligned sequence data and multiple aligned representations (a sequence may be a member of multiple alignments, each developed specifically for a given set of taxa). The problem of multiple alignments has not been pursued further in the current work and in the information models presented here. However it is clear that molecular sequences involve special problems and form a data type of their own.

22. Molecular sequence data are a specialized data type that is highly relevant to current biological research.
23. Original sequences and sets of aligned sequences are related, but different data types.
24. The length of individual symbols may be assumed to be constant throughout the sequence, but it may be longer than one letter.
25. Sequence positions with ambiguous data may be expressed through special symbols rather than requiring a separate syntax.
26. It is desirable to provide a mapping from ambiguity symbols to the set of fundamental symbols they represent.

Complex quantitative data types

Many (Stevens 1991, Thiele 1993) or perhaps all (Baum 1988) categorical characters ultimately may be measured by a quantitative method. In contrast to “typical” univariate quantitative measurements of angle, size, etc., however, these measurements are expressed in a more complex way, e. g., through coefficient vectors or matrices. Examples are the various ways to measure color or shape.

Quantitative color measurements may be expressed through spectrographic density functions or through color-space models. Each recording method produces a number of quantitative variables that allow more exact color comparisons than categorical color measurements (vernacular “red”, “green”, “blue”, etc. or codes in standard color charts). Each type of color recording has advantages and disadvantages:

- Categorical color names already include the psychological factors of human color perception and support simple data recording and communication.
- Color-space models (sRGB, AdobeRGB, CIE XYZ, etc.) allow quantitative expressions that are optimized for available color reporting methods (print, computer screen). Some colors like gold, silver, or many shades of brown, can only be expressed inadequately in color models, but then these colors cannot be reproduced on computer screens or with standard color printing techniques either. Color models allow the expression of color variation by defining an area (e. g., an ellipse or polygon) in color space. Computer-aided recording of variation in such model is possible. Recordings using a digital camera (based on color-space models as well) may be used to obtain approximate values (ignoring color perception).
- Spectrographic color reflectivity data are more exact than color model measurements. However, they have several disadvantages as well: the need for expensive recording equipment normally available only in laboratories, the need to deal with problems of human color perception (which is related to spectrographic color reflectivity in rather complicated ways), the limitation of being unable to create an equivalent color perception within the limits of computer screen or available printing techniques. Also, expressing *color variation* through spectrographical data requires highly complex data storage models, and may be very difficult to present in an aggregated (summarized) form.

The measurement of shapes may be even more complicated than color. Two-dimensional shape outlines may be recorded and reproduced by fitting mathematical functions to the shape (e. g., Bezier-equations or spline functions). Although these equations accurately reproduce a shape, they perform poorly in comparing similar shapes: Identical shapes may be represented by different sets of these functions. Studies exploring the use of methods of geometric morphometrics are, for example, Jensen (1990) or Rohlf (1996). Recently a new “superformula” with six parameters was proposed for a large number of shapes that supports a high number of symmetry axes, capturing, for example, circular, elliptic, triangular, rectangular, penta- and hexagonal shapes (Gielis 2003). The wide range of shapes covered and the relatively low number of parameters may allow better comparability. Also, the mapping of shape-ranges to shape categories may be defined in a more meaningful way.

Another interesting development is the recording and analysis not only of outline shapes, but of complex two- or three-dimensional topological arrangements. For example, the vein structure of hymenopteran fore-wings (Steinhage & al. 2001) or the venation of plant leaves (Kirchgeßner & al. 2002) have been studied. Agarwal & al. (2006), while working on two-dimensional leaf outlines, describe a vision of automated identification based on three-dimensional plant images. Three-dimensional complex shapes may be modeled using techniques developed for virtual reality. However, similar to 2-dimensional shapes, these techniques are optimized for reproduction of a shape, but do not necessarily facilitate the comparison of descriptions, e. g., during identification. See also the topic “Automatic identification” (p. 231).

The examples so far (color, shape) concern morphological data traditionally recorded as categorical data. Other measurements are almost impossible to record without support for complex quantitative data in the information model. Examples are raw or processed chromatographic data (Elix & al. 1988), or molecular pattern data (AFLP, RFLP, RAPD, etc.).

Most data types necessary for complex measurements can probably be modeled through very basic data types:

- binary objects,
- lists or sequences of floating-point values (compare requirement 17, p. 56), or
- one- or multidimensional arrays of floating-point values.

Using a general data type in all cases requires some separate metadata mechanism to inform analytical procedures how to interpret the data. The ties between data and data processing methods depend on the complexity and depth of documentation of the storage format. Often, binary data may be more complicated to process, but arrays or lists of floating-point values may have few advantages until the arrangement and the semantic interpretation of the elements are known to the processing software. Lists or sequences may be more general, but also more difficult to interpret. In real sequences of simple data types, all values refer to the same kind of quantity and are either all dimensionless or have the same measurement unit, whereas in a list of parameter-values each element in the list may bear different semantics. Furthermore, to express variability or to record time series data, sequences of complex data may be necessary. Again, technically it is possible to record a color polygon in RGB space simply as “51, 129, 9, 46, 208, 2, 1, 231, 61, 7, 155, 59” rather than as (R=51, G=129, B=9); (R=46, G=208, B=2); (R=1, G=231, B=61); (R=7, G=155, B=59)”, but methods reporting the data or searching with the data (e. g., for identification) must be appropriately informed about the additional structure.

Whether such reuse of generalized data types is more appropriate than defining specialized data types for color, shape, molecular pattern, etc. depends on how many complex types will be used in practice, what kind of link between general data type and concrete semantics needs to be stored, how important type-safety is to guarantee interpretable values, and how difficult it is to add new data types to existing data storage structures and data processing software.

27. Although many characteristics of an object are expressible through categorical variables, more complex data types are often desirable for data such as color or shape. Such data are often expressed using multiple values. Depending on the method the number and semantics of values may be fixed or variable. The descriptive information model should either provide a general model for all conceivable complex data types or should provide extension mechanisms to support the addition of additional complex types.
28. It is desirable to support sequences or sets of complex data values, e. g., to record shape variation or a color polygon.

Media data

Traditionally, media data (images, audio, or video data) are not considered a “data type” of descriptive data. The border between complex quantitative chromatographic or molecular pattern data on the one side, and image or sound data on the other side is, however, not easily definable. Whether a media object is considered to complement or support other information (“illustration” or “voucher”), or whether it is considered a “complex quantitative data type” in its own right, largely depends on the availability of methods that enable automatic comparisons or analyses.

Furthermore, with the improved information density of digital media it becomes more and more feasible to consider them primary objects that require descriptions and identification. Examples are studies using high resolution scans of soil, rock, or bark surfaces that contain lichens to be identified.

Implemented data type systems

To help in assessing priorities among the data type requirements defined so far, this section discusses the data type system of those computer applications for descriptive data in biology, for which information is available.

DELTA (p. 19) supports five character types (Table 8). The character types ‘UM’ and ‘OM’ support unordered (nominal) and ordered (ordinal) categorical data. Both types support lists of multiple values (values being called “attributes” in DELTA, compare Table 3, p. 34). Such a “multistate” list is a sequence of values, where the order of values may be significant (the analytical semantics of order are undefined, but order is to be preserved at least for generating reports for human consumers). Each value within such a categorical value list is a complex data type insofar as it may be associated with unconstrained text notes.

DELTA supports quantitative data in the character types ‘IN’ and ‘RN’ (for integer and real numeric, respectively). These types appear to be similar to those used in programming languages, but this is slightly deceptive. Each type may either contain a distinct value list (e. g., “2/4/8”, compare p. 86) or using one of twelve different patterns (structured by hyphens and parentheses, e. g., “(3-) 5-7-9 (-11)”), expressing a fixed set of statistical measures (see “Quantitative data and statistical measures”, Table 31, p. 112 for further information). “1.5” is a valid value for an integer (‘IN’) data type, since the mean of integer values may be real numeric.

The DELTA ‘TE’ character defined for unconstrained text is closely similar to simple string in a programming language. However, any character of the types ‘UM’, ‘OM’, ‘RN’, ‘IN’ can in practice also be used as a text character (i. e., it is possible to add a “comment” without a state or a value, the resulting data are indistinguishable from ‘TE’-based data). Thus, declaring a character as ‘TE’ only prevents adding categorical or numeric data. This behavior is not explicitly defined in the user guide, but implicit in existing data sets and the behavior of applications supporting DELTA. Implicitly, the ‘TE’-type acts as the base-type from which all other types are derived.

Table 8. DELTA character types (Dallwitz & al. 2000a).

Code	Name	Description
UM	Unordered Multistate	Multistate (including 2-state) characters in which the states are not arranged in a natural order.
OM	Ordered Multistate	Multistate characters in which the states are arranged in a natural order.
IN	Integer Numeric	Numeric characters which take only integer (whole-number) values.
RN	Real Numeric	Numeric characters which may take fractional or integer values.
TE	Text	Unconstrained text

“**New DELTA**” (p. 20) proposes to add two additional data types to DELTA:

- “LI”: lists of text tokens, e. g., for geographical or host organism names.
- “CY”: cyclically ordered categories.

In addition, a generally new structure “numbered lists” shall be introduced, which essentially allows to give text strings (for taxon names, literature, etc.) an ID-number, by which they may be referred elsewhere. This is not considered a “character type” in the proposal. Furthermore, New DELTA proposes a special symbol for “indefinite values” in quantitative characters, covering the case of “many” or “large”. The system is not extensible; categories like “few” or “tiny” that may occur together with quantitative measurements are not expressible.

DiversityDescriptions tries to remain compatible with DELTA for import and export purposes and therefore supports all five original DELTA character types. However, internally it supports only text, categorical and quantitative types, and treats the further distinctions as character metadata. Additional metadata, allowing finer distinctions are supported. The model for quantitative data is extended over DELTA, allowing the data set author to define an unlimited number of

statistical concepts (see “Quantitative data and statistical measures”, Table 31, p. 112 for further information). A special design feature of DiversityDescriptions is that it enables combining categorical and quantitative data types in a single character. This feature was added to simplify capturing data that use a mixed representation of quantitative and categorical values (e. g., 1, 2, ..., 12, 20, “many”). The information model outlined in Diederich (1997) suggests a similar mixed representation, although only for situations like “many” or “few” (called “fuzzy states”).

CBIT Lucid (p. 21) distinguishes between categorical and quantitative data types. Quantitative data support four statistical measures (Table 31, p. 112). Lucid supports no text characters.

Nemisys/Genisys (p. 21) introduces a specialized “type system”, which is discussed in detail in the following section “Basic property types”. It supports three statistical measures (Table 31, p. 112).

Prometheus description model (p. 21) largely follows the Nemisys/Genisys model, distinguishing quantitative and qualitative “description elements” (their preferred term for character). The support for statistical measures is not discussed in the publications and therefore unknown.

NEXUS (p. 18) supports both categorical (“Format DataType = Standard”) and quantitative data (“DataType = Continuous”, compare p. 112 and Table 31, p. 112, for further information). NEXUS has further special data types for molecular sequence data: DNA, RNA, Nucleotide (i. e., DNA or RNA), and Protein.

SDD (p. 20) distinguishes primarily between text, categorical and quantitative data. Information about measurement scale (nested within categorical and quantitative), and discrete versus continuous is added as properties (or “metadata” in the sense of Diederich & Milton 1991, Diederich & al. 1997) to each character definition. The fundamental split between categorical and quantitative data is similar to that implemented in DiversityDescriptions or Lucid, and the one proposed in Diederich (1997) – but the model details differ.

The data type system in SDD is designed to be extensible; specialized character types for color measurements, molecular sequences, etc. have been discussed, but not yet finalized in the first versions of the standard.

29. The data type systems implemented in current descriptive software or exchange formats may dictate secondary requirements where import or export to these systems is intended.
30. Quantitative data may occur together with categorical states like “few”, “many”, “large”, or “very small”; a general “indefinite large” and “indefinite small” may form a minimum requirement. A more extensible method may be desirable.

Basic property types

In the context of the Nemisys/Genisys model, Diederich, Fortuner and Milton developed a special classification system for quantitative and categorical data called “basic properties” (Diederich & al. 1997, 1998, Table 9). The system primarily covers morpho-anatomical descriptive data. The authors do not use the term “type” or “data type” for the “basic properties” they define. Instead, basic properties are developed as part of a character decomposition system (see “The Nemisys/Genisys model”, p. 117). They do point out, however, that some fundamental semantics of basic properties must be further defined through a “default range” with the values “binary, discrete, continuous” and a measurement scale with the supported values “nominal, ordinal, interval, ratio” (Diederich & al. 1997).

Table 9. “Basic morpho-anatomical properties” proposed in Diederich & al. (1997, 1998; identically). A “*” indicates properties expressing a relation between two structures.

Appearance	Dimension
Posture	Length
Shape	Height
Kind	Width
Texture	Diameter
Arrangement	Depth
Symmetry	Ratio-of *
Placement/Location	Size
Position-relative-to *	Quantity
Distance-to *	Presence
Orientation	Quantity
Angle	Number

(Differences in Diederich (1997) are: “Presence” is classified in Appearance rather than Quantity; “Posture” is missing; “Color” is accepted as a basic property within Appearance – rather than presumably being subsumed in “Kind” as shown above).

Identifying type-like concepts on a higher level than data-analysis has several advantages in that semantics and processing rules may be defined closer to the descriptive language (with less abstraction) and in a reusable way. On the other hand, the list above is incomplete, and some points may be argued, e. g.:

- “Ratio-of” is not a basic property acting on an object part (structure), since it must be combined with a basic property like size, length, width. It is a relation between two parts plus a property, rather than a single property acting on two parts. Diederich & al. (1997) mention this (guideline 5), but they do not propose a solution to handle this in an information model. The example shows that the simplicity of basic properties as a short, flat list is somewhat deceptive.
- In addition to an absolute “Angle” (presumably relative to the earth surface, in normal living conditions), a relational expression “Angle *” between two structures seems to be missing.
- Why is “Presence” not a special case of “Quantity = 0”?
- Why are certain quantitative measurements recognized under “Dimension” whereas others (weight, temperature, speed, conductivity, etc.) are subsumed under “Number”?
- Why is diameter (of circle) recognized as a special property, uniquely different from length, height, or width, but parameters of other shapes (e. g., “eccentricity of an ellipse”) not?
- Why is depth recognized, but not thickness?
- Presence and Quantity should be relational properties (“Presence-at *”, defaulting to the entire organism). They express a part-of relation (optionally with multiplicity) between two object parts (structures). In many cases it does not matter whether a specific structure is the named or not (e. g., “eyes present” → two insect eyes are part of the insect head, “insect wing count = 4” → 4 wings are part-of thorax), but in other cases this does matter (e. g., “three bars on hind-wing”, “bristles at tip of antennae”).

Most notable, as discussed in Diederich & al. (1997), all properties not considered of primary importance are subsumed under catch-all “generic properties”:

- unassigned categorical properties under “Kind”, and
- unassigned quantitative properties under “Quantity” or “Number”.

Table 10. A modified basic-property-concept extended with a generalization hierarchy (example).

Appearance	Cardinality/Multiplicity
Shape	Presence or existence
Two-dimensional (perhaps: open / closed?)	Count
Three-dimensional (perhaps: open / closed?)	Extension/Size
Symmetry	Length
Texture	Height
Roughness/Smoothness	Width
Surface ornamentation	Diameter / Radius
Hairiness	Thickness
Color	Depth
Vernacular color name categories	Area
Color chart values	Volume
color space values (sRGB, HSL, etc.)	Placement/Location
Pigmentation	Position-relative-to *
Color granularity	Distance-to *
Taste/Smell	Orientation
Five basic taste sensations:	Absolute orientation
acid, salty, sweet, bitter, umami	Orientation relative-to *
smell (human taste impressions are a combination of these and smell sensations)	Angle
	against absolute orientation
	Angle-to *

Ratio (including, e. g., Density) is excluded because it has to be considered on a higher level. A mapping exists between “count” and “presence” that may allow to calculate one property from the other: three legs → legs present, legs present → at least one leg, one leg → legs absent, legs absent → zero legs. – No special property of Boolean type is considered: Coding “winged/wingless” is not different from “wings present: true/false”, and any property with exactly two states (set of two things) are equivalent to Boolean. Sets of only two states or things do have special properties in calculations, but software can deduce this without a need to introduce a Boolean data or property type.

Which properties are considered more important, depends to a substantial degree on the organism group and the methodology used to describe organisms. It seems problematic to embed such decisions into the general model. For example, while some basic properties already provide reusable sets (enumerations) of states (i. e. categorical values), no reuse is possible for properties subsumed under “kind” – although reuse would be desirable. An example for a property hierarchy that is richer than Diederich's (as in Table 9) is shown in Table 10. This is, however, only another selection of preferred properties, informed by a specific point of view. Other views would lead to a different selection.

Diederich & al. (1998) argue that the concept of basic properties may be extended to cover physiological data, but doing so requires the introduction of abstract concepts (such as “resting period”) in the place of structures. These pseudostructures then may have properties like presence, duration, etc. However, it seems that much of the clarity and reusability that is present with morphological data is lost by doing so. Most notably, it is not possible to have these pseudostructures refer to specific morphological structures. The approach offers a way to occasionally include physiological data, but it appears a fix rather than a general solution.

One consequence of these problems is that basic properties will probably be under revision for an extended period of time. Designers of information models would be wise to provide a generalization mechanism rather than limiting themselves to those basic properties shown above. To the present author's knowledge no information model for basic properties has been published yet that would achieve both the desired generalization and specificity of basic property types. Further studies are necessary to decide whether “basic properties” indeed are fundamental enough to justify the development of such a model.

An interesting example for the kind of problems encountered with basic properties is that even the seemingly intuitive example of “shape” as a basic property seems to be full of pitfalls. Going back to the example in Fig. 15 (p. 45) it is obvious that shape is a typology to classify objects. “Shape” in Fig. 15 uses categories that combine more fundamental attributes or properties (number of corners, symmetry, number and length of linear sections, angles, etc.). Thus shape can also be viewed as a “summary character” in the sense of Diederich (1997) and Diederich & al. (1997),

which stands in contrast to their view of shape as a basic and atomic property. Nevertheless, the shape typology is indeed in most scenarios a very basic, useful element in descriptions. A combination of the constituent properties/attributes is difficult to recognize and visualize, whereas the shape type names are easily recognized and remembered by humans. However, the same argument applies to most typologies that under the basic property concept should be treated as summary characters and split into more atomic characters.

The fundamental question is therefore often how much information to include in a complex type, and where to split it into multiple elements. For example, one might want to define equal-sided triangle or right-angled triangle as additional types. Conversely, the definition of a square with rounded corners as a type will become questionable as soon as triangles, hexagons, etc. with rounded corners are to be described. Either one might extract “rounded corners” as an additional property from the complex “shape type”, or one may want to model an additional substructure “corner” (present only for some shape type values!) with a property “rounded”. These are just examples of the kind of terminological instability and inconsistency that Diederich and coworkers seek to prevent.

Furthermore, shape values are a complicated mixture of singular (“distinct”, like triangle, four-sided polygon, pentangle, etc.) and extended categories (i. e., shape instances may be intermediate between categories). Fig. 17f on p. 54 (with some categories connected but others distinct) has been drawn with the example of “shape” in mind. Thus, in addition to the question of decomposition of shape types into properties, also the definition of the resulting categorical values will be under discussion. For example, the ellipse shown in the second row of Fig. 15 (p. 45) is a very weak ellipse that may well be called “nearly circular”. Since in biology exact shapes almost never occur, it would be natural to a biologist to do so (calling it “subglobose” in biological terminology). A very similar problem occurs when attempting to separate the aspects of rounded corners from the shape typology, since an infinite number of intermediate shapes exists between a square with rounded corners and a circle (Fig. 18).

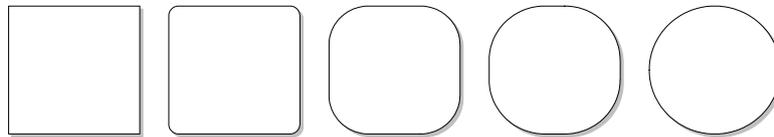


Figure 18. Squares with rounded corners form a continuum of intermediate shapes between a square and a circle.

The probable consequence of the problems discussed here is that the information model should expect terminological change and should not (as the basic property model seems to do) aim to reach stability by fixing a set of “basic” properties as special and unchangeable. Further, although it is desirable that different data sets use the same terminology, it is probably more important to focus on comparability. One way to do this is to make the assumptions behind the terminology readily available to machine-reasoning. An attractive model may be to treat values in complex characters (type-value, such as shape) as objects that again may have descriptions in terminologies using more completely atomized properties. However, many alternatives are conceivable (e. g., multiple class inheritance: a triangle with rounded corners being both an “object with rounded corners” and a “polygon with three sides”). A more detailed analysis of actual data storage models follows in a later section (“Description storage models”, p. 104).

31. “Basic properties” according to the Nemisys/Genisys model are a derived type system optimized for morpho-anatomical data. The simple system of 20 basic properties with one level of hierarchy offers pragmatic guidance for structuring such data, but is incomplete and not suitable as a general information model for descriptive data. The selection of quantitative measures not subsumed under “Quantity” or “Number”, and the selection of categorical properties not subsumed under “Kind” may be pragmatic for common morpho-anatomical data but is not essential.
32. A property classification, preferably with more than one level of hierarchy, is desirable to structure descriptive data. The model should be able to cope with different generalization hierarchies of properties, rather than fixing these.
33. A generalized “property type” system is conceivable, but will be complex and may be expected to be under considerable terminological evolution for an extended period. The information model must be able to support property information in a way that does not affect existing applications relying on the information model.
34. The information model should provide means to make description based on different terminologies (using different property choices, different level of decomposition of value types, such as shape) comparable using machine-reasoning.

4.6. Mapping between data types

As already mentioned, the same fundamental biological phenomenon can often be measured in various ways, resulting in data on different measurement scales (p. 49). Color may be measured quantitatively (spectrographically or as color model values, see p. 59), or categorically by comparing it with the fine categories of a color standard, or by referring to common “fuzzy” vernacular color names. Clearly these data are related and it is desirable that the information model provides mechanism to make relations explicit. The following sections discuss various cases.

Mapping univariate continuous measurements to categories

The most common mapping in statistical data analysis is the reduction from ratio or interval scale to the ordinal scale. This is done by grouping (discrete or continuous) values into classes with defined limits (class intervals) and usually results in a quantitative frequency distribution (e. g., Zar 1984, Sokal & Rohlf 1981). If the categorized data are continuous, the frequency distribution is also called a *histogram* (Zar 1984). Another mathematical way to express the process is to call it a “partition of a continuous value space” (R. Morris, pers. comm.).

Example: For identification purposes it is desirable to reduce a continuous size measurement to a small number of states like “small (< 10 cm)”, “medium size (10 cm to 2 m)”, and “large (> 2 m)”. In printed dichotomous keys this is the only option, but even where the software could query quantitative data directly, it is often desirable to request a coarse estimate instead of an exact measurement.

The fundamental process of measuring a continuous variable and applying class intervals is illustrated in Fig. 19. In DELTA the “Keystates” directive allows a method to express class intervals (a range from upper to lower range limit). Intervals may or may not be overlapping; overlapping definitions enable improved handling of border-values. The name of the DELTA directive is somewhat misleading; the directive was originally intended for the purpose of automatically creating dichotomous keys.

Statistical measures other than the mean may be used as the basis for a mapping process. When the categories are used for identification purposes it may be desirable to use a range measure like a confidence interval (Fig. 20) or even the extreme values.

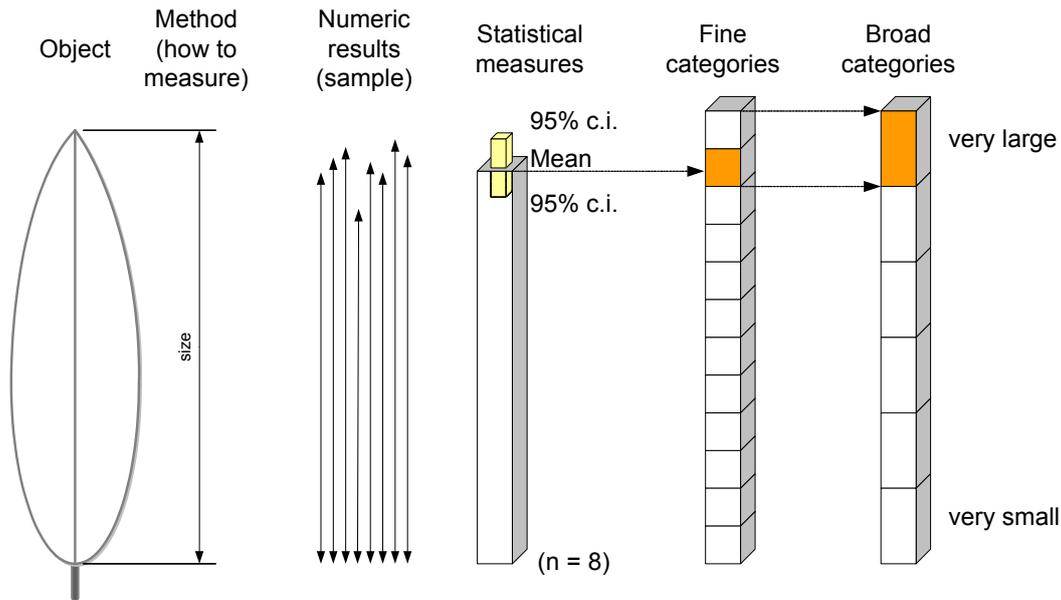


Figure 19. Mapping of quantitative length measurements to fine and coarse categorizations. Based on definitions of object and methodology a sample of measurements is obtained. This is aggregated (or “mapped”) to descriptive statistical measures (mean, confidence interval), which may be interpreted (or “mapped”) as categories with defined limits.

Table 11. Example for mixed (quantitative and categorical) data recording. Bundle scars are small dots or lines on the surface of a leaf scar marking the remains of the vascular bundles that supported the leaf before it fell off.

Bundle scars	Discussion
5	Single value, from the context it is implied that this is the only value, rather than a mean or mode
5, 7, or 9	Discrete distribution without frequency information (set)
5, rarely 7	Discrete distribution, partially with frequency information
5-7 or 9	Containing a range, presumably ‘6’ is included, but ‘8’ not.
many	Categorical value, a limit for “many” may or may not be defined exactly

(From “SDD data challenge: Collections of numerical values and mixed numeric and categorical statements”, www.diversitycampus.net/TDWG-SDD/Docs/SDD_DC_NumericalMixedCollection.html, based on an example by Stephen Seiberling from Flora North America).

Occasionally measurement data have a mixed data type with some data being quantitative, others categorical (see example in Table 11). The categorical values are typically “indefinite values” like “many”, “tiny”, “huge”, etc. (compare requirement 30, p. 62). In principle, defining the point at which data are no longer expressed quantitatively defines a mapping. In practice, however, mixed data occur because it is too time-consuming to obtain exact values for a quantitative property outside some range that can be conveniently measured. Thus the categorical values are the only available data and cannot be obtained from a mapping. The example from Table 11 could be solved by treating the values “5, 6, 7, 9, many” as categories on the ordinal scale. This solution is frequently used in DELTA data sets. However, the quantitative variable may be floating-point or integer but requiring statistical measures like mean and variance (e. g., plant height, recorded quantitatively up to ca. 3 m, then estimated in broad categories like “small tree or shrub”, “large tree”).

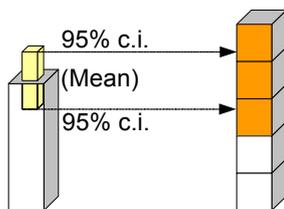


Figure 20. Mappings from quantitative to categorical data may be based on confidence intervals rather than mean (modified detail from Fig. 19).

Diederich (1997) discusses a similar example under the aspect that the exact value of “many” may not be known. He introduces the concept of “fuzzy states” and the outline of an information model provides for storing “fuzzy states” together with quantitative values in a single character. However, whether a mapping is known or not, is secondary. It is quite possible that the class interval for “many” is exactly defined (e. g., “greater than 12”). The situation of mixed data representation is unchanged, but it is no longer a “fuzzy state”.

35. Mappings from univariate continuous data to categorical data should be supported.
36. A mapping for a category may be based on a single range with two limits, or a list of values or ranges.
37. A mapping may be based on single values or several statistical measures. The preferred source for the mapping should be definable.
38. Mixed forms of data (some data are quantitative, others are categorical with defined quantitative limits, other categorical with no or ambiguous definitions) should be supported. If fulfilled, this covers automatically requirement 30 (p. 62).

Mappings within categorical data

A logical extension of the mapping from continuous to categorical data is to map a narrow categorization to a broader one (i. e., map a fine-grained to a coarse-grained categorization). For linearly ordered data the process is very similar to that described above for continuous measurements. In practice, continuous data are always mapped to categories of finite precision, e. g., length measured to the closest mm. A minor difference when mapping from data on the ordinal scale to another categorization on the ordinal scale is that different statistical measures must be used (e. g., mode or median).

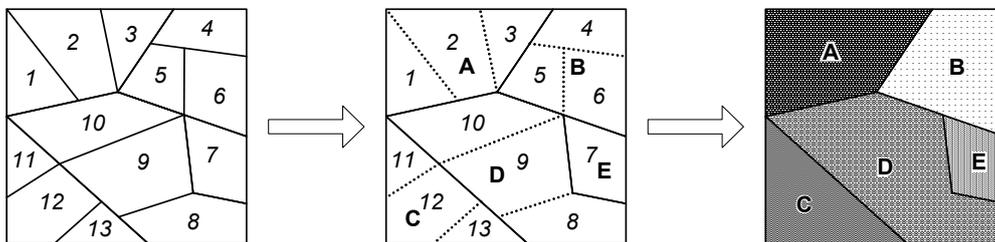


Figure 21. Symbolic presentation of a “value space” that is partitioned into fine-grained categories and for which a generalization mapping (center) to coarse-grained categories (right) is provided. Compare Fig. 17, p. 54.

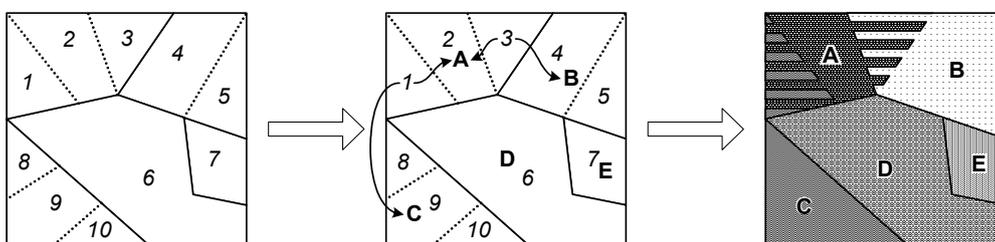


Figure 22. Mappings may provide for ambiguity or error tolerance. Arrows in the center drawing indicate the mapping of two selected fine-grained categories: a) Category ‘3’ is considered intermediate between ‘A’ and ‘B’. b) Objects belonging to category ‘8’ are occasionally mis-scored as belonging to ‘1’; Category ‘1’ is mapped to both A and C to improve error tolerance for identification. Compare Fig. 21 for the simple case.

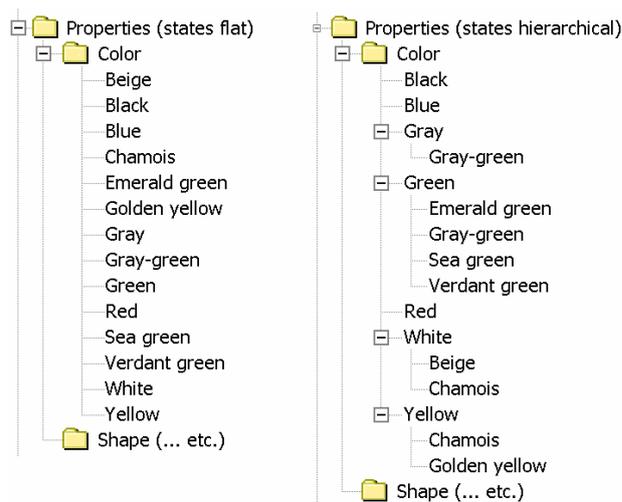


Figure 23. Illustration of a potential tree-view user interface displaying ambiguously mapped categorical values (right side). The property states “Chamois” and “Gray-Green” have multiple mappings (i. e., the concepts are considered ambiguous) and therefore included in multiple parents.

Mapping is, however, also highly useful for categories that cannot be brought into a linear order. One set of well-defined categorical states may be useful and highly expressive for an expert, but less so for beginners. During identification, even experts may prefer another set containing fewer states that have a broad definition. For example, all of: “lanceolate (symmetrical), obliquely lanceolate, oblanceolate (symmetrical), obliquely oblanceolate” may be as “lanceolate” with a definition indicating that this is *sensu lato* (Fig. 21). For this purpose it does not matter whether the categorical data are on a nominal scale or ordered in the form of a graph. Allkin & Bisby (1988) call such corresponding data representations “sister characters”.

Mapping of categorizations may be used both to represent strict generalization relationships (1 : n mapping, Fig. 21) and ambiguous relationships (n : m mapping,

Fig. 22). The latter is relevant when states are intermediate or to improve error tolerance during identification when states are likely to be misinterpreted. Note: Although the illustrations in Figs. 21-22 show a connected values space, the categories may just as well be disjointed.

The mapping of categorical states may be seen as a special case of a generalization hierarchy (kind-of relations) and could be expressed in the ontology web language (OWL, McGuinness & van Harmelen 2004). Another way to express the first three mappings in Fig. 21 would be to say ‘1’ is a kind of ‘A’, ‘2’ is a kind of ‘A’, and ‘3’ is a kind of ‘A’. In Fig. 22 one could also say ambiguously that ‘3’ is a kind of ‘A’ and ‘3’ is a kind of ‘B’. Ideally these statements might be more information rich: ‘3’ is correctly interpreted as ‘A’ and ‘3’ is misinterpreted as ‘B’, or one might say ‘3’ was poorly and ambiguously defined and without access to original data it is impossible to assess whether it really is ‘A’ or ‘B’. This (and the relation that “is misinterpreted as”) is a special case of “is not a kind of”) may, however, be difficult to express in OWL.

Where generalization/refinement mappings are defined within states of the same character, a user interface may use the information to display the states in a hierarchical manner, e. g., in a tree where the narrower definitions are listed as children of the wider definitions. Note that this includes cases of ambiguous mappings (Fig. 23).

39. Mappings from narrow to broad categories should be supported.
40. Ambiguous mappings, where one narrow category is mapped to more than one broad category should be supported.
41. A hierarchy of categories within a single property or character is expressible with a general mapping mechanism and no additional support needs to be added to the information model.

Mapping complex quantitative data to categorical data

Complex measurements (as discussed in “Complex quantitative data types”, p. 59) are usually made with the help of instruments. In the future, recording devices will often be directly coupled to computers, so that data recording becomes automated. If all data were recorded and analyzed by machines (when building the knowledge base of descriptions and during later identifications),

Table 12. Different scenarios depending on the data recording format used during identification and the format available in the knowledge base used for comparison.

Observation method during identification	— Format in knowledge base —	
	Categorical (by human)	Complex measurement (by machine) ¹
Categorical (by human)	The classical identification by comparing categories; a mapping to complex measurements is optional. If present, the computer may provide an improved visualization of the variation within each category (based on the complex values for the categorical class borders implicit in the mapping).	A mapping is required to compare knowledge base data to identification data. Further, similar to the categorical/categorical case it may be used to provide a visualization of categorical ranges. During the confirmation phase of the identification process, exact representations based directly on measurements in the knowledge base may be provided.
Complex measurement (by machine)	Allows accurate measurement of the object being identified. A mapping is required to compare categorical "legacy" data in the knowledge base data to identification data. The measurements obtained during identification may be stored to form the basis of the future complex measurement knowledge base.	Direct and optimal comparison of observations possible. A human intelligible representation of the measurement results is, however, desirable to avoid recording errors. Although a mapping to categorical data is optional, it may help in the task of human proofreading.

¹ A major problem of both cases in this column is that it will take a long time until the existing knowledge base of categorical data will be replaced with new measurements.

mapping to categorical expressions of these data types would be redundant. In the meantime, mechanisms that allow mapping these data to categorical expressions help in communicating results to humans (Table 12).

Neither complex measurement systems nor corresponding mappings are currently included in the SDD model. In the development version a "ColorRange" type is maintained to indicate the desire for further development in this area, and to clarify that extensions to the character type model are to be expected. The mapping mechanism in SDD are designed to be part of a specific description model (currently only univariate quantitative and categorical measurements are supported) so that appropriate mapping methods supporting, e. g., polygonal areas in color space, would be defined for new methods.

42. Special mapping mechanisms are desirable where complex quantitative data are defined.
 43. A complex mapping may help applications provide visualizations of the extent and variability of categorical states to humans. It may therefore be desirable even if no complex quantitative data are actually recorded.

Mappings and definition of categories

All three forms of mapping definitions discussed so far implicitly define the categories to which the mapping points. While this is generally desirable, it also demonstrates that some commonly used categories are poorly defined and obtain their definition from implicit and context-dependent knowledge. For example, a set of states such as "large", "mid-sized", "small" may have been re-used in multiple characters such as "body size" or "hair size", with widely different semantics. Furthermore, even in one character the meaning of "large" differs between whales and microbes.

One solution to this is to define multiple, property, object-part, and taxonomic scope dependent sets of "size categories". Clearly this leads to an inflation of sets. An alternative may be to provide special mechanisms in the information model to deal with this problem. It remains unclear which option is preferable.

44. The failure to define general mappings may point to categorical definitions that are problematic or context-dependent. Whether it is more desirable to have multiple sets of states with exact definitions, or a single set of "generalized" states with multiple, context-dependent mappings remains an open question.

Mapping unconstrained text to structured data

Interpreting natural language description under a defined terminology to obtain structured description is closely related to a mapping process. Two scenarios may be distinguished here:

- The natural language description is digitized and will be preserved as such. In this case markup referring to concepts, characters, states, etc. in the terminology may be added. A mechanism to do this is provided in the NaturalLanguageDescription type in SDD (p. 20).
- The natural language description is considered only as a data source. However, the mapping process that is implicit in interpreting descriptions under a defined terminology will be documented. Doing so can help with issues of schema evolution, and improve proofreading or the collaboration of multiple researchers. Under this scenario some part of the original text may be preserved together with the conclusions.

A variant of the second scenario is the lexicon mapping between “Cited states” and “Display states” proposed in Diederich (1997). “Cited states” are defined as those used in the published descriptions in the data source, and the “Display states” are those used in the current (and revisable) terminology. It is unclear from Diederich (1997) whether cited states are simply text labels, or whether they are considered structured machine-readable information. Their main described function is to be interpreted by humans; storing natural language text may therefore be sufficient. The advantage of making this mapping explicit is that it facilitates schema evolution. The model is clearly very helpful when digitizing legacy information, but it should perhaps be generalized to cover situations where different mappings for different audiences (e. g., taxonomic experts, students, customs personnel using identification tools) are desirable, or for cases where no published source exists and the taxonomic experts enter data directly (the most frequent scenario in which DELTA has been used so far).

Another variant is the problem that in a multilingually defined terminology, discrepancies between translations of term definitions will occur and lead to further schema evolution issues. To help with these issues it may be useful to record together with each coded term also the original label (or at least version and language) of a term that was seen by the author creating the description (proposed as an extension to the SDD standard by T. Paterson, pers. comm.).

45. It is desirable to express the relation between free-form, unconstrained text and descriptive terminology in a special form of mapping. This mapping differs from those discussed so far in that it is defined in descriptive data rather than in terminology. This is more similar to a markup process (like html) than to formalized, mathematical mappings.
46. It is desirable to be able to define a mapping between an original form of free-form text and its translations.

Mappings involving more than two characters

In biological descriptions, a general tension exists between atomic and complex characters (“summary characters”, Diederich 1997, Diederich & al. 1997). The human mind is more readily adapted to names referring to complex concepts by a single name, than to processing a list of characteristics that together expresses the same information. The naming of organisms itself is an example of extremely complex descriptions associated with a single term. For example, in botany the different inflorescence types are given different names. As discussed under “Basic property types”, p. 62, however, already the common “shape” property is truly a complex character that can be atomized.

Currently SDD is able to map categorical states to one or multiple states in one or multiple characters. It is, however, not yet able to handle a combination of states on the “from” side (Char. 1, state 2 ‘and’ Char. 2, state 4 → Char. 3, state 1).

47. Mappings that express the relationship between complex characters (“typologies”) and multiple basic or atomic characters are desirable.

Calculated characters

Many of the mappings discussed so far may be generalized as a function projecting one value space into another. This may be a simple function (e. g., to convert °F to °C), or a categorization of data, ratios, area, etc. (Table 13). However, even where the fundamental function is relatively simple, rather complex dependencies on data aggregation (p. 83), properties, and measurement methodology may exist, as the following examples show.

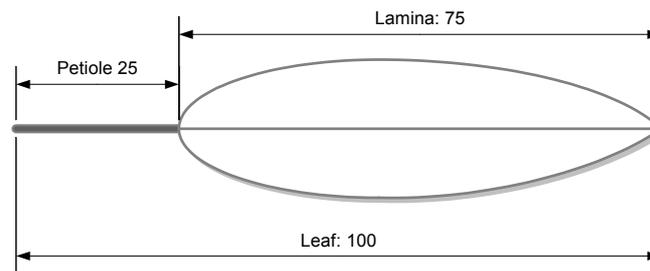


Figure 24. Multiple measurements may have fewer degrees of freedom than variables.

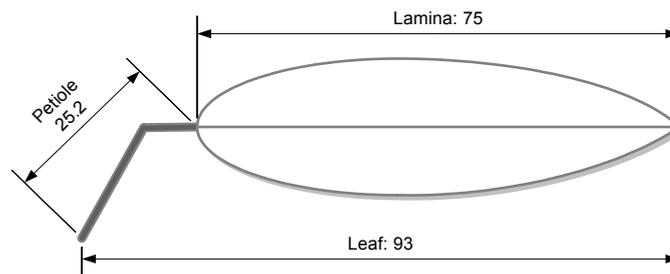


Figure 25. Depending on measuring method, length measures may not be additive.

Ratios: The most common ratios in descriptive data are based on measurements of the same dimension (or dimensionless, like counts). Examples are the ratio of the length of two parts, or the length/width-ratio of a single part of the object. Such ratios are especially convenient where parts are next to each other in a composition (e. g., sepals and petals in flowers, compare Fig. 53, p. 138); such a situation allows estimates (e. g., “smaller”, “about equal”, “larger”, or “twice as large”) with ease and precision. Calculations based on different dimensions are relatively rare. An exception is perhaps counts per extension or area (for example, density of hairs). Under the “basic property model” (p. 62), a ratio may thus be based on 1 or 2 parts and 1 or 2 properties.

As mentioned above (see Table 20, p. 91), ratios are an example where a calculation based on summarized measurements gives a different result than the calculation based on repeated measurements. When defining a calculation method, the scope of this method (repeated measurements or statistical measures like mean) should thus be defined.

Sums: In many cases the total length of a composite object may be calculated based on the lengths of individual parts (Fig. 24). If the individual lengths are additive, fewer degrees of freedom than variables exist. Note that, depending on how the measurement method has been defined, an interaction with other properties may exist. In Fig. 25 the measurement of the petiole is defined in a way that the length of lamina and petiole are additive only if the petiole is not angled (not “geniculate”).

Table 13. Examples of potentially useful mappings of measurement characters to “calculated” characters.

Source	Destination	Examples and Notes
Single quantitative measurement	Quantitative	Conversion of units like Fahrenheit → Celsius
Single quantitative measurement	Categorical	Direct measurement to classification: < 10 μm, 10-20 μm, > 20 μm
Single quantitative + categorical measurement	Quantitative	Spore width measured separately for septate/aseptate or hyaline/brown spores
Multiple quantitative measurements	Quantitative	Length/width ratio, area, volume, etc.
Multiple quantitative measurements	Categorical	Comparative statements like “Petals shorter than sepals”
Multiple quantitative + categorical measurement	Quantitative	Area or volume could be calculated separately for different shapes

No current information model or implementation supports a generalized function mapping. In general, support for calculated characters encounters the following problems:

- A sufficiently general and powerful mathematical language is required to define complex formulas. The language should, however, not be limited to a specific programming language (assuming terminological definitions should remain exchangeable). An option might be a programming-language-independent standard like MathML (Carlisle & al. 2003), but this would create a heavy burden on implementers while still lacking many concepts (current MathML is very weak on statistical functions and concepts). When the topic was discussed in the SDD no satisfactory solution could be found (see SDD minutes: Hagedorn 2003b). A possible path could be defining a subset of MathML that covers most practical needs and at the same time reduces implementation cost to a realistic level.
- The categorization of a quantitative measurement variable into classes (histogram) and corresponding categorical states requires knowledge of how these categories must be addressed. This mixes issues of a general mathematical language (which would be able to express the class intervals through ‘>’ and ‘<’ operators) and specific issues of the descriptive model. For categorical mappings it would be necessary to support mathematical concepts from category theory like “functors” (i. e., a generalization of functions that associate every object of one category with an object of another category).
- The available information is incomplete and different authors or publications may record different aspects of a complex of dependent/derived properties (see Table 14). As a consequence, some “character variables” may either be a calculated result or an original value.
 - In principle, calculated properties are similar to derived class attributes in UML. However, it seems that UML implicitly assumes an inverse function (the implementation may decide which attributes to store and which to calculate). However, the calculations used most frequently in descriptive data have no inverse function (e. g., aggregated ratios, categorization, or statistical measures such as average or variance).
 - It might be possible to create two variables, one containing calculated, the other original values. This would put a heavy burden on queries and consumers to understand the semantics of this relation. SDD proposes to add attributes on data instead, informing about the origin of a value.
- In some situations it is desirable to be able to record the parts of the description that are available and remain true to the data sources. This may include:
 - Recording values as they are published, even if they may be calculated from other variables (over-determined data).
 - Recording values even if some values contradict others, based on the assumptions implicit in the calculation method. Not only is it usually not immediately clear which values are correct and which false, but also, as the example in Fig. 25 shows, assumptions may be erroneous. Preserving possibly contradictory data, perhaps adding annotation to this fact, is preferable to ad-hoc decisions of which one is the correct value that must be made because the system only allows the entry of “correct” data.

Table 14. Example showing potential combinations of available measurements for leaf length measurements (compare Fig. 24).

Petiole length	Lamina length	Leaf length	Note
(can be calculated)	70 mm	80 mm	= minimally complete
10 mm	(can be calculated)	80 mm	= minimally complete
10 mm	70 mm	(can be calculated)	= minimally complete
10 mm	?	?	= incomplete
?	?	80 mm	= incomplete
?	70 mm	?	= incomplete
15	70	80 mm	= over-defined & contradictory!

Note that “formulas” or methods to define calculated values discussed here are different from the “Formula characters” directive proposed for “New DELTA” (p. 20); these “Formula characters” are designed as natural language formatting commands.

DELIA (p. 19) uses the term “derived characters” instead of “calculated characters”.

48. Support for calculated character values (based on one or multiple values from other characters) is desirable.
49. A standardized support for calculations in a descriptive information model is, however, highly problematic, both because of possibly complex dependencies and because notation systems for formulas are either specific of certain programming languages, or general but difficult to implement in a wide variety of applications. Support for calculated characters is not a priority.

4.7. Coding status

Data may be missing from a description for a variety of reasons and it is often relevant to have some classification of why this is so. The most basic distinction is between “data entry is incomplete or erroneous”, and “data cannot possibly be supplied”. Databases usually use a *Null* or *Nothing* value to indicate missing data or incompleteness. Examples of potential semantics of the Null value in databases are “absence of data”, “unknown data”, “undefined”, “not applicable”, and “to be added later” (from the documentation of Microsoft SQL Server 2000). This semantic “polymorphism” causes interoperability problems when exchanging data. A richer terminology is therefore desirable.

Existing forms of such “coding status” or “knowledge management” metadata in descriptive data are called “pseudo-values” (or “special symbols”) in DELTA (compare Table 15), “special states” in DiversityDescriptions (p. 322), and coding status values in SDD (compare Table 16). Coding status values may be interpreted in various generalization hierarchies (Fig. 26). The SDD

Table 15. Coding status values (also known as “pseudo-values”, “special symbols”, or “special states”) in DELTA.

Coding status	DELTA symbol	Notes
“no data recorded”	(implied → no symbol)	Usually interpreted as “not yet scored” or “unfinished work”
“unknown”	‘U’	The character scoring is “minimally complete”.
“not applicable”	‘_’	Manual alternative to declarative general character applicabilities. It may be used together with other states (e. g., in genus descriptions where a character may have a state in some species, but may be inapplicable in other species).
“variable”	‘V’	“Variable” may express a polymorphism, variability, or saturation (i. e., all states are true) of a character. It should <i>not</i> be used together with other states. Deprecated in DiversityDescriptions.

proposal “Indicators of coding status in class or object descriptions” discusses the topic in detail (Hagedorn 2004b). It includes detailed information on each proposed value as well as SDD XML instance examples, discussions of current practices, exclusiveness, information aggregation issues (in the sense discussed further below on p. 83), and the differences between coding status metadata of a variable and metadata (frequency and certainty modifiers, see p. 206 and 207) of values.

Table 16. Coding status values defined in SDD 1.0 and 1.1. This enumeration extends the DELTA values ‘U’ and ‘-’, but drops the DELTA “V = variable” value. The semantics of the latter are difficult to define and its current application is doubtful.

Coding status value	Possible Symbol	Description	Basic coding status	Information in general	Information in source data set: ¹
“To be checked”	“!”	Explicit indicator to revisit a character later. This may be used when data are missing (known to exist, but not at hand for entering) or together with data (check data against additional information source).	To be coded	May exist	Does not exist
“Not to be coded”	“~”	A decision was made not to enter data	Not to be coded	May exist	Does not exist
“Not applicable”	“-”	For logical reasons, it is assumed that data cannot exist.	Cannot be coded	Cannot exist	Does not exist
“Data unavailable”	“?”	Data could not be obtained despite that an effort was made.	Cannot be coded	May exist	Does not exist
“Not interpretable”	“#”	Data are known to exist, but are purposely not coded because not even an interpretation involving certainty modifiers was deemed possible.	Cannot be coded	Exists	Does not exist
“Data withheld”	“\$”	Data are present, but are not disclosed (e. g., because private or confidential).	Coded successfully	Exists	Exists
<i>(For the following coding status situations no explicit coding status values are defined; the status is implied:)</i>					
“Data recorded successfully”		(Evident from existence of data)	Coded successfully	Exists	Exists
“No data recorded”		(Implicit in not coding the character in the description at all; since neither a value nor a coding status is present this is also “status not yet evaluated”)	Not evaluated	May exist	Does not exist

¹ ‘Information in source data set’ refers to data storage (document or database) from which the current representation that includes the coding status values was directly or indirectly derived. The distinction is relevant in the case of ‘data withheld’, where no data exist in the current data set, but information exists in the source.

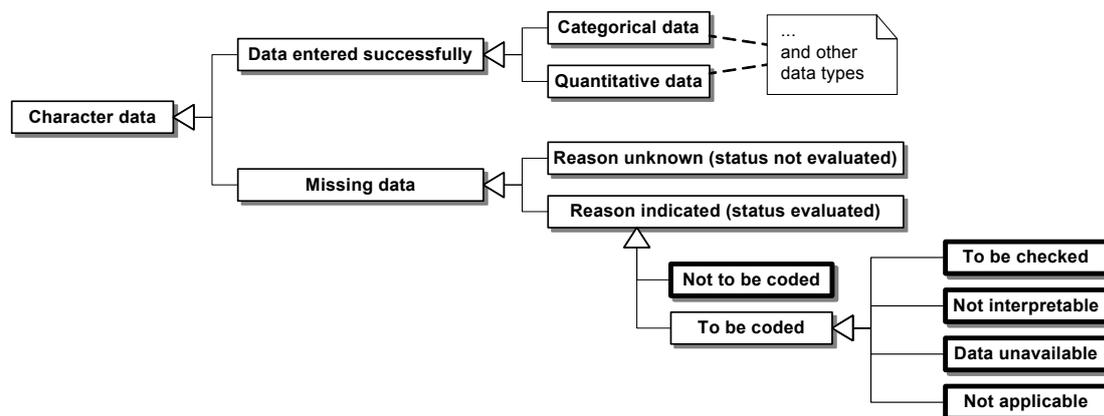


Figure 26. Generalization hierarchy of character data and coding status values. This diagram is intended to clarify intent, not as an architecture plan for implementations. In SDD (p. 20) values for the five classes with a thick border are explicitly present; the remaining classes are deduced from other data or complete lack of coding.

50. Support for coding status information in the information model is a central requirement to support knowledge management and collaboration scenarios for descriptive data.
51. The existence of categorical or quantitative data as well as the lack of any data in a description for a character that is defined in the terminology may be considered implicit forms of coding status.
52. A predefined list of coding status values is desirable to support interoperability. The hierarchical nature of coding status information may be implicit and does not have to be expressed in the data.

4.8. Character dependency

Character dependency in general

For phylogenetic and statistical analysis purposes, character data should ideally be independent and identically-distributed random variables (“i.i.d.” or “IID” criterion, see Felsenstein 1985, Swofford 1990, 2000). In practice, this can rarely be fully achieved (compare “Analysis of character correlation” in the use case chapter, p. 303). If a correlation between two characters is complete, one of these characters is redundant and may indeed be replaced by a calculated character (see p. 72).

This is not possible, however, if the value of another character may be predicted only for certain values of a controlling character. Expressing information about character correlations in some form of dependency rules does not add information content to perfectly correct descriptive data sets. A simple correlation analysis would suffice to obtain the information. However, most data sets contain factual and coding errors. Separately defined rules may then improve data entry, analysis, and identification tasks.

A general form of such a character-value correlation may be desirable, where defined values of a categorical or quantitative character determine the values or coding status values of another character. It is, however, neither elaborated here nor implemented in any descriptive models known to the author.

It could not be assessed whether or which form of character dependency the Prometheus description model (p. 21) supports. Because the composition is built newly for each data recording, data cannot be internally inconsistent (e. g., recording leaves absent and leaf properties). However, it is unclear whether non-structural dependencies are supported.

Character applicability rules

Current character dependency models (e. g., DELTA, p. 19; DiversityDescriptions, p. 322, XPER² and CBIT Lucid3, p. 21, see also Dallwitz 2006) focus on a special case of character-value dependency: certain values (states) of a categorical character determine whether another character is applicable or not (e. g., if “leaves absent” the leaf shape becomes inapplicable). In practice inapplicability based on categorical states is a frequent case and highly relevant for the efficiency of data recording and identification. However, it seems desirable to reserve the term “character dependency” for the general forms of character correlation and dependency. In contrast to the use in DELTA, the term “character applicability” is therefore preferred in this thesis for this special case.

The defining character in a character applicability relation is commonly called the *controlling* or *parent character*, the other the *controlled*, *dependent*, *child* or *applicable/inapplicable character*. The values (categorical states) of the controlling character contained in a character applicability rule may be called *controlling states*, the remaining states defined in the character *non-controlling states*.

In current character applicability models, quantitative characters can be controlled but cannot be controlling. Examples where the applicability of characters depends on a specific value range of a continuously varying quantitative character are difficult to find. One example may be that for very small organisms or cell organelles (e. g., $<0.5 \mu\text{m}$) light microscopic characters are no longer applicable (compare “Dependencies on circumstances of identification”, p. 175). However, a dependency on counts of object parts is frequent. Supporting only categorical controlling characters forces the developers of a descriptive terminology to artificially split the character informing about the multiplicity of an object composition (see “Describing object multiplicity”, p. 145) into two characters, one categorical for presence/absence, another for the number of objects if at least one is present. The latter is then usually controlled by the first one.

Character applicability rules may be expressed positively and negatively, called “inapplicable-if” and “applicable-if” rules here. In the terminology used in this thesis, they may be formulated as:

- **“Inapplicable-if”**: If only controlling states are present (i. e., “recorded” or “scored”) in a description, these make the controlled character inapplicable. The character remains applicable if either no state at all, or any non-controlling states are present.
- **“Applicable-if”**: If any controlling state is present (i. e., “recorded” or “scored”) in a description, these make the controlled character applicable. It is inapplicable if only non-controlling states are present.

Two special conditions are common to both forms:

- If the controlling character is inapplicable (through another applicability rule, or through an explicit ‘inapplicable’ coding status value), the controlled character is always inapplicable as well (see “Cascading character applicability rules”, p. 82).
- If no data for the controlling character are present in a description (data completely missing, or only equivalent coding status; see p. 74), the controlled character always remains applicable. In the case of an *applicable-if*-rule this definition is slightly unintuitive. However, separating the issue of default applicability from the form of the rule is a) generally desirable and b) required to achieve convertibility of the two forms (compare “Convertibility of applicability rules” below).

DELTA uses two directives (“Applicable/Inapplicable Characters”). The definitions in the user guide differ only in a single word (Dallwitz & al. 2000a): “*This directive specifies the values of ‘controlling’ characters which make other ‘dependent’ characters [‘applicable’ in the definition of the Applicable Characters directive, ‘inapplicable’ in the definition of the “Inapplicable Characters” directive]. If, in a given item, a controlling character takes only values which make its dependent characters inapplicable, or if the controlling character itself is inapplicable, then the dependent characters must not be given any values (other than the pseudo-value ‘inapplicable’, which is redundant [...]).*” Dallwitz (2006d) writes further: “*If any of the recorded values of a given controlling character do not make its dependent characters inapplicable, then the dependent characters may be recorded. For example, ‘leaves present or absent’ allows other leaf characters to be recorded.*”

These definitions are equivalent to those given above (the “*values which make its dependent characters inapplicable*” are the controlling states for *inapplicable characters* and the non-controlling states for *applicable characters*), except if the controlling character contains no data in a description. The “applicable-if” definition given above assumes that the default of the controlled character is inapplicable, whereas in DELTA the controlled character remains applicable.

Before the convertibility of the two forms of character applicability rules can be discussed, the evaluation of the rules may need some clarifications:

A character may control multiple dependent characters (each with one or multiple controlling states) and a character may be controlled by multiple controlling characters ($n:m$ relation). In the first case, applicability rules may be evaluated completely independently. For the second case, Dallwitz (2006d) defines the intended evaluation for DELTA as: “*If a given dependent character*

is dependent on more than one controlling character, then the dependent character can be recorded only if allowed by all of its controlling attributes.” Note: in Dallwitz's terminology, an *attribute* is the set of all values for a single character in a single description.

Table 17. Examples of evaluating applicable/inapplicable-if rules involving multiple states and the same combination of controlling/controlled character.

States present for controlling character 1 in given description	Result of <i>Inap- plicable-if</i> rules 1.a <input checked="" type="checkbox"/> → 2 <input type="checkbox"/> , 1.b <input checked="" type="checkbox"/> → 2 <input type="checkbox"/>	Result of <i>Applicable-if</i> rules 1.a <input checked="" type="checkbox"/> → 2 <input checked="" type="checkbox"/> , 1.b <input checked="" type="checkbox"/> → 2 <input checked="" type="checkbox"/>
{1: a <input checked="" type="checkbox"/> , b <input checked="" type="checkbox"/> , c <input checked="" type="checkbox"/> }	2 <input checked="" type="checkbox"/>	2 <input checked="" type="checkbox"/>
{1: a <input checked="" type="checkbox"/> , b <input checked="" type="checkbox"/> , c <input type="checkbox"/> }	2 <input type="checkbox"/>	2 <input checked="" type="checkbox"/>
{1: a <input checked="" type="checkbox"/> , b <input type="checkbox"/> , c <input checked="" type="checkbox"/> }	2 <input checked="" type="checkbox"/>	2 <input checked="" type="checkbox"/>
{1: a <input checked="" type="checkbox"/> , b <input type="checkbox"/> , c <input type="checkbox"/> }	2 <input type="checkbox"/>	2 <input checked="" type="checkbox"/>
{1: a <input type="checkbox"/> , b <input checked="" type="checkbox"/> , c <input checked="" type="checkbox"/> }	2 <input checked="" type="checkbox"/>	2 <input checked="" type="checkbox"/>
{1: a <input type="checkbox"/> , b <input checked="" type="checkbox"/> , c <input type="checkbox"/> }	2 <input type="checkbox"/>	2 <input checked="" type="checkbox"/>
{1: a <input type="checkbox"/> , b <input type="checkbox"/> , c <input checked="" type="checkbox"/> }	2 <input checked="" type="checkbox"/>	2 <input type="checkbox"/>
{1: a <input type="checkbox"/> , b <input type="checkbox"/> , c <input type="checkbox"/> }	2 <input checked="" type="checkbox"/>	2 <input checked="" type="checkbox"/> ¹

and indicate whether a state (indicated by lower case letters) in a description is scored or not; e. g., “1.a ” indicates that state ‘a’ of char. ‘1’ is present in a description. Applicable or inapplicable are represented by and .

¹ Applicability in the case of no information for the controlling character: A literal interpretation of “applicable if state scored” would be “inapplicable”; however the DELTA interpretation is that the controlling character is unknown, the result of the rule unknown, and therefore the controlled character applicable (see p. 80).

Furthermore, more than one state in a controlling character may control the same controlled character (Table 17). These states are to be evaluated together, splitting the states defined for a controlling character into a set of controlling states and a set of non-controlling states. Although the definitions given above are sufficient, a few points may be highlighted to avoid misinterpretations:

- The set of controlling states is defined for each controlled character independently. Adding a rule “1.c → 3 ” to the examples in Table 17 would not change the set of controlling states for character 2.
- If multiple controlling states are present, any of these invokes the inapplicability (“1.a *or* 1.b” in Table 17). It is not possible to establish a rule that is invoked only if multiple states (e. g., “1.a *and* 1.b” in Table 17) are present in a description.
- The evaluation behavior within and between controlling characters is the opposite.
 - If within a single controlling character (and description) multiple states are present, any state that makes the controlled character *applicable* will result in the controlled character being applicable. In the case of an *applicable-if*-rule such a state is any controlling state, in the case of an *inapplicable-if*-rule such a state is any non-controlling state.
 - If within a single description multiple controlling characters are defined, any character that makes the controlled character *inapplicable* will result in the controlled character being *inapplicable*.

(For the difference within and between characters, compare also “Boolean operators between states of categorical characters”, p. 95, and “Boolean operators between characters”, p. 98.)

Note that one may count all controlling states in a controlling character as a single rule, or one may count each controlling state as a rule of its own. In some scenarios of character evolution and federated (distributed) development of terminology the latter is desirable. However, this is a secondary matter of terminology and does not change the evaluation rules.

Convertibility of applicability rules

General considerations

The presence of two complementary forms of character applicability rules (*applicable-if* and *inapplicable-if*) is an unwelcome complication of any information model. The rules as shown above are convertible by applying the complementary rule to the complement of states (Figs. 27 and 29; analyzed in detail in the following section). The convertibility depends on knowledge of the coding status of characters. When modeling characters as variables and states as values a presence/absence of a part is expressed in two-state values (not in a single value being scored or not). Under this model, a character has always at least one value scored or entered as soon as the observation or measurement has taken place. In contrast under a state scoring model, the coding status of the character may be unknown (or needs to be managed separately, compare “Categorical data: Character matrix vs. character state matrix”, p. 104 ff).

The convertibility of the two DELTA directives is expressed in examples in the DELTA User's Guide (Dallwitz & al. 2000a) and has been used in the design of the DiversityDescriptions model (see p. 322) which only supports inapplicable-if rules. During import of DELTA data, the “Applicable Characters” directive is converted by adding an “Inapplicable Characters” directive to the complement of states (Hagedorn 1999a).

However, experience with DiversityDescriptions, has shown that it is undesirable to support only one form of applicability rules. The original solution in DELTA of providing separate directives is preferable. Two major issues could be identified:

- In many cases, applicable-if rules are easier to formulate and proofread. For example, a controlling character may record plant hairs as *simple*, *stellate*, *glandular*, etc. and controlled characters may contain special information for the various kinds of plant hairs, (if *stellate*: number of tip branches, ratio-branched/unbranched; if *glandular*: length/width of glandular head, etc.). Expressing this through applicability rules is logical and straightforward, expressing it as inapplicability if only other states are scored requires substantial logical effort when creating or proofreading the rules.
- Converted character applicability rules may deteriorate over time if new states are added to a character after the rule has been created (Fig. 28). As discussed in “Static versus dynamic terminology models” (p. 45), evolution of the descriptive terminology over time should be expected and appropriate provisions made.

As a consequence, it is considered desirable to preserve the original form of the rule (as it was intuitive to the creator of the rule). For the purpose of evaluating character applicability rules, however, convertibility is highly useful and simplifies the design of a system.

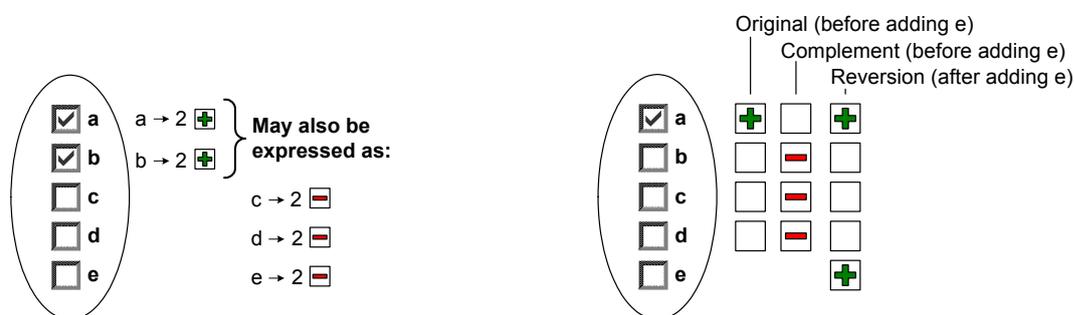


Figure 27. A character applicability rule defining “char. 2” as inapplicable if a given state is scored may be converted into the complementary rule applied to the complementary set of character states.

Figure 28. Original applicable-if-semantics are difficult to preserve when new states are added to a character after the original rule has been converted to the complementary inapplicable-if rule.

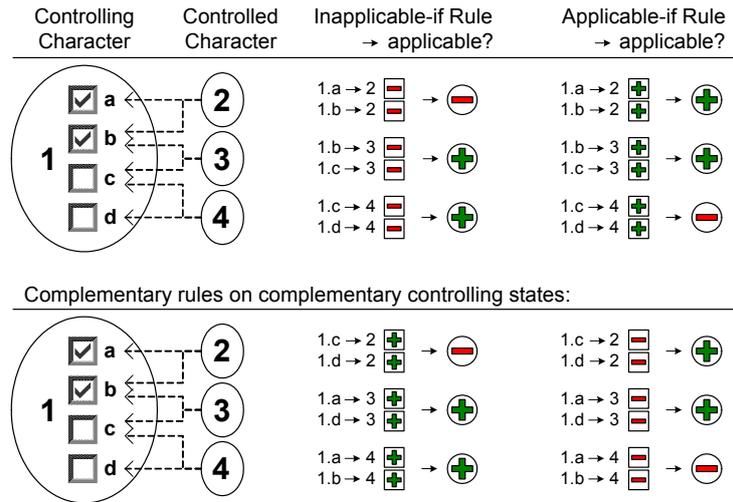


Figure 29. The results of an inapplicable-if rule, and the results of the complementary applicable-if rule applied to complementary states are identical. Small boxed plus and minus represent applicable-if and inapplicable-if rules, large circled plus/minus the final result given scored states shown are present in a description.

Analysis of convertibility

The convertibility of applicability and inapplicability rules depends strongly on some border conditions. The question whether or under which conditions applicability rules are indeed convertible was repeatedly raised in the SDD discussions, so that a slightly more detailed analysis of this seems to be justified.

Definitions: S is the set of states defined in the controlling character, I is the set of states making the controlled character inapplicable and A is the set of states under which the controlled character remains applicable. I and A are complements with respect to S , i. e., the states defined for a character are the union of I and A ($S = I \cup A$) and the intersection of the sets of controlling and non-controlling states is empty ($I \cap A = \emptyset$). D is the set of values (i. e. states) of the controlling character present (“scored”) in a given description ($D \subseteq S$), v and w are state values of this character ($v \in D$).

1. Controlling character in description is absent ($D = \emptyset$), coding status may be explicitly “unknown”: A literal interpretation of “*applicable-if*” might suggest that the controlled character is inapplicable if no applicable-if rule is in effect. However, if no state of the controlling character is present in a description, the value of this character is unknown; thus the result of the rule (i. e., the applicability of the controlled character) is unknown. This case is therefore not affected by a conversion between applicable-if to inapplicable-if rules (compare last line in Table 17 above).

A potential requirement, desiring a method to decide whether to present potentially inapplicable characters shall be presented to the user or not must be treated separately, not through *inapplicable-if* and *applicable-if* rules. It may, for example, be handled through a preference setting of the editing or identification application for the case $D = \emptyset$.

2. Controlling character inapplicable ($\forall v: (v = \text{Inapplicable})$): Similarly, the issue of applicability if the controlling character itself is inapplicable is handled identically for both forms of rules and does not affect the convertibility analysis. An inapplicable controlling character makes the controlled character inapplicable, see “Cascading character applicability rules”, p. 82. A special case is that the explicit coding status “inapplicable” may occur together with other character data or other coding status values, in which case it shall be ignored.

3. Applicability rules based on data present in a description ($D \neq \emptyset$): The “verbal” character applicability rules formulated above (p. 77) may be formally written as:

a) For “Inapplicable-if” rules: I is the set of controlling states mentioned in the rules and A is the set of other, non-controlling states.

$$\begin{aligned} IsInapplicable &= (\exists v : v \in I) \wedge (\neg \exists w : w \in A) \\ &= (\exists v : v \in I) \wedge (\forall w : w \notin A) \end{aligned}$$

Because of the relations between I , A , and D this may be simplified to:

$$\begin{aligned} IsInapplicable &= \forall v : (v \in I \wedge v \notin A) \\ &= \forall v : (v \in I) \\ &= \forall v : (v \notin A) \end{aligned}$$

Consequently the applicability is:

$$\begin{aligned} IsApplicable &= \neg IsInapplicable \\ &= \exists v : (v \notin I) \\ &= \exists v : (v \in A) \end{aligned}$$

Or, alternatively:

$$\begin{aligned} IsInapplicable &= (D \cap I = D) \\ &= (D \cap A = \emptyset) \\ IsApplicable &= (D \cap I \neq D) \\ &= (D \cap A \neq \emptyset) \end{aligned}$$

b) For “Applicable-if” rules: A is the set of controlling states mentioned in the rules and I is the set of other, non-controlling states.

$$\begin{aligned} IsApplicable &= (\exists v : v \in A) \\ &= \neg(\forall v : v \in I) \end{aligned}$$

Consequently the applicability is:

$$\begin{aligned} IsInapplicable &= \neg IsApplicable \\ &= \forall v : (v \in I) \\ &= \forall v : (v \notin A) \end{aligned}$$

Or, alternatively:

$$\begin{aligned} IsApplicable &= (D \cap I \neq D) \\ &= (D \cap A \neq \emptyset) \\ IsInapplicable &= (D \cap I = D) \\ &= (D \cap A = \emptyset) \end{aligned}$$

The rules thus mirror each other, only exchanging the association between controlling/non-controlling set of states and the sets I and A . Thus by applying the complementary rule to the complement of states, the rules may be converted.

The complete evaluation logic, including the case that a character is controlled by several controlling characters may be found in the appendix (p. 388) as pseudo-code and in SQL.

Coexistence of character applicability rules

The DELTA directives “Applicable Characters” and “Inapplicable Characters” (= “Dependent Characters”) may not occur together in the same data set (Dallwitz & al. 2000a). Considering the advantages of the two forms of rules discussed above, this seems to be an undesirable constraint. Problems may arise only if *applicable-if* and *inapplicable-if* rules address the same combination of controlling and controlled character. In this case the two forms may either be identical (and thus one is redundant) or contradictory (Fig. 30). An explicit contradiction arises if a controlling state is listed in both rules, an implicit contradiction arises if a state is mentioned in neither rules (and consequently the complementary forms of both rules are in explicit contradiction).

Although it would be possible to define precedence rules, the results could easily lead to confusion. Essentially, using the two forms of the rule is not different from defining two, differing

	No contradiction (redundant)	Explicit contradiction	Implicit contradiction
	a → 2	a → 2	a → 2
	b → 2	b → 2	b → 2
	c → 2	c → 2 c → 2	(c → ?)
	d → 2	d → 2	d → 2
	e → 2	e → 2	e → 2

sets of controlling states under the same form. It seems therefore desirable to require that for any combination of controlling and controlled character only a single form of applicability rule may be defined.

This would allow a single character to be controlled by different

Figure 30. Character applicability rules using both applicable-if (plus symbols) and inapplicable-if (minus-symbols) rules for the same combination of controlling and controlled character may be explicitly or implicitly contradictory.

controlling characters, each using *inapplicable-if* or *applicable-if* rules, and a controlling character to use both forms of the rules for different controlled characters.

Two secondary requirements for applicability rules may be:

- the controlling states truly form a set, i. e., in the database or exchange format no state may be listed twice
- the set of controlling states I (or A , respectively) must be smaller than the set of states defined for the controlling character (S), i. e., I is a *proper* subset of S : $I \subset S$ (or $I \subset A$, respectively).

Cascading character applicability rules

The controlling character in an applicability rule may itself be controlled by another character value (Fig. 31). This may lead to deeply cascading applicability relations.

In DELTA, cascading definitions are supported and appropriately evaluated. Note that according to Dallwitz & al. (2000a), a character whose controlling character is inapplicable should always be considered inapplicable, regardless of whether it is covered by an applicable-if or an inapplicable-if rule.

The DiversityDescriptions model supports cascading definitions in the data, but the Diversity-Descriptions application currently does not evaluate cascaded definitions (Hagedorn 1999a). It is not possible to use standard SQL queries for such recursive evaluations where the depth of recursion is unknown. In practice, this can be overcome by making “Petioles (hairiness)”, etc. directly dependent on “Leaves (presence)”. Naturally, this is less convenient and more error-prone.

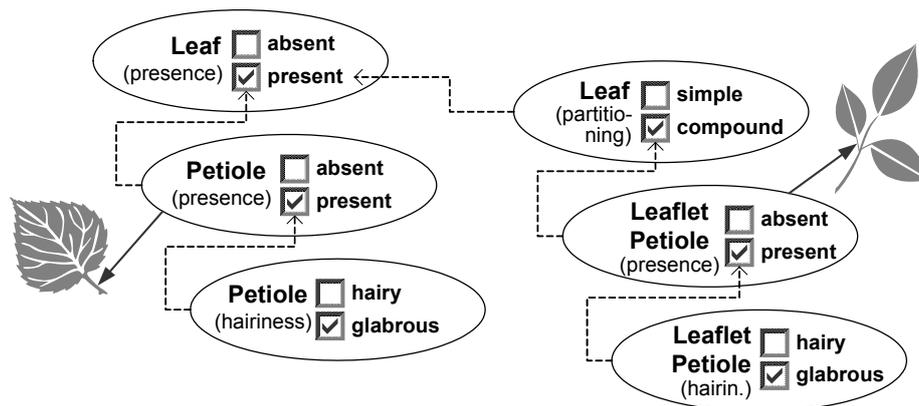


Figure 31. Character applicability rules may cascade several levels deep.

Character dependency and applicability is also shown in the use case diagrams on definition of descriptive terminology (Fig. 165, p. 285) and character correlation analysis (Fig. 188, p. 305). The use of character applicability to improve character guidance algorithms for identification purposes is discussed on p. 274.

Current support in some applications and data standards

- Support in *DELTA* and *New DELTA* is identical and has already been discussed above.
- *NEXUS* (p. 18) does not support character dependency. As mentioned above (p. 76), methods of phylogenetic inference like maximum parsimony or maximum likelihood assume that characters are independently, identically distributed. In practice it is known that this requirement is almost never fulfilled and that statistical inferences (e. g., bootstrapping, Felsenstein 1985) will yield only approximately correct results. However, the kind of absolute character dependency discussed here as character applicability seems to be not foreseen in NEXUS.

- *DiversityDescriptions* supports the DELTA “Applicable” and “Inapplicable Characters” directives in import, but internally converts them to inapplicable-if rules (p. 329). As discussed above, this causes problems in the context of character evolution, if states are added to a character where the rule would be most logically expressed as an applicable-if rule.
- *CBIT Lucid* (p. 21) supports a similar model to DELTA.
- *SDD* (p. 20) supports both *applicable-if* and *inapplicable-if* rules. A major innovation in SDD is that the dependencies may be defined as part of normal concept hierarchies (i. e. character trees). They are inherited down the concept hierarchy tree, affecting all characters below a given node (i. e., direct or indirect through other nodes). The advantage of this design is that whereas in other models the rules are anonymous and often difficult to maintain during the evolution of the terminology, in SDD they are attached to labeled nodes, which in themselves have a logical hierarchy. In addition, these nodes often already exist. For example, in a compositional concept hierarchy the leaf-node will be a natural place for leaf-dependency rules. SDD up to 1.1. does not support quantitative controlling characters yet.

In all formats and applications controlling characters must be categorical (i. e., quantitative counts cannot be controlling).

53. Character dependency definitions are important information items for data entry, character management, and analysis purposes.
54. A general form of value-dependency may predict values in another character for some (but not all) values of a controlling character. It may be desirable to implement this, but it has not been pursued in current models.
55. A special form of value-dependency is that some values predict the applicability of another character (character applicability rules). This is highly desirable and implemented in several descriptive models.
56. It is desirable that the controlling character may be of categorical or quantitative type. Current models only implement categorical controlling characters.
57. Because of character evolution issues (adding states to existing characters) and to improve the clarity of expression, both positive and negative character applicability rules (“Applicable-if”, “Inapplicable-if”) are desirable.
58. Combinations of applicable-if and inapplicable-if rules within a data set are desirable. However, any combination of controlling and controlled character may be covered only by one form of the rule.
59. Support for the evaluation of cascading character applicability rules is desirable. This may be expressed in specialized graph structures in the information model, but may also be supported only during evaluation of rules.

(Note: a related topic of method dependency will be discussed later on p. 175.)

4.9. Raw data and data aggregation

Introduction

Before discussing various models to record structured descriptions, a final fundamental problem that is independent of the specific model is addressed: The level of data aggregation and the structure of the set of objects that is the subject of the description.

Data aggregation is used here in the sense that it is the process of representing multiple individual data points in a transformed, more compact form. Typical aggregation methods are statistical measures (mean, variance) for quantitative data and frequency histograms for categories. Depending on perspective, these processes may also be viewed as “summarizing” (Macfarlane

1993a, White 1994, Hagedorn 1999a), “agglomerating” (Maxted & al. 1993), “collating”, “consolidating”, “aggregating” (three terms used frequently during SDD discussions, the last also in Diederich & al. 1997), or “abstracting”, “generalizing”, or “amalgamating” (all three by Pullan & al. 2005) data. The term “aggregation” was selected in SDD discussions as preferable. Pullan & al. (2005) combine the concept of data aggregation methods (where the number of data points changes) with the concept of transformations from quantitative to categorical data (where the number of data points remains constant, see “mappings” p. 66) as “levels of abstraction in the description-building process”.

A major reason for data aggregation is the transformation from specimen to taxon descriptions. Lebbe & Vignes (1998) speak of a “double nature of a taxon as a concept and a set of instances”, that makes it possible to have contradictory statements (e. g., “present or absent”) in a taxon description. However, the step from the individual to the taxon is only one of many aggregation steps that may occur in descriptions. For example, even though a simple measurement such as the length or shape of a leaf can only be obtained by measuring a single leaf, values for leaf length or shape may be reported on the following data aggregation levels:

1. repeated measurements of a single leaf (i. e. concrete measurements);
2. multiple basal leaves of a single individual;
3. all basal and stem leaves of a single individual combined;
4. multiple individuals in a single population according to some scope (male individuals, juvenile individuals, etc.);
6. all individuals from a single population (or infraspecific taxon);
7. all sampled populations of a species (or infraspecific taxon);
8. all taxa classified in a higher taxon (creating, e. g., a genus description).

Clearly many more combinations of these aggregation criteria are possible (all male juveniles of a species in Germany...); a fixed sequence of aggregation level seems therefore not desirable. The problem of multiple instances of a part in individuals and aggregation at populations is also noted by Diederich & al. (1997) and Pullan & al. (2005). Ideally an information model for descriptive data should be flexible enough to allow entering both repeated sample data and aggregated data at multiple such levels.

Another distinction that is often made is that between “raw”, unprocessed data and “synthetic” or “aggregated” data. Storing raw data is desirable:

- in general, to archive them as a reference for data generated within a study;
- in cases where the processing can be automated, e. g., in generating descriptive statistics of repeated measurements.

The example of leaf measurements shows, however, that this distinction has a very complex relationship with the conventional taxonomic set structure of individual, lowest-ranking taxon, and hierarchical taxa. A slight simplification may be made by ignoring the relatively rare case of multiple measurements on a single object as part of the methodology (e. g., to reduce measurement error). Some instruments may even do this internally, but report only a single measurement. Nevertheless, even if both level 1 and 2 above are considered “raw data”, whether a relation between “raw data” and the individual organism exists or not depends on the multiplicity of the object part. In some cases this may be deduced from the fundamental organization of organisms in a larger taxon group, but in many cases this is species-specific (for example, some plant species have a single stem, others multiple stems).

Similarly, the aggregation levels 3 to 4 indicate that descriptions often have a scope that does not directly match the taxonomic hierarchical classification. A geographical scope is not necessarily congruent with an actual or potential infraspecific taxon. Differences between geographically scoped descriptions may be due to genetic mechanisms or may be based on, e. g., a systematic influence of climate on the phenotype.

Several requirements can already be formulated:

60. Structured descriptive information models must provide methods to describe properties of sets of objects.
61. Both repeated sample data, and the results of statistical and non-statistical data aggregation methods should be supported.
62. Aggregation methods are required for descriptions of classes. In biology, sets of individuals form taxa, sets of taxa form higher taxa. No difference could be detected between aggregating from individual to lowest level class and lower level class to higher level class.
63. Aggregation methods are also required for descriptions of individuals, containing either multiple parts or changing over time (discussed in detail further down, p. 93).
64. The difference between a descriptive information model for individuals (e. g., in a specimen database) and taxonomic classes (e. g., in a taxonomic database) with respect to aggregation methods is negligible.
65. Classes or sets of objects may be defined by non-taxonomic means, e. g., through a geographic scope (see also “Secondary classification resulting in description scopes”, p. 215).
66. A fixed sequence of aggregation levels (such as “object part, individual, taxon”) is covering only a subset of aggregation cases and should not be part of the information model.

The topic of raw data and aggregation is also presented in a use case diagram (Fig. 178, p. 297).

To simplify the following discussion, most examples will be discussed assuming a taxon-specific object/property model (Fig. 13, p. 42), but they are equally applicable to a generalized model (e. g., Fig. 14, p. 43).

Standard aggregation methods

Descriptive data for sets of objects can simply be stored as a big collection of repeated values. However, such collections are not very digestible or intelligible for humans. Other methods that reduce the amount of information that has to be processed are preferable.

The most frequently used data aggregation methods are based on univariate descriptive summary statistics. Depending on the measurement scale (p. 49), the set of univariate statistical measures may be highly limited or very large. For all categorical data (nominal or ordinal) sample size, a mode, and a frequency distribution can be calculated. Furthermore, in the case of ordinal data the total range (minimum to maximum) and median are frequently used. A much wider range of summary statistics is applicable to continuously varying quantitative numeric data (e. g., mean, variance, range measures, or confidence intervals).

For some statistical measures such as confidence intervals, the number of measures is potentially infinite, because a confidence measure may be defined for any desired confidence probability value. A similar problem occurs with percentiles. In practice, the list of commonly used measures is limited (e. g., 90%, 95%, 99% confidence intervals, 60%, 80%, 90%, 95% percentiles). One solution is therefore to consider each of these frequently used confidence measures a separate category. A more general solution is to define a basic vocabulary of confidence-measure classes that is supplemented by a quantitative parameter (taking values like 0.9, 0.95, 0.99, etc.). The latter solution is used in SDD.

Integer data may be summarized either using the methods intended for continuous data or using a frequency distribution. The latter method is especially appropriate if the distribution is unusual (e. g., spores that have only 3, 5, or 7 septa, or a leaf with either 15 or 17 leaflets).

In biological descriptive data one of the most customary aggregation methods for categorical or integer data is a special form of “frequency distribution” where the frequency information itself is ignored or not available and simply the set of values with frequency > 0 is reported. This method is used when saying “has linear, lanceolate, or ovate leaves”. In statistical data analysis this aggregation method is sometimes called “value list”, but this term may also be used for a

total list of original value (i. e., values occurring repeatedly). It is therefore proposed to call the aggregation method “*distinct value list*”. It is applicable to categorical and discrete quantitative data (as in the example “3, 5, or 7 septa” above).

Where frequency information is given, it is often simplified and represented only by verbal estimates (e. g., “rarely”). It is possible to define or estimate frequency ranges corresponding to these verbal frequency statements, see “Frequency modifiers”, p. 206.

In addition to using defined statistical measures, another customary aggregation method is to record human estimates of “typical ranges”. This is not truly an aggregation method for data that have already been measured, but a replacement for it. An estimate of “typical range” may be achieved by visually comparing objects and measuring samples that appear relatively small and large while ignoring unusual or extreme cases. Only the “aggregated data” will then be recorded. The reduced precision and potential problems when analyzing estimated data are often accepted because of the increase in work efficiency.

Some statistical measures are related and may even be substitutable for certain purposes. Mean, mode, and median are all a kind of central value; standard deviation, variance (both with $df = n-1$ and $df = n$), and standard error are a variance measure, the first four referring to values, the last to a mean (Fig. 32). Such relations may be expressed in a class hierarchy or through ontological kind-of relations. This is desirable during data analysis, but may also be desirable when entering data. Thus in addition to mean, median, mode, and human mean estimate, it may be desirable to label a value directly as “central measure”.

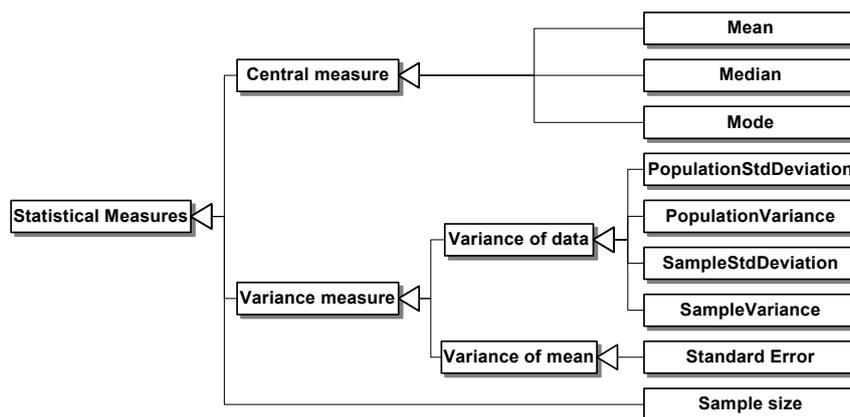


Figure 32. UML class diagram showing a selection from a generalization hierarchy of statistical measures.

The approximate substitutability of statistical measures is especially relevant for identification and report-generation purposes. To compare a measure of an object to be identified with descriptive data in a database, most kinds of range measures (confidence interval, appropriate percentiles, mean \pm std. dev., and even estimated ranges) are useful. The different measures have to be distinguished only for more exact analyses. However, as a design requirement for descriptive data systems, knowledge about the similarity of statistical measures does not necessarily have to be a part of the terminology; it may equally well be incorporated into analysis software.

DiversityDescriptions was probably the first information model for descriptive data to introduce a fine-grained and extensible model for statistical measures (p. 356). This included a concept for “undefined ranges”, but not for generalized central measures, or human mean and range estimates. SDD (p. 20) strongly improves on this, containing an enumeration of 38 statistical methods (“UnivarStatMeasureEnum” and “UnivarStatMeasureWithParamEnum”), informing for each of these methods whether values are dimensionless or not, and categorizing them into reporting and method classes (Table 18).

The question how data referring to statistical measures may be integrated into the data storage model is discussed later on p. 110.

Table 18. Classifications of univariate statistical methods used in SDD. These provide semantic information about statistical methods intended to enable the creation of generalized software.

Enumeration	Category	Description
Reporting classes	Central Measure	Any kind of central measure, like mean, mode, median, etc.
	Lower Range	The lower value of any kind of range measure, like ‘mean minus standard dev.’, confidence interval, percentiles, etc.
	Upper Range	The upper value of any kind of range measure, like ‘mean + standard dev.’, confidence interval, percentiles, etc.
	Lower Extreme	The absolute minimum value (a category of its own).
	Upper Extreme	The absolute maximum value (a category of its own).
	Variance Measure	Any kind of variance measure, like standard deviation, variance, etc.
	Sample size	Sample size is a category of its own.
Method classes	Other	Any other kind of statistical measure.
	Statistical estimate	Measures estimated by statistical methods. Examples: Sample mean, minimum, confidence interval, standard deviation (n-1 degrees of freedom).
	Statistical parameter	Values calculated by statistical methods that are exact in relation to the population under study (statistical estimates are exact in relation to the sample, but estimates in relation to the population under study). Examples: Sample size, standard deviation (n degrees of freedom).
	Observer estimate	Values estimated by humans without using mathematical/statistical methods.
	Unknown method	Values obtained by an unknown method. This may be a statistical method or a human observer estimate. Many legacy data sets and data published in print fall into this category.

Reporting classes are similar to the five basic measurement classes supported by DELTA. Most applications that report information for human use can rely on these reporting classes in their decision how to present the data, increasing compatibility with future SDD/UBIF versions. Implementers may, however, choose different methods of handling the statistical measures.

Method classes inform about very general quality properties of measures.

67. Support for range of univariate statistical measures is required. The set of applicable measures depends on data type and measurement scale.
68. Most univariate statistical measures report only a single value.
69. Some univariate statistical measures such as percentiles or confidence interval limits are best represented as a combination of a result value and a method parameter.
70. In addition to exact measures, support for human estimates such as “typical range” is required.
71. To support legacy data such as DELTA, support for undefined measures is desirable (e. g., in DELTA a value is known to be a central value, but not whether it is a single measurement, a mean, median, or mode).
72. Some standard descriptive statistics report a collection of data items rather than a single value for a statistical measure. Frequency distributions and distinct value lists must be supported. A distinct value list is a frequency distribution with unknown frequency.
73. It is desirable to support two forms of frequency distributions: with frequency values and with frequency categories (i. e. frequency modifiers).

Inappropriate aggregation results

Recording a temperate tree in winter as “leaves absent” and in summer as “leaves present” may well be considered appropriate. However, aggregating such data would lead to a taxon descriptions like “leaves present or absent” which might be considered “silly” for a temperate tree. A

very similar situation in animals may be considered less silly: The Arctic hare (*Lepus timidus*) having “white or brown fur” seems considerably more appropriate. And is a description of a butterfly as “with or without wings” (i. e., as caterpillar) appropriate or not?

Again, considerable information about the expected background knowledge of the data consumer seems to be required. Repeating general knowledge leads to descriptions that appear “silly”.

The problem also has some serious implications for rules for data recording, analysis, and identification. When recording data for individuals, care must be taken about the expected scoping of a description. A character like “leaves (presence)” may be expected to be validly coded on the taxon level, i. e., it is negative only if the organism never produces leaves. This may be desirable for analytical purposes studying phylogenetic or genetic processes. However, when recording data on the individual level, data are both constrained by individual variability and development over time. If the individual winter tree is identified by someone not knowing that it has leaves in summer, the answer “leaves absent” would lead to a failure of identification (p. 310), if the taxon description of the tree is limited to “leaves present”.

Further studies are needed to clarify how these problems can be resolved. The current standard solution seems to be that in most characters *a-priori* knowledge of temporal variability or the entire life-cycle of an organism is expected (e. g., presence of leaves, flowers, or fruits) and therefore the taxon-level knowledge is applied to guide the user both during data recording and identification. Where the knowledge is lacking, one may choose to live with “silly” descriptions. This solution is clearly not satisfactory, especially since the expertise of data recorders and users of identification tools will typically be quite different. Several other solutions may be envisaged:

- Use separate characters for winter and summer.
- Use temporal modifiers (p. 204) for winter and summer. The aggregated description could then read as “leaves present (in summer) or absent (in winter)” and “brown in the summer, white in the winter; ears tipped with black all year round”.
- Create scoped descriptions for different life-cycle or seasonal stages (see “Secondary classification resulting in description scopes”, p. 215).
- Attach metadata to characters, guiding the aggregation process as to whether temporal occurrence of a character does occur (this is related to “Dependencies on circumstances of identification”, p. 175).

The examples show, that data synthesis to obtain taxon descriptions may be more complex than using aggregation methods

74. Character metadata informing about the expected scope or recording level of a character may be required. For genetic traits the scope is summarized over the entire life-cycle, for diagnostic purposes individual points in time may be more appropriate.

75. In addition, or perhaps alternatively, character metadata informing about dependency of observation on circumstances or temporal development (and therefore the likeliness that data recorded on individuals represent the entire developmental cycle) may be required.

Aggregating aggregated data

Data aggregation frequently occurs on sets where the members already contain aggregated data. This occurs, e. g., when generating a genus description from species descriptions, or when generating a species description from specimen descriptions for a property that occurs multiple times in each individual.

Frequency distributions, distinct value lists, and some statistical measures (such as minimum, maximum, or sample size) can easily be further aggregated. For other statistical measures such as mean this may be possible if sample size is recorded, but most statistical measures do not easily

support further aggregation. The calculation of exact statistical measure in the latter case requires access to the original sample data.

In certain cases, further options may exist. For example, although defined range measures like confidence intervals or percentiles cannot be aggregated as such, it may be desirable to degenerate them to a lower quality measure (e. g., “unknown range method”, or (human) “range estimate”), which in turn can be aggregated (Table 19).

A general alternative is to report the list of individual member measures, for example: “standard deviation 3.2, 4.6, or 1.9”. In many cases it may be more desirable to express the aggregation of multiple ranges as a set of ranges rather than combining them to a new range. This occurs when the union of member ranges of the set is non-continuous. For example, when aggregating the individual ranges “3-5”, “4-7”, “20-25” the set expression “3-7” or “20-25” is more informative than the combined range “3-25”. It seems desirable, however, to use some heuristic method to decide whether it is more appropriate to use a combined range or not. For example, when aggregating the individual ranges “3-5”, “5.5-10”, “11-25” the combined range “3-25” is probably preferable.

Table 19. Aggregating statistical measures.

Entity	Minimum	Lower range	Central value (Mean/Median)	Upper range	Maximum
Entity 1	2	3 (conf. interval)	5.5 (mean)	7 (conf. interval)	
Entity 2	1.5	2 (human estim.)	4.5 (median)	7 (human estim.)	9
Entity 3	5	6 (unknown)		8 (unknown)	
Summary:	1.5 (min.)	2 (undefined)	5 (undefined central value)	8 (undefined)	9 (max.)

and with an additional entity 4 added to entity 1 to 3, this would look like

Entity 4		1	8	12	
Summary:	⁽¹⁾	1 (undefined)	6 (undefined central value)	12 (undefined)	⁽¹⁾

⁽¹⁾ In the summary of entity 1-4, the minimum and maximum measures are omitted, because they are already included in the normal range of at least one entity.

76. Methods to aggregate aggregated data are highly desirable and can be devised in certain cases. Supporting all necessary data for this in the information model is highly desirable.
77. Where statistical measures cannot be aggregated further, either access to sample data is required, or the member values may be listed, or in certain cases measures may be degenerated to lower-quality measures that in turn can be aggregated.
78. In the case of ranges, if the union of ranges has considerable gaps, an aggregation as a set of ranges is more desirable than combining the ranges into a single range.

Data recording levels (sample data)

Ideally, a descriptive information model should support both collections of original, repeated data and descriptive summary statistics. The original data are often very valuable for later reanalysis and support for preserving them is a design requirement for SDD. Furthermore, supporting them improves the workflow, because the original data recording and the automatic calculation of descriptive statistics can be performed by a single tool. The current workflow is often inefficient. Data may be recorded on paper, statistics are calculated using a pocket calculator or spreadsheet, and the results are transferred to a database.

Although descriptive statistics may be calculated on the fly for report-generation purposes, not supporting them in the descriptive information model means that any knowledge where original data are no longer available (such as most scientific publications), would be excluded. Thus, descriptive statistics (mean, ranges) should be supported both as results of calculation referring back to original sample data, and as primary data itself. Some kind of “status flag” may be desir-

able to distinguish between calculated statistics and statistics that are primary information in the database.

In the information model for descriptive data, one might be tempted to limit the applicability of original repeated sample data to object descriptions (specimens), since direct observation of values for class descriptions (taxa) is in principle not possible. However, this would create major problems when attempting to record data from class descriptions. Neither repeated values for a single property (e. g., “5, 7 or 9 bundle scars”), nor values for different properties are necessarily actually or even potentially obtained from a single specimen. Rather than representing a class description by a single “abstract specimen”, this would strictly require to introduce one artificial “dummy” specimen for each value in a class description for which no specimen information is available. Similarly, in certain cases it may not be known whether the value reported in a class description is an average from multiple specimens, or the only measurement for the only studied specimen of the taxon. For these reasons, the SDD models repeated sample data both for class and object descriptions.

Another aspect of recording aggregated data is that in many cases the statistical methods are not properly reported in the publication. In taxonomic publications the method is often not “statistical” at all: a value range may simply reflect the author's estimate of what is considered usual and what is considered rare. One solution for this problem is to introduce non-statistical “human-observer estimates” in parallel to the statistical measures (chosen by SDD, p. 20).

Finally, multiple separate samples may have been taken within a class of objects. For many purposes (checking outlier values detected during analysis, analyzing possible reasons why results in an entire sample differ from other samples) it is desirable to preserve the original sample structure together with same metadata about the sampling circumstances.

One advantage of preserving original sample data together with sample structure is that appropriate summary statistics can be calculated for any level of aggregation desired (e. g., subspecies, species, genus description). Recording only the summary statistics limits further aggregation. Only minimum, maximum, sample size, and (provided sample size is known) mean can be aggregated, but no variance measures, confidence intervals, or percentiles. In these cases the only available reporting method is to report the individual summary statistics multiple times for each sample.

79. Structures for original sample data (sets of values observed under the same conditions) are required on individual object descriptions as well as on class descriptions.
80. Sample data may be associated with metadata (conditions, time, place, etc.).
81. The structure of multiple samples (each containing multiple observations) should be preserved.

Linked observations

In many cases multiple properties of an object are recorded together. When recording the length, width, shape, and number of septa of the spores in a fungal specimen it is possible to sample a set of spores for width, another for length, another for shape or septation. In many cases, however, it is desirable to record all four properties for each individual spore in the sample. Such linked (or “correlated”) observations increase the options for analysis. For example, it is common to analyze the size and shape of aseptate, uniseptate, and multiseptate spores separately. Furthermore, length-width-ratio calculated as the average of individual length/width ratio values is the most correct information, and is different from the ratio of average length divided by average width. Similarly, other correlations can only be calculated if both the original data are present and the linking status is documented.

Table 20. Examples of repeated quantitative and categorical measurements for independently measured data for a single object (or specimen). The leaf observations are linked, the siliqua observations not.

Sample	Object /Unit	basal leaves				siliquae			
		leaf blade			– petiole –	– young –	mature		
		length	width	shape	length	Length	length	width	seed #
1	1	8.0 cm	4.0 cm	ovate	5 mm				
	2	8.0 cm	4.5 cm	ovate	5 mm				
	3	9.0 cm	3.5 cm	lanceolate	6 mm				
				
	10	7.5 cm	4.0 cm	ovate	4 mm				
2	11					5 mm			
	12					7 mm			
	13					5 mm			
			
	20					4 mm			
3	21						20 mm	2.5 mm	14
	22						25 mm	3.0 mm	16
	23						21 mm	3.0 mm	15

	30						22 mm	3.0 mm	16
Summary:									
	Mean	8.0 cm	4.0 cm	(n/a)	5 mm	5 mm	22 mm	3.0 mm	15.6
	Min	7.5 cm	3.5 cm	(n/a)	4 mm	2 mm	19 mm	2.5 mm	14
	Max	9.0 cm	5.0 cm	(n/a)	6 mm	7 mm	26 mm	3.5 mm	16

	Frequ.-distrib./ Distinct value list	(n/a)	(n/a)	usually ovate, rarely lanceolate	(n/a)	(n/a)	(n/a)	(n/a)	14, 15, 16

Note: Characters measured from the same object (a single leaf, siliqua) are recorded together in a row. All measurements are synthesized into a summary description using statistical parameters for quantitative measurements and frequency distribution/distinct value list method for categorical or integer data. The data are simulated.

A descriptive information model should be able to express both linked and non-linked repeated measurements. This requires the introduction of one container concept for the sample, and another concept for the linked observations. The latter may be called “sampling unit” (shown as rows in Table 20).

Such an extension is necessary both for character models and object composition models explicitly modeling object parts (discussed later in detail under “Character decomposition models”, p. 116). Although a “container” for repeated parts already exists in the latter type of the models, the part that is repeated (e. g., “basal leaves”) may be on a higher level than the parts for which linked property observations are recorded (blade, petiole).

The problem of linked observations is not limited to repeated measurements on object or parts. If a genus contains two species, one with green ellipsoid spores, the other with brown round spores, the genus description may be aggregated to “spores green or brown, round or ellipsoid”. However, this description ignores the linking of values and as a result a species with green round spores would fit the description. The problem is discussed further below under “Boolean operators between states of categorical characters” (p. 95).

- | |
|--|
| <p>82. Repeated sample data should preserve the sampling context and linking of observations (multiple properties observed on the same part or individual).</p> <p>83. Aggregation of data should be able to preserve information on linked observations.</p> <p>84. Aggregation of data should be able to express the lack of information on linked observations.</p> |
|--|

Special aggregation cases

Many data types do not fit into the data aggregation and analysis framework for categorical or quantitative data. Examples are certain molecular data such as protein/nucleic sequences or patterns (RFLP, AFLP, RAPD, etc.), raw spectrographic or chromatographic data, or repeatedly recorded media data like images, bird songs, etc.

Similar to the case of categorical or quantitative data discussed in the chapter introduction (p. 84), a – perhaps intuitive – classification as “raw data” is not appropriate. Although processing, standardization, and aggregation methods will often be performed on these data types before they can be used to compare or identify organisms, this process may have several steps, each of which is based on data that are “raw”. For example, for DNA sequence data based on a four-color sequencer (e. g., ABI™) at least the following steps will be distinguished:

- Two-dimensional density data images for each of the four fluorescent colors based on the different dyes with which the ddATP (dideoxyATP), ddCTP, ddGTP, and ddTTP had been marked
- A single four-color density data image, combining the single-colors after correcting them for their respective differences in retention time
- The four-color image with vector information (detection of lanes, detection of bands within these lanes, annotation which lane is A, C, G, or T) added
- A four-color one-dimensional density chromatogram for each sequence fragment (unverified, single-strand-reading DNA sequence between two primers, e. g., “SCF” or “ABI”-files)
- Multiple sequence fragments aggregated into a “contig”-sequence, where each part is present in both reading directions and differences are corrected/resolved, together with the respective chromatograms (e. g., Sequencher™ SPF-files)
- The extracted corrected consensus sequence as a string of AGCT letters.

Perhaps a more relevant distinction is, whether a data type is intended for machine-based aggregation and analytical processing at all. Especially where data are stored in both a more raw and more processed format, it is likely that the first is primarily intended for human consumption, possibly as a voucher for the processed information. An example may be a case where in addition to data on presence of concentration of a chemical compound, also a digitized mass-spectrographic printout is stored.

The distinction cannot be made based on the data type itself. Whether a picture is intended as an information voucher, or whether it is intended for algorithmic image comparison methods is not self-evident, and may, in fact, change over time. Therefore, whether metadata on “original intent” shall be recorded or not, cannot easily be decided. It may be desirable to do so as hints for consumers of a data set, but is not an absolute necessity.

In general for each complex data type (p. 59) processors must be informed about available aggregation and analysis methods and the conditions under which they are appropriate to use. It is desirable to model this as generally as possible, since it is likely that new aggregation and analysis methods (like algorithmic image comparison) will become available over time. Examples of specialized aggregation methods are:

- Multiple nucleotide sequences may be aligned and analyzed using phylogenetic or distance methods, or aggregated using “consensus sequences”, where differences are expressed using ambiguity symbols (compare p. 57). Consensus sequences are a general method, which may be used for various purposes like phylogenetic, functional, or structural analysis. However, the usefulness of “consensus sequences” is limited to sequences that are aligned and relatively similar. Information content is quickly lost when aggregating distantly related sequences using consensus sequences (many base positions becoming the ambiguity symbol ‘N’, indicating that any of ACGT may occur).

- It may be possible to process chromatographic data in similar ways (after normalizing for variation in retention time, and probably after peak identification and peak area integration, since isolated peaks are easier to integrate than overlapping peaks).

For media recordings no true aggregation method is currently known to the author. A common method to deal with a large sample of media recordings (images, video, or audio) is to select one or few “typical” items that are considered “representative”. Care must be taken to express the variance as well as the median (e. g., in birds with markedly different song dialects).

85. “Raw data” is no absolute category that has special properties. Data processing often occurs in multiple steps, each of which may be called to be based on data that are “raw” relative to the results.
86. Some data may not be intended for machine-processing at all, but rather provided as “information vouchers”. It may be desirable to support this distinction through metadata, but it may also be possible to let a processor assume any data for which it cannot find aggregation or analysis methods, to belong to this category.
87. Special data aggregation methods may be available for certain types of descriptive data, either truly aggregating data into a new form, or by selecting “representative” data items from the full set. The information model should provide both for linking base data and derived aggregated data, and for selecting some from repeated data as being representative (especially for media data).
88. The support for data aggregation methods should be extensible, providing for future methods. It is currently unclear how data structures that might be necessary for specialized aggregation methods can be anticipated in the information model.

Aggregation within individuals

Often several values exist for a single property variable in a single individual, leading to data aggregation within individuals. The three major scenarios are:

- biological objects change over time (ontogenetic development),
- patterns, gradients, or intermediate values result in the coexistence of values in a single object instance (Fig. 33 a), and
- the object occurs multiple times in the individual (Fig. 33 b-d).

All three reasons of variation are a source of problems in structured descriptions (see also the sections “Change of object concepts through temporal development”, p. 162, “Pattern versus composition”, p. 165, and “Spatial gradients”, p. 150).

Importantly, multiple occurrences of object parts in an individual may be structured into assemblage levels. For example, an individual plant may have multiple shoots, each with a single inflorescence, each with multiple flowers, and each with multiple petals. These levels are difficult to generalize and many levels are specific to a species or higher taxon. Examples: Species having always only a single shoot; species in the plant family Apiaceae having always one or two levels of umbels (simple or compound umbel).

In structured descriptions one seeks a basic structure that enables comparability across such differences. One approach is to create a potentially complete model, enabling an unlimited number of levels on which objects may be assembled and then to provide methods to make these levels comparable. The standard aggregation methods do support hierarchical aggregation of multiple levels (at least when original data are available, aggregating descriptive statistics is more limited). The problem is, however, that it is difficult to establish a system that provides comparability of the various levels, their hierarchy, and taxonomic dependency. Solving this problem requires considerable investment both in modeling and implementing the generalized model, and devising a terminology that either in advance knows about all of compositional (= structural) diversity, or can evolve during data recording. Few models have attempted to model the complexity

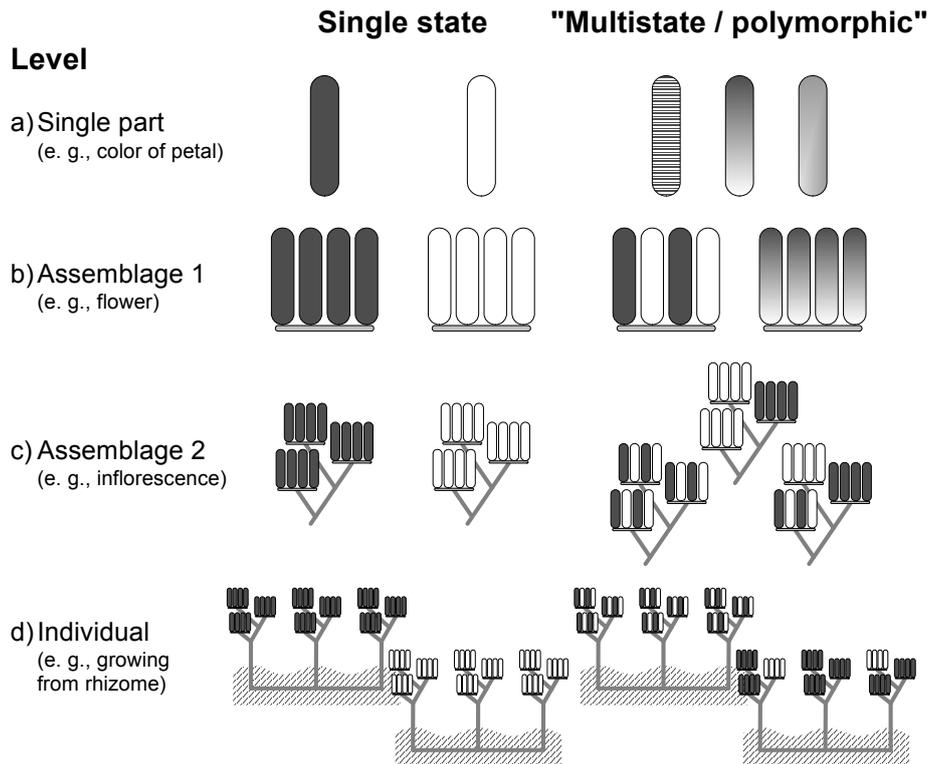


Figure 33. Examples of multistate (= polymorphic) situations in a single individual at a single point in time. A range of property values may exist already in a single object part, e. g., due to a pattern, a gradient, or intermediate values. More frequently, polymorphic data are due to objects existing multiple times; this may occur at multiple assemblage levels within an individual. Only a subset of all possible combinations is shown in the diagram.

of part compositions (see “Object composition”, p. 131, and “Character decomposition models”, p. 116 for detailed information).

Most existing information models for descriptive data attempt to reduce the complexity by ignoring some levels and focusing on those that are most general, most frequently the cause of polymorphism, and are sufficiently comparable.

The level of the individual is probably most easily general in biology. An individual may be defined as the coherent part of tissues of the same genotype. For example, each monozygotic (genetically identical) twin is an individual, but all shoots of a plant growing from the same root belong to one individual. Whereas in most animal groups the recognition of individuals is not contentious, in some groups of fungi, algae, non-vascular and vascular plants, recognition of true individuals may require expensive and time-consuming genetic methods (compare Fig. 33 d, plant shoots growing from a rhizome). However, it is rare that a polymorphism occurs between shoots of a single genetic individual. In practice these variations are most likely attributable to phenotypic plasticity or systematic variation in response to the environment, which occur between individuals on the population level. For practical purposes, a shoot or a fruiting body of a fungus is therefore usually considered to be equivalent to an individual.

Another problem is that occasionally important descriptive information depends on levels above the individual. Examples are colonial corals (Anthozoa, Cnidaria), or lichens (where the fungus depends on a mutualistic symbiosis with algae from one or several species).

89. Aggregation methods are required within individual specimens where observations are repeated over time or due to object parts occurring multiple times. Generalizing from specimen data to taxon data is only a special case of these general aggregation methods.

Boolean operators between states of categorical characters

In principle, descriptive statements may involve Boolean operators such as ‘*and*’, ‘*or*’, ‘*xor*’, and ‘*not*’ (where ‘*A or B*’ is true for ‘*A*’, ‘*B*’, or ‘*A and B*’, and ‘*A xor B*’ is true for ‘*A*’ and ‘*B*’, but not ‘*A and B*’). Using a syntax modeled after a combination of SDD and MathML (Carlisle & al. 2003), one may write, e. g.,

<p>for “petal color: red or orange or yellow”:</p> <pre><char id="petal color"> <apply> <or/> <state id="red"/> <state id="orange"/> <state id="yellow"/> </apply> </char></pre>	<p>and for “petal color: red and (orange or yellow)”:</p> <pre><char id="petal color"> <apply> <and/> <state id="red"/> <apply> <or/> <state id="orange"/> <state id="yellow"/> </apply> </apply> </char></pre>
--	---

Table 21. Example data to test ‘*and*’ and ‘*or*’ statements on different aggregation levels (compare text).

Aggregation level:				Values:	
Species	Individual	Flower	Single petal	Color	Shape
S1	I1	F1	P01	red	round
S1	I1	F1	P01	orange	round
S1	I1	F1	P02	red	round
S1	I1	F1	P03	orange	elliptical
S1	I1	F2	P04	red	round
S1	I1	F2	P05	red	round
S1	I1	F2	P06	red	round
S1	I2	F3	P07	yellow	elliptical
S1	I2	F3	P08	orange	round
S1	I2	F3	P09	orange	round
S1	I2	F3	P10	orange	round
S1	I2	F4	P11	orange	round
S1	I2	F4	P12	orange	round

A major problem, however, is that the semantics of the Boolean operators ‘*and*’ and ‘*or*’ between prepositions is ill-defined if multiple aggregation levels are present and the level to which a statement should apply is not defined. Since the set of states for species S1 from Table 21 contains all states, descriptions may be:

- “Species S1 has red, or orange, or yellow petals”, or
- “Species S1 has red, and orange, and yellow petals”.

In practice, however, the second expression is misleading. Since a species is a class of individuals that cannot be directly observed in nature, most biologists would assume that the implied meaning is:

- “Individuals of species S1 have red, and orange, and yellow petals”,
- which is not true. A true statement would be:

- “Individuals of species S1 have red, and orange, or orange and yellow petals”.

The closely related statement:

- “Petals of individuals of species S1 are red, and orange, or orange and yellow”

could be interpreted that rather than that the individual has at least some petals that satisfy this condition, each petal must satisfy this condition. The corresponding true statement would be:

- “Some petals in species S1 are red and orange, others only red, or only orange, or only yellow”.

Similar care is necessary when using:

- “Flowers of species S1 have red and orange, or red, or orange, or yellow petals”.

This statement is true, but it remains ambiguous whether the variation is within flowers or between flowers. A more exact statement might be:

- “Each flower of species S1 has either (red and orange) petals and red petals and orange petals, or only red petals, or yellow petals and orange petals or only orange petals.”

Such statements are normally not made in natural language descriptions and their meaning is very difficult to decode. As a consequence, much available data is ambiguous. Designers of information models that attempt to model the exact situation would either need to decide that they can only be used for new data, or they would need to implement methods to deal with the ambiguity that is present in current data. The Prometheus model follows the latter path, and, requiring to explicitly build and clone object parts, describes them individually. This allows a much greater expressivity with regard to Boolean operators.

Existing DELTA-based software attempts to avoid the complexity of this problem. It allows a combination of states within a single character (i. e., a property of the entire object or a part of it) with either ‘*or*’ or ‘*and*’, and always assumes an ‘*and*’-relation between characters. However, for comparison and data-retrieval purposes during identification, the ‘*and*’ operator is treated as identical to ‘*or*’. The distinction is used, however, when natural language wordings for human users are to be generated. By this method, DELTA avoids the need to specify the intended aggregation level exactly, and allows both statements that are intended for individuals and statements intended for the species level. The implied level is usually supplied by the background knowledge of the human reading the natural language description. The statement:

- “Petals yellow or orange”

may mean that individual petals are both yellow and orange, but also that each flower has yellow or orange petals, or some flowers in each individual, or that some populations have yellow, others orange flowers. In contrast, when reading:

- “Petals yellow and orange”

probably a level of individual organism, individual inflorescence, or individual flower is implied. However, based on the use of the ‘*and*’ not even the individual level can always be assumed.

Whether reading:

- “Occurring in Germany *or* France”, versus
- “Occurring in Germany *and* France”

humans implicitly provide the knowledge that this statement indeed is meant to be expressed on the species level, since it is unlikely that the species only contains individuals growing on the exact border of the two countries. Most humans would consider the two statements to express the same semantics. Some people would consider the *or*-ed statement more natural, visualizing a species description containing several statements. Most biologists, however, would consider it more natural to use the ‘*and*’-wording, visualizing *multiple* individuals (or perhaps dots on a map). The importance of implied semantics becomes even clearer when contrasting the distribution example above with (a) a property that can possibly occur only a single time in an individual, such as size:

- “size 10 cm *or* 11 cm”, versus
- “size 10 cm *and* 11 cm”,

and (b) a property where it is ambiguous whether it refers to multiple parts of the individual or to multiple individuals in a species:

- “roots branched *or* not”, versus
- “roots branched *and* not branched”,

and finally (c) a property where it is intuitive that it refers to different parts of the individual:

- “leaves opposite *or* alternate”, versus
- “leaves opposite *and* alternate”.

Clearly the preferred Boolean operator in the case of distributions (“occurring in Germany *and* France”) is rejected as nonsense in the size example (“size 10 cm *and* 11 cm”). The choice of operator changes the perspective in the root example (“roots branched *and* not branched” probably implies that the plant has several root systems and that the term “root” refers to the major branches rather than to the entire root system). Finally, in the leaf example it may be accepted and considered to imply that in each individual plant both opposite and alternate leaves occur.

One result of this is that the semantics of Boolean operators in legacy data (e. g., printed natural language descriptions or printed keys) is often ambiguous and requires interpretation. The information model should therefore ideally support lack of semantics as well as interpreted or original semantics.

In **SDD** (p. 20) the model descriptor for states in a single categorical character data element may be changed from the default (“OrSet”) to other values, compare Table 22.

Table 22. Values of the StateCollectionModelEnum in SDD (used in SummaryData/Categorical/@statemodel)

Value	Label	Description or Examples
OrSet	Unordered set of states, combined with 'or'	Multiple states scored for a character in a description form a set. The order of states has no special meaning and may be changed. In natural language output the states should be combined with 'or' to express that in individual objects (that belong to the class that is being described), the states may occur together or alone.
OrSeq	Ordered sequence of states, combined with 'or'	Multiple states scored for a character in a description form a sequence, i. e., the state order carries some semantics and should be preserved in output. The precise semantics of the sequence is not explicitly defined, but assumed to be intelligible to human consumers; presumably relating to concepts of relevance or importance. In natural language output the states should be combined with 'or' to express that in individual objects (that belong to the class that is being described), the states may occur together or alone.
AndSet	Unordered set of states, combined with 'and'	Multiple states scored for a character in a description form a set. The order of states has no special meaning and may be changed. In natural language output the states should be combined with 'and' to express that (in any individual object that belong to the class that is being described) the states will always occur together. Example: two colors that occur together in a pattern.
AndSeq	Ordered sequence of states, combined with 'and'	Multiple states scored for a character in a description form a sequence, i. e., the state order carries some semantics and should be preserved in output. The sequence semantics is not explicitly defined, but intelligible to human consumers and presumably relates to some concept of relevance or importance. In natural language output the states should be combined with 'and' to express that (in any individual object that belong to the class that is being described) the states will always occur together. Example: a black part with small red markings is more appropriately described as 'black and red' than 'red and black'.
WithSeq	Primary together with secondary states	This is a special case of AndSeq, and in many circumstances (except natural language generation) may be treated as AndSeq. Example: "Green with brown" (often this may be two characters, e. g., base color and dot color). All states except for the first are considered secondary.
Between	Intermediate value between states	True value lying intermediate between (usually two) states. Example: "Between oval and elliptic" = "Oval to elliptic".

90. Boolean operators connecting descriptive statements that refer to the same property are problematic because they interact with implied semantics (knowledge whether an object part is repeated or not), and the customary data representation of a property.

91. The semantics of ‘*and*’ or ‘*or*’ in natural language descriptions or in DELTA data sets is often ambiguous. It may be desirable to be able to distinguish in the information model between an “ambiguous or” in the sense of one of ‘*and*’, ‘*or*’, and ‘*xor*’, and an ‘*or*’ defined in the sense of Boolean logic.

Boolean operators between characters

Closely related to Boolean operators between prepositions referring to the same character is the case of combinations of different characters. In DELTA by default the states within a character are combined with *'or'* and different characters are implicitly combined with *'and'* (example: “flowers sympetalous and red or orange”, not “flowers sympetalous or red or orange”. This leads to two questions:

- Firstly, should the DELTA model be extended, combining states with *'and'* by recording the same character twice? The SDD model supports repeating character data elements in a description-container and attaches this semantic. The primary reason for repeated characters is, however, not recording flowers “black and white” (because striped), but supporting modifiers. Modifiers may be used to express length measured at different places, or flower color recorded at different development stages. The intended semantics are an *'and'* over the entire development of an organism, not that both characters must be observable in a given individual at a given point in time. This is in line with the interpretation of *'and'* between different characters. Again, some characters may relate to flowering, others to fruiting characters, and no observation of both is guaranteed. If both can be observed, the intended Boolean operator is, however, an *'and'*.
- Secondly, under which conditions does the information model need to support *'or'* between characters? As long as characters are independent, this is unlikely. However, a need may exist to express more complex statements. An example given in Pullan & al. (2005) is that a plant may have “green hairy leaves” and “yellow glabrous leaves”, requiring an expression of the form “(green *and* hairy) *Or* (yellow *and* glabrous)”.

Expressing information as in the last examples requires either an arbitrarily complex Boolean expression language, or at least one level of containers for which Boolean rules have been defined. DELTA, like most descriptive information models, does not offer an explicit aggregation container for individual object parts. Thus the variation between object parts must be aggregated up to the level of at least individuals and the two kind of leaves summarized as “(green *or* yellow) *and* (hairy *or* glabrous)”. An individual plant having green and glabrous leaves would match this description, even if no such plant was ever recorded. In the context of species identification this may often be acceptable: it is better to return too many results – which can then be further reduced by additional criteria – than too few results. However, in the case of descriptions of higher taxa (e. g., plant families) the lack of support for additional levels is often problematic because identifications lack resolution power. Lebbe & Vignes (1998) call this the “problem of over-generalization” resulting from a “conjunctive bias”.

SDD provides two kinds of containers that may be used to express such knowledge: On the lowest level it may be expressed on the data recording level through the use of the SampleData container intended for original measurements. Sample data may contain linked observations like “color+hairiness” (see “Linked observations”, p. 90). On higher taxonomic levels the preferred solution in SDD is to create as many description-containers as necessary, even for a single individual object. Each leaf could be recorded in its own description, optionally scoped to describing only single leaves. This option allows recording any mixture of direct measurements and aggregated data in a linked form. Because any two descriptions (e. g., two individuals in a taxon) are combined with *'or'*, no further support for this case was considered necessary in SDD.

The Prometheus description model (p. 21) offers greatly improved options to describe objects at different spatial aggregation levels, by first “creating” a part hierarchy and then associating property values with these. By “cloning” a leaf multiple times in a description, it is possible to express separate color+hairiness for each instance of the leaf. The use of *'and'* and *'or'* is fully defined in Prometheus. The model uses the opposite default assumptions to DELTA: Multiple states within a character (“description element” in Prometheus) are always combined with *'and'* and multiple instance of the same character with *'or'* (Paterson & al. 2004, Fig. 2). Thus, the Boolean operator between character data elements depends on whether the same (\rightarrow *'or'*) or dif-

ferent (\rightarrow ‘and’) properties are referred to. Nevertheless, where variability or polymorphism is the result of repeated parts with different properties, this model is perhaps quite intuitive. However, despite considerable complexity of the Prometheus model, it covers only the third of the three causes of infra-individual variation discussed on p. 93.

Whether descriptive information models support aggregation and Boolean statements within individuals (such as object parts, phases in life cycle) or not, the ambiguity regarding the individual versus class level is very similar to the cases discussed in the previous section.

92. Boolean operators connecting descriptive statements that refer to different properties have similar problems to those mentioned under requirement 90, p. 97.
93. Addressing all potential Boolean combinations in a structured way easily leads to highly complicated models. A simpler requirement may be the support of multiple container levels with a defined operator behavior between them. This remains an open problem in current models.

4.10. Inheriting data

Data compilation versus data inheritance

Shattuck & Fitzsimmons (2000) make a distinction between data compilation and data inheritance, defining *data compilation* as an active process. In data compilation, the results of aggregating methods are accepted as new a verified form of “analytical data”. Compiled data may be based on a selection of available source data (e. g., after removal of outliers) and later changes in the base data will not automatically update the compiled data. Compiled data have the same trust mechanism that manually entered data have. If the compilation occurs within the system, the only specialty is that it may be desirable to maintain some information linking back to the source data that have been used during compilation. Doing so simplifies collaboration, peer review, and analysis of errors suspected long after the compilation occurred. The situation where some data in a compilation reside inside a descriptive information model and some outside should be considered and handled through appropriate citation mechanisms, including options for handling cases where some of the sources are unknown.

In contrast, *data inheritance* (i. e., up or down the taxonomic hierarchy) may be defined as an automatic process, where inherited data change immediately with the source data. This process has different properties, and may result in more or less reliable data. It is probably commonly agreed, that a data inheritance model is to be designed so that entering data at a given point will stop inheritance and replace the inherited data with the new data.

94. For “data aggregation”/“data compilation” it may be desirable to add a feature enabling the documentation of aggregation/compilation source. The model should be flexible enough to provide machine-readable citations for data in the same information system, human-readable citations for external but citable sources, and explicit options to inform on ignorance, perhaps mixed with source references.
95. Support for data inheritance is desirable. Inherited (automatically updated) data may have a different level of reliability, and their nature must be communicated to the data consumer.

Inductive inheritance (upwards)

The previous section 4.9 (“Raw data and data aggregation”, p. 83) discussed the various aspects of aggregating information upwards from one level of abstraction to the next higher level. This may be from parts to individual, individuals to population, populations to lowest taxonomic rank

(lowest class), and from lower classes to higher classes, but also along lines dictated by a specific analysis interest (within an ecosystem, within a season, etc.).

The term “inheritance” is here considered to be limited to the less general case of operations along the lines of the taxonomic hierarchy. Aggregating along this line is a frequently desired feature, e. g., to automatically create a genus description from all species in the genus for which data have been recorded.

Inheritance upwards the taxonomic hierarchy is not much different from manual forms of data aggregation (data compilation). In both methods documentation of the source data on which aggregation was based is desirable, and both methods may result in the same data. The meta-information desirable to support is whether a compilation has been reviewed and should not again be changed by the system, or whether inherited data may be updated at any time by a system.

96. A distinction between manually compiled (aggregated and reviewed) aggregation data, and inherited (i. e., automatically compiled) aggregation data is desirable. This may be a meta-data item on the data.

Deductive inheritance (downwards)

Inheritance may also occur from higher to lower levels of abstraction. Terms used for this process are *data propagation* (Maxted & al. 1993), *downward inheritance* (Shattuck & Fitzsimmons 2000), or *deducing data* (SDD discussions: Hagedorn 2003c). “Data propagation” seems to be insufficiently intuitive about the direction of propagation; the latter terms are perhaps equally appropriate.

Data deduction is possible because members of a class will normally match the class description. This is guaranteed in monothetically defined and likely in polythetically defined classes (Radford & al. 1974). In phylogenetic work, the question whether it is possible to deduce data requires a distinction between descriptions of higher taxa (class description) and inferred descriptions of ancestral taxa (phylogenetic reconstructions like a ground plan/ground pattern = “Grundmuster” sensu Ax 1984). The latter may contain features involved in later autoapomorphies or reversals, being not deducible to all children. The description of all higher taxa would be monothetic if they are limited to exactly the shared symplesiomorphous characters of all included taxa. In this case, information can be fully deduced. (No similar phylogenetic reasoning is necessarily available for upwards inheritance of data aggregation; aggregating data of a property like color may be meaningful diagnostic purposes, whether or not this is due to homology or not).

Although the arguments above are relevant, they make the assumption that all data are known. If this would be the case, deducing data from higher taxon descriptions would simply be a method to save storage space. In practice, the value of deductions lies in the case of missing information. A considerably more relevant case is that a feature has been studied in a more or less representative sample of class members and is believed to be monothetic for the entire group. The class description thus contains a hypothesis with respect to this feature that is accepted until falsified. Such hypotheses will commonly be made if the taxonomic hierarchy is phylogenetic and well established, and if the feature in question is known to be conservative. Information models providing for deductive data inheritance allow modeling this precisely, rather than forcing data managers to copy deduced information down to member taxa.

Deduced information should not normally be included in a data compilation process. Scientific data should always be reliable, and even if all other species in a family share a given characteristic that has not yet been observed in a given species, it may well be different in the current one. In practice, deduced data may influence compilation decisions, especially when dealing with data that can be proven to be error-prone (“noisy”). No explicit mechanism in the information model

is needed for this and in applications it may be desirable to deliberately make deductive or downward compilations difficult rather than easy.

A major reason why contrary to the above in existing systems data are often copied downwards is that matrix-based multi-access keys become more efficient, if the matrix is highly filled. Thus descriptive statements may be copied downwards from family or genus descriptions, even though they have never been verified. For infraspecific taxa or races, even conventional characters may not be explicitly known. Often only those characters distinguishing infraspecific taxa from each other are available directly. If characters have been copied without reflecting their origin, the trust in the entire data set may quickly deteriorate if new studies show contradictions to the recorded. Offering a data inheritance method for deduced information is therefore highly desirable and may improve the reliability of descriptive data sets.

97. Support for deductive data inheritance from descriptions of higher classes to lower classes (or individuals) is highly desirable.
98. Support for deductive data compilation is *not* desirable.

Current models

Building a descriptive data system with full support for compilation, inheritance, aggregation, and deduction is a difficult task. BioLink (p. 22) is the only implementation known to the present author supporting this in its native storage model. The inheritance feature has been little tested (S. Shattuck, pers. comm.). BioLink is currently no longer publicly supported and the model could not yet be studied.

In principle, whether inherited data are included in native data storage (flagged as such) or not, is an implementation-specific question. However, to decide whether data contain coding artifacts (especially inherited data copied down) it may be relevant to know whether and to which extent an information model supports inheritance.

The question may be different for data exchange standards. Adding the results of inheritance of calculations to data being transferred enables consumers to analyze data (including using it for identification) without requiring them to perform all kinds of calculations that the originator application was able to perform. It is desirable to flag data that are the result of calculations (e. g., ratio calculations, manual or automatic aggregations, deductive inheritance).

DELTA does not support flagging of inherited data, although it supports a limited *one-level* mechanism called “variant items” or “multi-item taxa”. This mechanism allows deductive inheritance from the description item immediately above for either subspecies or specimens. However, the underlying problem is a general one, which may occur repeatedly on different levels of the taxonomic hierarchy.

New DELTA (p. 20) proposes to add support for two “coded comments” (see p. 193 in “Modifiers”) with the following definitions: “@up = Attribute generated from information passed up the taxonomic hierarchy” and “@down = Attribute generated from information passed down the taxonomic hierarchy”. These proposed flags are not able to distinguish between data compilation and inheritance (in the sense of Shattuck & Fitzsimmons 2000).

SDD (p. 20) provides a tentative solution in the DataOriginEnum for this purpose (Table 23). The value “Inherited” is intended to mark the case of “deductive inheritance (downwards)” as used here. A similar value for inductive inheritance upwards (i. e., aggregation which may be overwritten at any time) is missing.

Further metadata may be desirable, allowing data set authors to influence whether a data item will be inherited by other taxa. This may be implemented both as “do not let my information be inherited” or “do not inherit information from elsewhere”. This topic needs further analysis. No examples demanding this feature could be found, but in the absence of applications implementing data inheritance little experience exists so far.

Table 23. Enumerated values in the “DataOriginEnum” in SDD 1.1, documenting the origin of a descriptive data value. The description of ‘Inherited’ is in contradiction to definitions used in this thesis and may have to be clarified.

Value	Description
OriginalData	The data are directly entered by a machine or human agent. These are the original data all other cached data (Origin unequal ‘OriginalData’) are based upon.
Calculated	The data are calculated from other data using a calculation rule. Examples: a ratio calculated from other characters, a mean calculated from a sample that is available under SampleData/Sample (if a mean is calculated from data no longer available, it would be recorded as ‘OriginalData’).
Mapped	A special case of “Calculated”, where the data are calculated using a mapping definition (either from numeric to categorical, or from fine-grained categorical to coarse-grained categorical).
Aggregated	The data are derived from data in classes placed below the current class in the class hierarchy. This applies both to aggregating data from objects to classes, as generalizing lower classes to higher classes. Note: BioLink (p. 22) calls this ‘Compile from below’.
Inherited	The data are derived from data in classes placed above the current class in the class hierarchy.

99. Information whether an information model does or does not support data inheritance mechanisms may be important to assess data quality, especially whether data might have been copied downwards to improve the operation of identification tools.
100. Whether inheritance needs to be broken at the source (e. g., “do not allow this to be inherited”) or the destination (e. g., “do not inherit from above, even if missing”) by means other than adding data needs further study. It may be desirable, but complicates the system and no good example cases could be found.

Implicit data

Certain information occurs so frequently in descriptions that it is usually omitted, only deviant states being reported. Most plant descriptions do not contain “plant upright, leaves green, root present”, even though many plants do deviate from this. The decision to consider certain character states “implicit” for a taxonomic group may have originally been in the interest of saving printing space. However, saving the “attention span” of data consumers remains an equally important goal. In traditional descriptions the decision was often not conscious and documented, but rooted in experience and tradition. In a system for coded descriptive data, this decision must usually be made consciously and should be based on sound algorithms.

The problem of implicit data is a large obstacle to extracting coded data from natural language descriptions. In descriptions authored by humans, often data that would not be implicit are overlooked and forgotten nevertheless. Algorithms that try to recover implicit data usually fail to distinguish these erroneous situations from truly implicit data. Again, the availability of algorithms deciding which data can be considered implicit may help to distinguish these cases.

Implicit data are not equivalent to “default values” in databases. “Default values” are inserted into a new database record in the absence of other information, and from then on become indistinguishable from normal values. Such a mechanism may be used to fulfill part of the desired functionality, but subsequently these values need to be recognized and suppressed during certain types of data output (e. g., natural language descriptions).

DELTA supports an “implicit” state mechanism, enabling – on the level of terminology – to define character states as “implicit states”. These must be defined for the entire project, which occasionally causes problems where a project combines heterogeneous taxon groups. Implicit states apply if in a given description no information has been recorded for a given character. DELTA is not explicit whether this includes coding status values or not, i. e., if the character value is set to ‘U’ for unknown, whether the implicit state ‘1’ should be treated as present or not.

Rules for inserting implicit states, i. e., converting them to normal data, are not part of the DELTA exchange standard.

The problem may be reconsidered when data inheritance and deduction are implemented in a system. DELTA essentially supports deduced data inheritance at the very top (for an entire project, through “implicit states”) and the very bottom (for specimens or subspecies through “variant items”). With a general mechanism available at all levels of the taxonomic hierarchy, neither of these specialized mechanisms is needed.

DiversityDescriptions supports the DELTA model (Boolean CS.Implicit), but makes little use of it (other than supporting inserting as normal data). It is planned to use this information in a future implementation of a general data inheritance model to provide data for the top node in the taxon/entity hierarchy, to be deduced downwards. In the absence of a top node, a system-generated “default description for project” may be created. The DELTA variant item option is not supported at all by DiversityDescriptions.

101. Support for data inheritance and deduction removes the need to support defining “implicit states” in the terminology. Data using the “implicit states” model are convertible into a hierarchical data inheritance model.

Compatibility testing as a quality control measure

If information from several sources is available, an alternative to directly combining the information through data aggregation (p. 85) or deduction (p. 100) is to compare them for “compatibility”. To some extent, this method may be used as a plausibility and quality control measure.

One approach is to consider descriptions compatible if the state combination found in the synthesis also occur either positively or as unknown/uncertain in at least one description on which they are based (Fig. 34). This handles certainty modifiers and coding status satisfactorily (Fig. 34, character 2 and 3).

Similar analyses are possible for quantitative characters, e. g., they may be considered compatible if any of their range parameters overlap (perhaps including calculated ranges where mean and variance measure are available). The sensitivity of the analysis can then be influenced by using normal ranges or always the absolute range (where known). Alternatively, the condition for compatibility may be that the mean should be in the combined range of all descriptions for which a range is available.

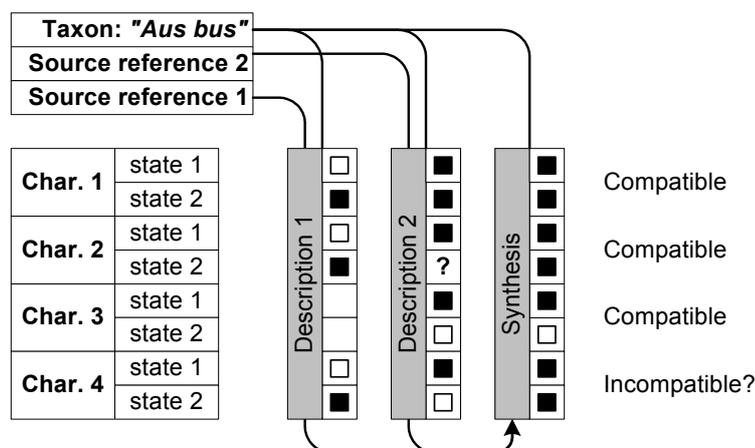


Figure 34. Compatibility of descriptions. Description 1 and 2 are considered to be compatible for character 1 to 3 (the “?” in char. 2, state 2 representing a certainty modifier like “doubtful”), but not for character 4 – even though the situation in character 4 may validly occur.

The result of a compatibility test can highlight potential conflicts, but cannot establish that a conflict exists. For example, for a given character, a species may be polymorphic, but individuals or populations may be monomorphic. Similarly, species within a genus will quite often have incompatible descriptions. In a well developed data set, however, it is less likely that a specimen description is incompatible with its species description. Here a form of compatibility testing may be used as a plausibility check in a data editor.

The practical usefulness of compatibility testing needs further analysis. It may be useful only if a sufficient number of descriptions are available (e. g., to search for “outliers”). Its greatest use may occur when testing data that are deduced from higher taxa. For example, one may want to search infrageneric descriptions that are incompatible with a compiled genus description.

4.11. Description storage models

Introduction

Until now a discussion of the exact mechanism of storing structured descriptive data and corresponding descriptive terminologies (compare p. 42) has been avoided. The intention was to lay as much groundwork as possible without entering this contentious topic.

All description models discussed here share a very basic common pattern: objects or classes (also called “individuals”, “items”, “taxa”; compare Table 3, p. 34) are described by a number of character variables (also called “attributes”, “properties”, “characters”; see “The term ‘character’”, p. 31 and Table 3, p. 34). Some questions differentiating between descriptive models are:

- Is the primary level of variables a complex type, containing sets or lists of values, or are all variables of a simple type and arranged in a flat list or matrix? The first model is often called “character matrix”, the second model (at least for categorical data) a “character state matrix”.
- Should variables be fully defined in the terminology, or should terminological elements be made combinable at the time when the descriptions are recorded? Specifically:
 - Is the whole of the object described directly in a flat list of variables, or does this occur through independent object composition (i. e., concepts having part-of relations) and variable types (“property”) hierarchies, which are freely combinable in descriptions?
 - Are other parts of the terminology (such as modifiers, see p. 189) freely combinable in descriptions?

The term “*entity*” will be used when referring to the thing being described, without saying whether it is a class (taxon) or an individual object (specimen, etc.).

Categorical data: Character matrix vs. character state matrix

Descriptive data are commonly visualized as a matrix of variables \times entities (or entities \times variables). As Diederich (1997) points out, biologists may prefer a matrix view even when the matrix is extremely sparsely filled. A major point of discussion is whether this matrix should be based on characters (Table 24), or rather on states (Table 25). Each model has some visualization and data processing advantages, especially concerning characters where the data are a set of state scores. However, as will be shown, the two models are essentially convertible.

A character \times entity matrix is the preferred visualization in the new CSIRO DELTA Windows editor (although storage in the DELTA format does not use a matrix). However, such a matrix is directly used as the data storage format by NEXUS (p. 18), Pandora (p. 19), and some databases supporting the EFG project (p. 20). For applications using a relational DBMS for data

Table 24. Character \times entity matrix. In char. 1 states are exclusive, in char. 2 and 3 multiple states are scored (polymorphism, shown in parentheses).

Entity	Char 1	Char 2	Char 3
1	1	1	(1/2/3)
2	1	[?]	3
3	2	2	2
4	2	(1/2)	

storage, the approach may lead to scalability problems. Most DBMS have a system limit on the number of columns per table so that it may not be possible to scale from small data sets with few characters to large integrated data sets (which easily exceed thousand characters). The design further requires a solution for multiple values (multi-state data or multiple statistical measures) per character. The NEXUS Matrix subcommand uses parentheses to indicate multiple categorical values as shown in Table 24. Since this is inconvenient when editing NEXUS files in a text editor, a special “Equate” subcommand allows defining a single symbol for common polymorphisms. In Pandora a database table directly represents a character \times entity matrix and multiple state values are stored as DELTA-formatted text with delimiters (R. Pankhurst, pers. comm.). In a standard relational database this would be problematic because any querying and processing would require parsing of the data content. The Advanced RevelationTM system used for Pandora has special support for multiple-value fields, separately indexing delimited subfields, and the performance of queries acting on states remains good. The EFG project is in fact experiencing problems with handling multistate character values, treating each combination as a separate value (R. Morris, pers. comm.).

The alternative representation is a character state \times entity matrix (Table 25). Each cell in this matrix has a simple data type and if desired, the matrix could be directly implemented as a table in a relational database. Since each character state has its own column, the scoring is now binary (state present/absent). The disadvantage of this model is that it is often more difficult for humans to understand because the binary information is distributed over a large number of columns. If characters are well-defined concepts, each column in a character \times entity matrix will answer a question of interest to humans like “what is the leaf shape?”, whereas each column in a state \times entity matrix answers questions like “is the leaf shape lanceolate?”, “is the leaf shape ovate?”, etc.

For this reason, most programs provide for character-like concepts in the terminology, even if they use a state matrix for descriptive data. Exceptions are MEKA (Meacham 2007) and AditKey (Adit 2004), which do not have a character concept. Character states may be arranged into various index categories, but within an index category a flat list of states is displayed.

Table 25. Character state \times entity matrix. In character 1 the states are exclusive; in character 2 and 3 multiple states may be scored.

Entity	Char 1 State 1	State 2	Char 2 State 1	State 2	“?”-state	Char 3 State 1	State 2	State 3
1	<input checked="" type="checkbox"/>	<input type="checkbox"/>	<input checked="" type="checkbox"/>	<input type="checkbox"/>	<input type="checkbox"/>	<input checked="" type="checkbox"/>	<input checked="" type="checkbox"/>	<input checked="" type="checkbox"/>
2	<input checked="" type="checkbox"/>	<input type="checkbox"/>	<input type="checkbox"/>	<input type="checkbox"/>	<input checked="" type="checkbox"/>	<input type="checkbox"/>	<input type="checkbox"/>	<input checked="" type="checkbox"/>
3	<input type="checkbox"/>	<input checked="" type="checkbox"/>	<input type="checkbox"/>	<input checked="" type="checkbox"/>	<input type="checkbox"/>	<input type="checkbox"/>	<input checked="" type="checkbox"/>	<input type="checkbox"/>
4	<input type="checkbox"/>	<input checked="" type="checkbox"/>	<input checked="" type="checkbox"/>	<input checked="" type="checkbox"/>	<input type="checkbox"/>	<input type="checkbox"/>	<input type="checkbox"/>	<input type="checkbox"/>

A modified version of the character state \times entity matrix is, for example, used by CBIT Lucid3 (p. 21) for data storage. In the user interface, however, characters are recognized and the states are organized into characters. Lucid modifies the binary state presence/absence matrix insofar, as it stores 8-bit integers and uses the 7 remaining bits to store selected modifier information (frequency, coding certainty, and coding status information) with each state.

It is possible to slightly reduce the size of the matrix, by treating characters defined as exclusive (char. 1 in the example) in a single column (Table 26). In practice, however, characters with exclusive values are extremely rare. Even in object (specimen) descriptions often a part occurs multiple times or changes over time, so that the description is already a data aggregation rather than original observations (compare “Aggregation within individuals”, p. 93). The mixed model is therefore not discussed any further.

Table 26. Mixed model, using the character \times entity matrix for “character 1” with exclusive states.

Entity	Char 1 (Exclusive)	Char 2 State 1	State 2	“?”-state	Char 3 State 1	State 2	State 3
1	1	<input checked="" type="checkbox"/>	<input type="checkbox"/>	<input type="checkbox"/>	<input checked="" type="checkbox"/>	<input checked="" type="checkbox"/>	<input checked="" type="checkbox"/>
2	1	<input type="checkbox"/>	<input checked="" type="checkbox"/>	<input checked="" type="checkbox"/>	<input type="checkbox"/>	<input type="checkbox"/>	<input checked="" type="checkbox"/>
3	2	<input type="checkbox"/>	<input checked="" type="checkbox"/>	<input type="checkbox"/>	<input type="checkbox"/>	<input checked="" type="checkbox"/>	<input type="checkbox"/>
4	2	<input checked="" type="checkbox"/>	<input checked="" type="checkbox"/>	<input type="checkbox"/>	<input type="checkbox"/>	<input type="checkbox"/>	<input type="checkbox"/>

In practice the number of columns in character state \times entity matrices will be much higher than in the examples shown. For example, as of May 2007, the LIAS data set (Rambold 1997, Rambold & Triebel 2007) used 844 categorical characters with an average of 9.8 states per character, totaling 8 271 states (G. Rambold, pers. comm.). Including columns for coding status (as in Table 25), the number of columns needed is even 10 732. Clearly it is not advisable to attempt a direct implementation of the matrix metaphor in a relational DBMS. The Pandora system is able to do this (a character \times entity matrix is implemented as columns and rows in a table), but most DBMS limit the number of columns (attributes) that can be defined per table.

The approach used both in the DiversityDescriptions (p. 322) relational database model for DELTA data and in the storage outline for the Genisys model (Diederich 1997) is therefore based on a list model (Table 27). This model is also rather close to the lists of states that are recorded in the DELTA text format. Each occurrence of a state is now recorded in its own row. States that are absent (in Tables 25-26) are not recorded.

Both the character matrix and the state matrix model can be generated by two different cross-tabulation transformations (also called “pivot analysis”) from the list, if information about the terminology used during coding is separately available. When converting a list model to a character matrix, the terminology provides information about absent characters; when converting it to a character state matrix, the terminology provides also the information about “absent” states. Although this conversion assumption holds for the primary requirements of a descriptive information model (after all, functional applications exist for all models!), some differences under specific perspectives do exist. These are discussed in the following:

1. Coding status: One situation where the character state matrix model differs from the character matrix or list models is expression of coding status (e. g., “unknown”, “not applicable”; compare p. 74). Coding status here is assumed to apply to a character as a whole. The fundamental assumption is that for well-defined characters (characters independent, states within each character dependent), it is impossible to observe only a subset of states in an object. In the examples given so far, the “[?]” represents an example for an explicit coding status value. This is visualized as a separate column in the character state \times entity matrix (Table 25). Initially one may assume that this is problematic because the concept of a coding status value includes an assumption that no other information will be coded for the present character in the present entity. Although this is correct for original recordings, it is not true for aggregated data. For example, a genus description with “leaf hairs simple, not applicable, or unknown” is meaningful, if the aggregated information originates in three different species of the genus. Thus a general rule that in the character state matrix model “coding status values cannot occur together with states data” cannot be stated. Although the separate “matrix columns” for coding status values do have some special properties, they are not artifacts.

Table 27. Presentation of the matrix models (Table 24-26) as a list.

Entity	Char	State
1	Char 1	1
2	Char 1	1
3	Char 1	2
4	Char 1	2
1	Char 2	1
2	Char 2	[?]
3	Char 2	2
4	Char 2	1
4	Char 2	2
1	Char 3	1
1	Char 3	2
1	Char 3	3
2	Char 3	3
3	Char 3	2

Notes: Rows involving multi-states scoring are highlighted. The condition that states of character 1 are exclusive is no longer enforced by the model itself. Also note that no record exists for entity 4, char. 3.

The case of implicit status “not coded” (i. e., no coding occurred at all; shown as empty cell in Table 24, entity 4, char. 3) is not easily translated into the presence/absence values of the character state matrix model. It is possible to extend the model to a triple-state of ‘present’, ‘absent’, ‘uncoded/never considered’. The coding status would apply if all states of a character have the “uncoded” value, and the system would be responsible for appropriate initialization.

2. “State-absent statements” (i. e., an explicit statement that a state is not present) may be desirable for a variety of reasons. Some arguments for “state-absent statements” are weak:

- The most frequent desire is perhaps to express certainty (see p. 207). However, “state 1 probably absent” and “state 1 perhaps present” express the same information. Normalization is desirable in any case; here negative statements with certainty modifiers may impede rather than improve comparability.
- Many situations where it seems to be desirable to qualify the absence of a state (or to express coding status on a state-by-state basis) are due to poorly defined summary characters, combining independent variables. An example for this would be a character “secondary metabolites” with different metabolites defined as states. Since the metabolites are independent, they should be placed in separate characters (if necessary using states like: ‘present’, ‘absent’, ‘unknown: all literature studied’, ‘unknown: literature not yet checked’). The desire to combine multiple characters into a single one is probably related to the lack of mechanisms to organize characters hierarchically in current applications.

However, other arguments cannot be easily refuted:

- In practice it is almost unavoidable that the *descriptive terminology evolves over time*, including the addition of new state values to previously defined characters. The influence of new state values on existing descriptions must then be analyzed. No influence should be present if states are truly distinct and well-defined – the assumption being that a new state is being introduced as soon as it is needed. However, in practice states often somewhat arbitrarily partition a continuous value space (i. e., the real world values are extended and adjacent, Fig. 17 e-f, p. 54). If new states are intermediate between existing states existing descriptions using neighboring states may have to be revised. If either all existing states that had already been considered are marked with some kind of “state-absent statement”, alternatively, if for new states an “uncoded state” or “not yet considered” marker is inserted into the data matrix, the management of such revisions would be simplified.
- Occasionally negative statements like “not so”/“not green” etc. need to be captured from *published descriptions*. In the simplest case, where the description in the source aims at being complete and the set of potential states in the source and in the terminology used for digitization are identical, such an “absent coding” can simply be converted into a complement of positive “state-present” scores. However, (a) the source information may not claim to be a complete representation of state values (especially in dichotomous/polytomous keys or brief diagnostic descriptions (p. 39) information considered irrelevant for identification will simply be missing), (b) differences between the states used in the terminology of the source and the data set may make such a complement very difficult, (c) a negative statement may express that the character is highly variable, but one or several states are specifically known not to occur. One – rather unsatisfactory – coding method would be to record the complement (all states except those mentioned) as “perhaps present” (compare “Certainty modifiers”, p. 207).
- Similarly, if data are edited collaboratively, it may be desirable to explicitly contradict a statement somebody else has made, rather than simply deleting that statement. This is less important for the correction of simple errors (if a full history of all changes is available, state deletions may be reported using data comparison tools), but for controversial changes and where a discussion is actually desired, a “rejected-state statement” may highlight a need for discussion and provide a place to attach free-form text containing additional discussion material. Analysis of the best methods to organize collaborative revisions and discussions is an important topic for the future. It is not yet implemented in any current descriptive software. For exam-

ple, it remains an open question whether support for contradictions should be on the character or state level.

3. States “depending” on the taxonomic tree: Another form of metadata associated with absent states is an expression that a state is “not occurring” for an entire higher taxon (e. g., an order or a family) and all included lower taxa. This situation must be distinguished from character applicability (i. e., some character like “leaf hairiness” cannot logically be recorded because of the state of another character like “leaves absent”; see “Character applicability rules”, p. 76). In a well-defined terminology, where all states are indeed categorical values of the same character variable, a logical dependency of states that is independent of a character dependency is believed to be impossible (examples claiming this are usually combinations of multiple characters thrown together for the convenience of coding, such as “secondary metabolites”, where each metabolite, because independent, should be a character of its own). However, an analyzable correlation may exist between states and the taxonomic tree. Here, in a given branch of the taxonomic tree certain character states do not occur in the data set as recorded so far. As soon as this correlation is sufficiently stable for the purpose of coding new object descriptions, this information may help to remove irrelevant coding options in the interest of short editing and data review forms. Normally, however, “not occurring” information may be considered to be analyzable from the data set itself. Although it may be desirable to preprocess and cache this information, these are implementation details with which the information model should not be concerned. Under rare circumstances it may be desirable to also demand complete freedom of form design, excluding character states from editing forms even where they do exist, but in general this will severely impede the data recording process, whenever a new taxon does have such an excluded state, so that this option is not recommended (assuming well-defined characters). In contrast, the equivalent option to exclude characters in some branches of the taxonomic tree (because considered irrelevant or too expensive to code) is much more relevant. This case may be handled by adding a “Not to be coded” coding status (p. 74), which may then be inherited up and down the taxonomic tree.

The data items called for in the discussion points above can be provided in all three models discussed. Coding status is perhaps more natural to the list or character matrix model. On the other hand, the character state \times entity matrix (Table 25) probably offers a more natural place for information why a state is absent. The character matrix and list models need separate extensions for this (e. g., separate data structures to express metadata about the scoring process, including negative statements and “states-considered”, or perhaps a ‘*not*’-modifier that has to be observed during identification or data analysis).

Current software applications and data exchange formats offer no insight into this problem: no implementation seems to have been attempted so far. This includes the state matrix-based CBIT Lucid3 (p. 21), where modifier information is only available on positive-state scores.

One consideration that may be helpful for the decision how “State-absent statements” shall be handled, may be the expected behavior during data compilation or inheritance up the taxonomic hierarchy (aggregation, p. 83) and inheritance of data down the taxonomic hierarchy (deduction, p. 100). The scoring of states as present and absent is traditionally considered an asymmetrical process. Because a state may occur with a low frequency, a negative statement is usually interpreted as possibly meaning “not (yet) observed”. As a result, during aggregation, positive statements win over negative scores (Table 28). Explicit “state-absent statements” do not change this behavior.

Conversely, where information is deduced from higher taxa, traditionally inheritance occurs only where no state for a character has been scored (Table 29: species 1 inherits all state values, because not scored for leaf shape, species 2 inherits nothing because “leaf shape: lanceolate” has been scored). This model can easily be extended to prevent inheritance also if a species has at least one explicit “state-absent statement” for a character (Table 29: species 3). However, whether it is meaningful that “state-absent statements” from higher taxa are inherited by lower

Table 28. Asymmetry of positive and negative state scores during aggregation (induction of knowledge “up the tree”). Explicit state-absent statements (“☒”) behave identical to not-scored (“□”). Parentheses (“(☒)”, “(□)”) indicate aggregated (or “inherited”) values.

Entity	Leaf shape		
	ovate	lanceolate	linear
Species 1, specimen 1	☑	□	□
Species 1, specimen 2	□	☑	□
Species 1, specimen 3	☒	☑	□
Species 1 aggregation	(☑)	(☑)	(□)

Table 29. Asymmetry in the intuitiveness of positive and negative scores inherited from higher to lower taxa (deducing knowledge “down the tree”). Parentheses (“(☑)”, “(□)”) indicate inheritance; “☒” indicates a “state-absent statement”.

Entity	Leaf shape		
	ovate	lanceolate	linear
Family	☑	□	☒
Species 1	(☑)	(□)	(☒)
Species 2	☑	□	□
Species 3	☒	□	□

taxa (Species 1, “linear”) is difficult to decide and needs further consideration. The result would not necessarily be incorrect, but inherited “state-absent statements” would be considerably less intuitive to interpret than “state-present statements”.

The most important decision in this discussion is, whether characters are considered logical entities that – in contrast to states – may be independently observed. If this is affirmed, observing a categorical character implies that all its states have been considered and are either “present” or “absent”. All existing descriptive software, whether using a character matrix or a character state matrix, makes this assumption. One consequence of this “character-model” is that certain rules (character dependency, character hierarchy, coding status, inheritance) can be formulated in a simpler way. However, another consequence is that some of the advantages of the character state × entity matrix model are countered by the additional provisions necessary to handle situations applying to entire characters. Indeed, in Lucid3 such special code is present, handling coding status logic which in the user interface is scored together with state-modifiers on a state level, but still maintains character-wide logic and consistency rules.

If, however, it is desired to accept “ill-defined” characters (i. e., characters combining independent data, like the “secondary metabolite” example), the entire character model would have to be dissolved. Coding status values on a character would make little sense, and instead of character dependency, a state dependency, and instead of character-scope-out a state-scope-out would have to be defined. These could then be inherited like character data or coding status values. For example, a state-data item could then have values like: ‘state-present’, ‘state-absent’, ‘state-not-considered’, ‘state-scoped-out’, ‘state-inapplicable’, ‘state-hidden’, etc.

102. For most purposes the “character matrix”, “character state matrix”, or “character state list” models are equivalent. For actual data exchange (especially when considering federated relational databases or XML formats) a list model may be the most flexible choice.
103. One area requiring different considerations in the three models is coding status and character dependency.
104. Whether information that states are considered “absent” or “false” should be preserved as data (and aggregated or inherited along the taxonomic hierarchy) is contentious. For the primary purposes of representing the descriptive data this is not necessary. However, a number of important secondary purposes exist, under which preserving this information may be valuable (negative statements, collaboration and discussion, evolution of terminology). In principle such information may be stored in all three models, but the state matrix model may be the most intuitive for this purpose.
105. If negative statements are supported, it may be desirable to *not* support certainty modifiers on these.
106. Dependency rules or “do-not-code” (also called “out-of-scope”) rules controlling single states instead of entire characters are probably *not* desirable.

Quantitative data and statistical measures

As discussed under “Standard aggregation methods” (p. 85), quantitative data values measured on samples may be summarized as frequency distributions (i. e. histograms), as distinct value lists (compare p. 86), or using univariate statistical measures (or human estimates of these). The total number of measures used in statistics is large; in some areas of classical morphological taxonomy, however, only a few measures are commonly used. The statistical detail necessary to adequately record quantitative characters strongly depends on the difficulty of the measurement process. Measuring plant leaf size is simpler than measuring spore sizes that are close to the optical resolution of light microscopes. Unfortunately, difficult measurements are indispensable in many groups (viruses, bacteria, microscopic fungi, microscopic plants, or microscopic animals), because the number of simple characters is too low. Thus, part of the differences in the number and kind of statistical measures supported in various description models depends on the taxonomic group they have been optimized for.

In the character state \times entity matrix model each statistical measure requires a new column. Implementations preferring this model may therefore prefer a relatively small number of implementation-defined measures, e. g., CBIT Lucid3 (p. 21) uses a fixed set of four values to express the extremes and “normal range” of quantitative variables. They are slightly confusingly labeled as “absolute minimum” (i. e. minimum), “minimum” (i. e., lower limit of some range measure), “maximum”, and “absolute maximum”. The Flora of North America-model uses a similar set of four extreme and range values (P. B. Heidorn, pers. comm.).

When categorical data are visualized as a character \times entity matrix, each matrix “cell” is already complex. Using a complex content for quantitative as well as categorical data probably appears more natural. When following a list-model for data storage, a flexible model supporting many statistical measures looks very similar to the categorical model (Table 30). However, the original DELTA format supports only fixed set of five measures: a minimum, an undefined lower range limit, an undefined central value (like mean, mode, or median), an undefined upper range limit, and a maximum. In DELTA these are expressed in an elaborate formula syntax of twelve alternative patterns (structured by hyphens and parentheses, e. g., “(3-) 5-7-9 (-11)”). Only the two extremes have a full semantic definition. The range and central measure/value are only weakly defined; their exact semantics are expected to be expressed in free-form text documentation associated with a data set. The limitation that is perhaps most practically relevant is that

DELTA cannot distinguish between a single value and a mean of sample. As a consequence, it is often difficult to decide which quantitative error tolerance to use during identification (see “Equality criteria and error tolerance”, p. 264), and the difference between ‘IN’ and ‘RN’ only informs collaborators and data consumers about the intentions of the data set author, but cannot be used even for plausibility checking during data entry. Only minimum and maximum in an ‘IN’ data type will always be integer.

As an alternative to measures, DELTA supports an unlimited number of values in a distinct value list.

The CSIRO DELTA windows editor (p. 19) requires knowledge of the DELTA-rules for forming these patterns in the DELTA language (Fig. 35). The example shown in the figure further illustrates that the statistical interpretation of DELTA data may be difficult. It is the responsibility of the author to provide semantics for the positions in the pattern, and the use of a frequency (e. g., “mostly”) on a statistical meas-

Table 30. Examples of statistical measures in the list model.

Entity	Char	Measure	Value
1	Char 1	minimum	2
1	Char 1	mode	6
1	Char 1	maximum	8
1	Char 1	N (sample size)	30
2	Char 1	min	3
1	Char 2	minimum	11.2
1	Char 2	5% conf. interval lower (2.5%)	11.8
1	Char 2	mean	15.3
1	Char 2	5% conf. interval upper (97.5%)	18.8
1	Char 2	maximum	20.4
1	Char 2	stand. dev. (n-1)	10.9
1	Char 2	standard error	1.7
1	Char 2	N (sample size)	50
2	Char 2	mean	14

Notes: Note that it is possible to use any subset of the measures defined for a character (entity 2).

	1	2	3	4
	<longevity of plants>	<mature> culms <maximum height: data unreliable for large genera>	culms <whether woody or herbaceous>	culms <whether branched above>
1	Agrostis <L.>	1/2 (3-)5-100	2	2
2	Andropogon <L.>	1/2 8-250(-430)	2	1/2
3	Anisopogon <R.Br.>	2 60-110	2	
4	Bambusa <Schreber>	2 (200-)500-3500	1	1
5	Chloris <O. Swartz>	1/2 10-300	2	
6	Cynodon <Rich.>	2 4-60(-100)	2	

Figure 35. Illustration of the CSIRO DELTA Windows editor in grid view. Character 1 is categorical, character 2 quantitative (real numeric). Data can be edited only in the lower left area; for quantitative data this requires knowledge of the DELTA language and the positional patterns supported.

ure indicates that the example uses some non-statistical semantics.

The data storage outline for the Genisys model (Diederich 1997) is limited to a different set of statistical measures: “Value, Low Range, High Range, Std. Deviation”. The authors themselves note that a more elaborate representation of measurements may be necessary.

CID	Type	CharName	[um thick]	Modifier	X	TXT	N
1	TE	Family - Subfamily - Tribe					
2	RN	Body length	(Min) Minimum value			35	K
3	RN	Body width	(Mean) Mean (= average)			32.5	K
4	RN	Body thickness	(Max) Maximum value			40	K
5	UM	Body flatness					
6	UM	Body robustness					
7	UM	Body form					
8	UM	Colour					
9	UM	Gnathosoma form of oral stylet					
10	UM	Gnathosoma form					
11	UM	Chelicerae form					
12	RN	Chelicerae length					
13	UM	Gnathosoma distal end whether with m...					

Figure 36. Illustration of the DiversityDescriptions database editor. A terminology-defined number of statistical measures (upper right area) can be entered in a form, plus coding status (DELTA pseudo-values) and the DELTA text-override (‘TE’) can be entered (middle and lower right area).

DiversityDescriptions (p. 322, Fig. 36) improves on the DELTA standard by supporting a larger number of semantically defined statistical measures. The actual data storage looks very similar to the example shown in Table 30, except that the column “Measure” is used for both categorical character states and statistical measure identifiers (field DD_DESCR.CS in Fig. 227, p. 335). Similar to categorical states, the measures are defined – separately for each character – by the designer of the terminology. These character-specific definitions enable the designer to limit the choices for a character, e. g., allowing only minimum and maximum in some characters. DiversityDescriptions supports a relatively large number of statistical measures (Table 60, p. 356), including variance measures and sample size which are notably missing in DELTA. The list of measures can in principle be extended by user-defined ones. Extended measures can be

edited and reported as simple label-value pairs, but are not further recognized by the built-in algorithms. Only recognized measures will be treated in equivalence classes (e. g., all range measures are treated as comparable in identification) and will be reported in special ways (e. g., as a min-range-central-range-max (sample-size) formula in natural language generation. The DELTA concepts of “undefined statistical measures” are treated as a special case (and the information about the uncertain definition can thus be preserved).

The NEXUS data exchange standard (p. 18) also provides a larger and flexible number of semantically defined statistical measures. With “Format DataType = Continuous”, the choice of measures and the order in which they appear in the matrix may be defined in the “Items” subcommand. The possible statistical measures (NEXUS: “items”) are: Min, Max, Average (default), Variance, StdError, Median, SampleSize. Alternatively, NEXUS supports an unlimited number of measurements in a distinct value list (invoked by “Items=(Average States)”). The values of the defined measures are arranged in a matrix where, similarly to the states of polymorphic categorical characters, parentheses enclose multiple statistical measures per character. Because of this structure, the format may be viewed as a character × entity matrix.

Germeier & Frese (2001), in a model for characterization and evaluation data for plant genetic resources, support another fixed set of statistical measures (modeled as attributes in the AccessionObservation and GenotypeObservation tables).

SDD supports a large set of 38 statistical measures. Whereas DiversityDescriptions in many cases made an arbitrary choice which confidence intervals or percentiles are supported (compare Table 60, p. 356), SDD uses parameterized statistical measures in these cases. For example, all percentiles are represented by two categorical values (“PercLower” and “PercUpper”), and elaborated with a quantitative parameter (e. g., “95%”).

It is interesting to note that those software applications and data exchange formats that severely constrain the number of statistical measures (i. e., except DiversityDescriptions and SDD), seem to be informed by different requirements and do not agree on a common set (Table 31). It seems obvious that a more general approach to statistical measures is desirable, supporting a large set of statistical measures, or a medium-sized set with an option for extensibility.

Table 31. Comparison of statistical measures supported in descriptive software applications and exchange formats.

Statistical Measure	DELTA	Lucid ¹	Genisys	NEXUS	Diversity- Descript. ²	Germeier & Frese (2001)	SDD
Minimum/Maximum	<input checked="" type="checkbox"/>	<input checked="" type="checkbox"/>	<input type="checkbox"/>	<input checked="" type="checkbox"/>	<input checked="" type="checkbox"/>	<input checked="" type="checkbox"/>	<input checked="" type="checkbox"/>
Range (undefined)	<input checked="" type="checkbox"/>	<input checked="" type="checkbox"/>	<input checked="" type="checkbox"/>	<input type="checkbox"/>	<input checked="" type="checkbox"/>	<input type="checkbox"/>	<input checked="" type="checkbox"/>
Range (defined) ³	<input type="checkbox"/>	<input type="checkbox"/>	<input type="checkbox"/>	<input type="checkbox"/>	<input checked="" type="checkbox"/>	<input type="checkbox"/>	<input checked="" type="checkbox"/>
Defined percentiles/ confidence intervals	<input type="checkbox"/>	<input type="checkbox"/>	<input type="checkbox"/>	<input type="checkbox"/>	<input checked="" type="checkbox"/>	<input type="checkbox"/>	<input checked="" type="checkbox"/>
Mean	<input type="checkbox"/>	<input type="checkbox"/>	<input type="checkbox"/>	<input checked="" type="checkbox"/>	<input checked="" type="checkbox"/>	<input type="checkbox"/>	<input checked="" type="checkbox"/>
Median	<input type="checkbox"/>	<input type="checkbox"/>	<input type="checkbox"/>	<input checked="" type="checkbox"/>	<input checked="" type="checkbox"/>	<input type="checkbox"/>	<input checked="" type="checkbox"/>
Central measure or value	<input checked="" type="checkbox"/>	<input type="checkbox"/>	<input checked="" type="checkbox"/>	<input checked="" type="checkbox"/> ⁴	<input checked="" type="checkbox"/>	<input checked="" type="checkbox"/>	<input checked="" type="checkbox"/>
Variance	<input type="checkbox"/>	<input type="checkbox"/>	<input type="checkbox"/>	<input checked="" type="checkbox"/>	<input checked="" type="checkbox"/>	<input checked="" type="checkbox"/>	<input checked="" type="checkbox"/>
Standard deviation	<input type="checkbox"/>	<input type="checkbox"/>	<input checked="" type="checkbox"/>	<input type="checkbox"/>	<input checked="" type="checkbox"/>	<input checked="" type="checkbox"/>	<input checked="" type="checkbox"/>
Standard error	<input type="checkbox"/>	<input type="checkbox"/>	<input type="checkbox"/>	<input checked="" type="checkbox"/>	<input checked="" type="checkbox"/>	<input checked="" type="checkbox"/>	<input checked="" type="checkbox"/>
Sample size	<input type="checkbox"/>	<input type="checkbox"/>	<input type="checkbox"/>	<input checked="" type="checkbox"/>	<input checked="" type="checkbox"/>	<input checked="" type="checkbox"/>	<input checked="" type="checkbox"/>
Coefficient of variation	<input type="checkbox"/>	<input type="checkbox"/>	<input type="checkbox"/>	<input type="checkbox"/>	(<input type="checkbox"/>)	<input checked="" type="checkbox"/>	<input checked="" type="checkbox"/>
Skewedness	<input type="checkbox"/>	<input type="checkbox"/>	<input type="checkbox"/>	<input type="checkbox"/>	(<input type="checkbox"/>)	<input checked="" type="checkbox"/>	<input checked="" type="checkbox"/>
Kurtosis	<input type="checkbox"/>	<input type="checkbox"/>	<input type="checkbox"/>	<input type="checkbox"/>	(<input type="checkbox"/>)	<input checked="" type="checkbox"/>	<input checked="" type="checkbox"/>

¹ Lucid (in version 1 to 3) is the only explicitly character state matrix model

² DiversityDescriptions has 23 built-in measures and is user-extensible for further measures (indicated by “(□)”).

³ Six lower and six upper defined interval measures (percentiles and confidence intervals)

⁴ In NEXUS value lists may be achieved using “Average States”

In principle, a large number of statistical measures could be directly available for data entry in any quantitative character. However, this introduces some issues in the user interface – and perhaps in the storage model as well (if storage is based on a character state \times entity matrix model and requires one column for each statistical measure). Furthermore, when in practice only few statistical measures are used, a large number of available measures may be confusing to data entry personnel and lead to data entry errors. These problems can be avoided, if the information model provides some means to define preferred statistical measures. One solution for this may be the DiversityDescriptions model, where only those statistical measures defined in the terminology are available for data recording. This model is especially attractive where a high number of statistical measures may create data storage problems (character state \times entity matrix model). However, if this is not the case, an alternative model may enable all available statistical measures for data storage and provide a character-specific “secondary filter” defining the preferred measures for data entry form. These could then be immediately visible, perhaps while providing a more complex method to (e. g., a separate form) to enable the occasional data entry of rarely used measures.

Explicitly negative statements: A parallel to state-absent statements discussed for categorical data is difficult to find for quantitative data. In principle a negative quantitative statement like “not 3 to 5 cm” may be converted into “ < 3 cm or > 5 cm” (i. e., “not 3 to 5” = “not (≥ 3 and ≤ 5)” = “not (≥ 3) or not (≤ 5)” = “ < 3 or > 5 ”). As a single statement it cannot be expressed in any set of statistical measures discussed so far. It can, however, be expressed as two separate conditions, combined with ‘or’: “maximum = 3” or “minimum = 5”.

107. The number of statistical measures is large and no general agreement exists on a small subset to fit all purposes for which descriptive information models are intended. The various existing denormalized models all use different measures. A flexible model able to store a larger number of different statistical measures is desirable.
108. The statistical measure model should be extensible and offer generalizations that allow applications to support classes of statistical measures, rather than only individual measures.
109. The fundamental applicability of statistical measures depends only on the data type (measurement scale and continuous/discrete), but not on individual characters. As a consequence, no schema evolution issues exist when new statistical measures are added to the terminology.
110. The designer of the terminology should be able to limit the statistical methods available to data entry personnel. This leads to more concise data entry forms and can reduce errors.
111. Statistical measures are fundamentally applicable to all characters. If no data storage problems prevent this, it may be desirable to view the limitation of measures as a “recommendation” or “secondary filter”, affecting only the primary data entry form rather than data storage.
112. No equivalent to explicit state-absent statements (which may be desirable in categorical data, see requirement 104) occurs with quantitative data.

Value order in character data

Whenever data are recorded based on a terminology, the sequence of recorded data elements may follow either the sequence defined in the terminology or the sequence of user data entry. For example, when entering fields into a database table, the sequence of data entry is not preserved, and reordering of fields will automatically reorder all data. On the other hand, if experimental data are entered into a spreadsheet-like editor, the user would expect that data entry sequence is honored. The following cases may be distinguished:

1. **Order of characters in a description.** A general agreement seems to exist that – for the sake of comparability – characters should be in the same order in all descriptions to be compared.

Different sequences for different purposes are often desirable, but all of these then apply to all descriptions. In DELTA only a single character order may be defined; in SDD order is expressed through Character Trees, of which an unlimited number may exist.

2. **Order of repeated measurements (original sample data).** Although the order of repeated measurements should never be semantically relevant, preserving this order is highly relevant for the sake of workflow, e. g., proofreading or other comparisons with external notes. See also “Data recording levels (sample data)” (p. 89) and Table 20 (p. 91).
3. **Order of values within a quantitative character.** Similar to characters, the order of statistical measures is important for comparison purposes and normally defined exclusively in the terminology or report-generation methods. However, an important exception is that some models, including DELTA, support repeated “central measure or values” (Table 31, p. 112), for which the same arguments as for original sample data apply. Thus, for repeated occurrences of the same measure – where the information model supports such – the original sequence may have to be preserved.
4. **Order of values within a categorical character.** This is perhaps the most contentious issue. Firstly, in some models it may not be recognizable, whether a sequence of states is intended as repeated measurements or as summary data. Distinguishing this should be a goal of the model (which may be in part achieved by preventing repeated state values, unless modifiers or notes differ). For the remaining states, in many cases the data set authors will desire them to be reordered if the sequence of states in the terminology is updated. In some cases, however, it is vital that a sequence in a given description shall be preserved. The remainder of this section will discuss this in detail.

In general, the order of categorical character states in the terminology may express an innate order of the concepts. This is simplest in the case for ordinal data, where the order expresses that the distance from first to third state is greater than any other distance among these states (even though it is unknown whether first-to-second, or second-to-third have a greater distance). However, nominal data may also be at least partially ordered, often in complex ways so that a deliberate choice has been made to simplify the analysis by reducing these data to the nominal scale. This and other reasons, including traditions, may lead to an ordering expectation by human consumers. Defining this order once in the character terminology (or in report definitions) for all descriptions together fulfills a minimal requirement.

In many cases, however, terminology-defined order and the desired order in a description differ. For example, in a character with the ordered states:

- 1. very short bristles
- 2. short bristles
- 3. long bristles
- 4. very long bristles

frequency or certainty modifiers (p. 206 and 207, respectively) may appear in a specific object description as “3 or rarely 1” or “4 or perhaps 3”. Clearly, although the “scoring sequence” may not matter in identification or character analysis, it greatly facilitates communication with human users, especially when generating natural language descriptions. A statement “bristles very short (rarely) or long” is considerably more difficult to understand; and “bristles rarely very short or long” is even likely to be misunderstood. Similar examples may be found for spatial or temporal modifiers (p. 203 and 204), e. g., “flower blue, or violet (at the base)” or “flower violet changing to red when mature”.

If the order of modifiers has been explicitly defined to be semantic (compare “Modifier sets and sequences”, p. 199), the modifier order may thus have to take precedence over the state order. Unfortunately, some expressiveness may still be beyond analysis, e. g., where special situations are annotated in free-form text. Again, further analysis of existing data sets is required to determine whether it is sufficient to place states with annotations last. The default order of states in summary data of descriptions could thus be:

- by ascending temporal modifier rank,
- by ascending spatial modifier rank,
- by descending certainty rank or values (certain before uncertain states),
- by descending frequency rank or values (frequent before infrequent states),
- states without notes before states with notes,
- by ascending order in which the states are defined in the character definition (for all modifier cases, character data without modifiers would be ordered first).

The minimum requirement for order of categorical states in descriptions is that the resulting reports and natural language descriptions should not hinder communication. Whether the rules above suffice for this and whether they can be extended to include all modifiers for which order may be defined as semantic, requires further analysis. In preliminary experiments of thought no counter-example of ranked modifiers could be found, which – if used to reorder state sequences – would not also improve the readability of the resulting description.

Despite this, many biologists may desire support for manual value ordering that goes beyond the algorithmic support outlined above. Biologists normally write descriptions as unconstrained text. Many biologists will assume that "round or obovate" and "obovate or round" are different, assuming an unequal (but unspecified) frequency to be implied in the order of states. This may be viewed as a "bad habit" in biology, but it may be unwise to try to force content providers to abandon relatively harmless bad habits. Furthermore, constraints on value ordering can only be enforced for future data; at least for legacy data (digitized descriptions, DELTA-coded data), ignoring value sequence is likely to cause some problems.

Individual applications may desire to support additional manual ordering methods or not. For data exchange standards and general information models designed to support multiple applications, some agreement on manual ordering must be sought. Unfortunately, to always preserve the value order has some drawbacks, if:

- the terminology is revised and the order of states in the terminology redefined,
- data are aggregated (e. g., descriptions of multiple specimens combined into a new species description) that have different scoring sequences.

Current support for value order in some applications and data standards:

- The *DELTA* and *New DELTA* standards defines that the order of states in data files is always significant.
- The *CSIRO DELTA* programs (p. 19) will always store the scoring sequence and ignore the sequence of states in the terminology. Separate methods for "reordering character states" based on the terminology exist.
- *NEXUS* (p. 18) applications probably preserve the sequence in which polymorphic state scores (in "{ }") are written, but definite information about this is not known. Mesquite (p. 18) still needs to be tested in this respect.
- The first versions of *DiversityDescriptions* used only the state sequence defined in the terminology and considered the sequence of states in descriptions irrelevant. This turned out to be a major source of discontent for users so that later versions added an optional manual state sequence. By default all states in a description are ordered in the sequence defined in the terminology, but if the user chooses to rearrange this sequence in a description, this new sequence will be permanently stored and is no longer influenced by changes in the terminology. If this decision is later considered undesirable in some characters, a method is provided to reset state order to terminology order for all descriptions in a single step. See the documentation of the logical model (p. 331) for further information.
- *CBIT Lucid* (p. 21) never outputs any descriptions where the state sequence may be relevant. It is optimized for interactive identification, in which case the state sequence is irrelevant.
- *SDD* (p. 20) the "statemodel" xml-attribute within the character data element may be changed from the default ("OrSet") to other values like "OrSeq" (compare Table 22, p. 97). *SDD* thus

prefers no manual ordering (therewith simplifying evolution of terminology, like adding or re-ordering states), but allows this to be changed where explicitly desired.

The solution chosen by *DiversityDescriptions* and *SDD* has the advantage that normally changes in the terminology are dynamically and automatically reflected in reports (even in federated databases where the terminology may be changed independently of the descriptions). However, it makes data entry slightly less intuitive, since it requires an explicit understanding and choice between default and manual state order. This is especially problematic if during the part of the recording phase the desired scoring sequence in a given description is identical to the sequence in the terminology, but the latter is later rearranged.

The topic has previously been discussed in an *SDD* proposal (Hagedorn 2003g).

113. A general order of characters and a general order of character states within a character are meaningful for communication with humans, even where it is not meaningful for machine interpretation or analysis (e. g., states on the nominal scale).
114. For characters, multiple alternative ordering definitions are desirable.
115. Negative requirement: It is not necessary to preserve, in a given description, the order in which data relating to different characters have been entered.
116. In a given description and character, the order in which multiple values or states have been entered may have to be preserved. This is unequivocal for repeated measurements in sample data, but restricted to special situations in summary data.
117. In a given description and quantitative character, multiple occurrences of a statistical measure may have to be preserved in sequence (some models use this as a replacement for sample data).
118. In a given description and categorical character, it may be desirable to provide a method to let data set authors decide whether the sequence of multiple states may be rearranged according to the sequence in the terminology, or whether it is to be preserved.
119. When reordering the states in a given description and character, modifiers for which order has been defined as semantic (ranked modifiers) may have precedence over the state order.
120. It is desirable that the information model encourages distinguishing sample data and summary data in an unambiguous way, e. g., by preventing unqualified repeated occurrences of the same value or state in summary data. States with different modifiers or annotations, however, have to be accepted.

Character decomposition models

The majority of descriptive data applications in biology are interested in individual objects or classes of these. In the matrix- or list-based description models discussed so far these specimens, units, taxa, items, etc. are considered a fundamental entity that is described through a list of variables, called characters. These models have a number of advantages. They are close to:

- the entity-type/attribute/value model in ER modeling,
- the table/field/value model in the physical view of a database,
- the class/property/value model in many object-oriented programming languages,
- the class/attribute/value model in UML (compare also Table 3, p. 34).

Object-oriented information models generally support complex object properties composed of the same (array, vector, matrix, collections) or different (structure, record) types, making it relatively straightforward to consider the character variables as object properties. This is slightly less intuitive in traditional relational databases, where the entity attributes are often limited to simple data types (including strings). However, as the *DiversityDescriptions* model (p. 322) shows, an entity \times character model can easily be implemented in a simple relational DBMS, without requiring an object-relational DBMS. An unstructured list of variables is the prevalent data format in phyloge-

netic or multivariate statistical analysis, and is used in the dominant DELTA or NEXUS exchange standards for descriptive data.

These advantages of the simple character model are offset by a number of problems that are experienced when defining and applying characters. Fundamentally, the definition of character variables is a complex and difficult task. Characters should be independent (or as independent as possible). Many characters depend on specific circumstances (taxonomic or other scope, object part, instrumentation, measurement methods, etc.). Defining characters in a manner that data providers and consumers communicate successfully with each other is a serious problem. It is not uncommon that even the creators of a terminology start introducing duplicate (or near duplicate) characters when a large terminology contains over 1000 characters.

In an attempt to solve these problems, a number of proposals have been made to “decompose” the characters into more fundamental data items and base the information model on these items. To the author’s knowledge, the first explicit application that not only conceptually analyzed characters as part-plus-property, but also explicitly stored them as such, is Taylor (1995). However, Taylor is primarily interested in automatically parsing large bodies of natural language descriptions, and only briefly describes this approach (and the need for relational characters, see below). Two other projects have subsequently analyzed and tested the character decomposition approach in greater detail; these will be discussed in the following.

The Nemisys/Genisys model

Despite problems with the identification of object parts (Fig. 10, p. 38), object parts are a central concept in morphological or anatomical data. The majority of morpho-anatomical characters may be interpreted as a limited number of abstract observation methods applied to either the entire organism or a large number of object-parts (including regions and functionally defined “organs”). Building on earlier studies by Lebbe (1991), Diederich, Fortuner and Milton in a series of articles (see Nemisys/Genisys model, p. 21) proposed a descriptive information model which decomposed characters into “structures” (i. e., “parts” or “physical components”) and “properties”.

In this model characters are the intersection of two more or less hierarchically organized dimensions: object parts and basic properties (a concept they introduce, combining instrumentation, selected properties and methods, with data types, see “Basic property types”, p. 62). The authors explicitly recognize that the model is optimized for morpho-anatomical data, but maintain that it is also useful for all other forms of descriptive data (e. g., physiological data).

Some publications on the Nemisys/Genisys model may be interpreted as a set of rules to restructure and reorganize an existing character list. Diederich & al. (1998) mention their recognition of 272 parts and over 1000 characters, and that the potential number of characters of 272 parts \times 20 basic properties could grow into more than 5000 characters. This suggests that the entity “character” remained a useful concept under their model. Some of their proposals may best be interpreted as an analytical tool to organize characters in a pattern that increases the manageability of the terminology and that does not affect data storage management.

On the other hand, Diederich (1997) outlines a new data storage model where the combination of object part and basic property is no longer under terminological control and where part and property concepts may be combined freely at data recording time (Table 32). In addition, they introduce a concept called “name extension” that allows ad-hoc modifications of both part and property concepts. This model might perhaps look like Table 33. No field is mentioned in Diederich (1997) to store the object parts of relational basic properties; a column has been added for this in Table 33. Further, the model contains extension mechanisms: a) “Name extension” for the basic property (although often object parts are involved in the extensions) and b) “Qualifier” for states. Both mechanisms are closely related to the mechanisms discussed in “Modifiers” (p. 189). Note that to directly support any kind of ratios in a fully decomposed model, further columns may have to be added. For example, in an insect a ratio value may be calculated as the distance (= “property 1”) between the attachment point of front legs (= “part 1”) at the body (= “part 2”)

Table 32. Fundamental part/property model.

Entity (Taxon)	Object-part (Structure)	Basic property	State
1	Body	"kind" (color)	brown
1	Body	shape	ellipsoid
1	Head	"kind" (color)	dark red
1	Head	shape	round

and the attachment point of the middle legs (= "part 3") at the body (= "part 2"), divided by the length (= "property 2") of the segment of the front leg nearest to the body (i. e., front femur, = "part 4"). Clearly, this is a constructed example, but similar characters are not totally unrealistic because ratio estimates of immediately neighboring part lengths are relatively conveniently done

without precise measurements or calculations. An actual example is whether length of the hind-leg of a frog is longer than the distance from hind-leg attachment to the nose of the frog.

Regardless of the details of the model, an important aspect of a property/object-part decomposition is that it is relational (i. e. two-dimensional) rather than hierarchically nested. For example, if during identification a compositional part of a biological object (e. g., a flower) can already be recognized, it will often be best to study multiple properties grouped by object part. If, on the other hand, the parts are difficult to distinguish, but an intuitively recognizable property concept is found, it may be more useful to list characters grouped by property. For example, in a fungal colony in a Petri dish it may be impossible to distinguish which hyphal structures are responsible for the color effect, but the color itself is readily observed.

In the above example, if the compositional hierarchy is a kind-of hierarchy (i. e., a generalization), properties could be generalized. However, the examples in Diederich & al. (1997) rather suggest a part-of hierarchy of "structures and substructure". This topic is further discussed in "Composition versus generalization" (p. 153).

The Nemisys/Genisys model introduces valuable new approaches to descriptive data. However, it seems to be optimized for a particular form of morphological data (compare also requirement points 31 ff, p. 66). It is unclear to which extent it has actually been implemented; no formal documentation of an information model could be found.

Table 33. An example based on the detailed Nemisys/Genisys model (including the "name extension" and "qualifier" proposals).

Entity (Taxon)	Object-part (Structure)	Basic property	Related part (Structure)	"Name extension"	State	Qualifier
1	Hemizonid	position relative to	excretory pore	-	anterior	slightly
1	Body	"kind" (color)	-	at excretory pore	brown	-
1	Eye	Distance to	Antenna	-	touching	-

Notes: After Diederich (1997), where the model is outlined and discussed, but not shown in exactly this form. Here two columns are added: an ID-reference column for the entity, and a column to express the second object-part (structure) discussed for relational basic properties. The discussed version column is not shown here. The first example is from Diederich, the other added.

The Prometheus description model

As mentioned, the Nemisys/Genisys model is a conceptual model that is only partly documented and the concepts of which are evolving from publication to publication. The Prometheus description model (p. 21) is the second character decomposition model described so far and develops a fully functional model. As discussed on p. 31, Prometheus replaces the term "character" with "description element", partly perhaps to stress the character decomposition-model. In line with the remainder of this thesis and to facilitate the comparability with the Nemisys/Genisys model, this is not accepted and the term "character" is maintained here.

The Prometheus model differs from Nemisys/Genisys in several respects, for example:

- Generally much more focus is placed on defining the terminology. Where in the Nemisys/Genisys model it is occasionally unclear whether ad-hoc natural language definitions are supported or even intended, Prometheus is unambiguously clear to support only defined terms.
- A strict distinction is made between quantitative and qualitative (i. e. categorical) properties.
 - “Quantitative properties” are a subclass of “Defined Terms”, may not be hierarchically arranged, and may not be constrained to the context of specific object components (“structures”).
 - “Qualitative properties” (term used in Fig. 1 and 4 of Pullan & al. 2005) or “State groups” (term used in text of Pullan & al. 2005) are no “Defined Terms”, may be hierarchically arranged, and may be constrained to specific object components (“structures”).
 - The data recording process follows this distinction. Whereas quantitative properties are explicitly recorded together with the value, for categorical data only the character states are recorded; the qualitative properties are implied.
- Structural models (using part-of relations) are used both in the terminology (potential composition, general constraints), and in description instances (actual composition). This mechanism is used for several purposes:
 - It supports and constrains data recording;
 - it replaces specialized plant structure terminology, but building specializations (leaves on stem, in inflorescence) in an ad-hoc mode, constructing “structural paths”;
 - it provides the containers for part-specific measurement or aggregation data (compare “Boolean operators between characters”, p. 98).
- The problem of spatial areas or regions like tip, bottom, or center is addressed by introducing a subclass of structure (“Region”) which is freely combinable with structure. A similar subclass “Generic Structure” is created for parts that occur inconveniently frequently on multiple other parts (e. g. hairs). Again, instances of Generic Structure may be used on any part, without relations being present in the terminology.
- The explosion of codable points that is a result of freely combinable part and property terminology is reduced, first by placing constraints on state terms to which object components they are applicable, and secondly by allowing project managers to create so-called “pro-forma” definitions (i. e., something not essential, but only for form's sake). The “pro-forma” mechanism seems to be related to database views, creating restricted subsets of the entire terminology, but may entail other, more complex setup information as well.
- The concept of modifiers is significantly enhanced. This is discussed separately, see p. 196. The hierarchical arrangement of properties addresses many of the issues criticized for the “Basic property types” (p. 62) of the Nemisys/Genisys model. Also, the qualitative property/state group model is freely extensible, avoiding the need for artificial catch-all properties like “kind”. However, it remains unclear why quantitative and qualitative properties may not both be hierarchical and both be defined terms (or “concepts”). It would be desirable to create a hierarchy for size measurements (e. g., length, width, length including bristle, excluding bristle, etc.) and probably other properties as well. Furthermore, in the light that quantitative measurements may be expressed both as continuous and categorical measurements, and that mappings (p. 66) between these may be defined, it seems unfortunate not to be able to browse data using a property hierarchy irrespective of the data type used (including the use of complex data types, p. 59, e. g., for color).

The direct recording of character states without going through a hierarchical level of properties is certainly a very interesting feature. However, it seems to be truly a question of the user interface, not requiring changes to the information model (see p. 129). Indeed it may be noted that the assumption that the property is implied by the states holds only for data recorded using unique identifiers. In natural language categorical states may be ambiguous (“hot” in an animal is likely to be temperature, in a fungus likely to be taste). Conversely, for qualitative data the property is often implied in the measurement unit (e. g., ‘g’ or ‘°C’ have implicit properties “weight” and “temperature”).

The introduction of “structural paths” seems to be a generalization of the two-level (structure and substructure) storage model described in some versions of the Nemisys/Genisys model. Structural paths simplify the structural terminology by avoiding the need for terms like “ground leaves”, “stem leaves”, etc. At the same time, they remove the possibility to give these parts a name. This seems to be unfortunate, since the question whether a part has a separate name is language- and culture-dependent as the example of German and English shows. In German, it would be logical to construct both petiole (leaf stalk) and pedicel (flower stalk) as a structural path, because German botanical language has no special terms for these. It is questionable whether this is desirable to an English botanist. Moreover, not only bracts (i. e., “leaves with single flower growing in axil”), but also sepals, petals, etc. are in fact modified leaves that could be expressed using structural paths rather than specialized terms.

Structural paths come at the expense of a complication of the storage model, requiring some means to store a path of unlimited length, and be able to both search for the exact path (e. g., only “upper-stem-leaves”) as well as for generalized concepts (e. g., “any kind of leaf”). It is unclear, whether Prometheus stores the entire path in each in the description, or whether an anonymous specialized concept is created for each path, which is then referenced by a system identifier.

In general, Prometheus seems to deduce some of its requirements from particular features of the English language, e. g., when requiring that “state terms” may not be used as part of structure terms, and that any term may at most be “coded using one or two words” (Pullan & al. 2005). Such rules need some generalization to make them compatible with languages that prefer derived nouns over adjective-noun clauses or require more than two words to express a single concept. Even in English it is doubtful whether these rules indeed guarantee that “data can only be coded one way, even when entered by different authors” (Pullan & al. 2005).

If for each description an actual “structural path” is created that is based on first class object parts (i. e., structural terms for which compositional constraints exist in the terminology), then this is easily extended by adding elements for which no such constraints are defined in a similar fashion. These are the “Generalized Structure” and “Region” terms introduced by Prometheus. Of these, the regions are truly general (compare “Absolute object orientation” and “Relative object orientation in compositions”, p. 147). The concept of regions seems to be closely related to spatial modifiers (which also exist in Prometheus, see p. 196) and it remains open why two separate mechanisms are required. The mechanism of “Generalized structures” is less a logical requirement than a convenience mechanism. Clearly, part like “hairs” are not truly applicable to all part of an organism, especially not if the compositional hierarchy includes anatomical parts. However, it remains at the discretion of the builder of an ontology, whether the mechanism is used or not.

The pro-forma mechanism is further discussed on p. 127 (compare also Figs. 39-40).

An essential feature of the Prometheus model is that it tries to reform the way taxonomy and descriptions are performed. Although other models allow recording of individuals (including DELTA, and with increasing support DiversityDescriptions and SDD), Prometheus goes to the extent of considering abstract taxon descriptions as a set of “virtual specimens”, thus encouraging to record actual specimen data instead. Similarly, biological terminology may only be used if it fits the assumptions of the structure + property/state group model. In some cases a decomposition of characters into parts and properties requires reformulating biological terminology and organizing knowledge differently, especially where functional concepts are used as organizing principles. This may be less convenient during identification because it corresponds less well with expectations of the identifying person, but it may lead to more consistent use of terminology (Table 34).

Clearly, such an approach has advantages, but it is yet unclear whether it provides the flexibility that biologists desire for their work. The Prometheus authors themselves refer to extensive testing that is required. Results of this have not yet been published.

Table 34. Examples of conventional characters that are difficult to decompose into property and object part (Lucid, left) and proposals how they might be handled in Prometheus.

Lucid:		Prometheus description model:			(Notes by T. Paterson; edited G. Hagedorn)
Character	States	Object part ("Structure")	Property/ StateGroup	States	
Salt tolerance	<ul style="list-style-type: none"> plants tolerating high salt levels (halophytes) plants not salt tolerant 	Entire Plant	Ecological adaptations	halophytic	<i>(list of alternatives, or "not")</i>
General habit	<ul style="list-style-type: none"> tree shrub climber (woody or herbaceous) herb grass- or sedge-like 	Entire Plant	Habit	tree, shrub, herb, etc.	<i>(more specific data may be collected by scoring more states for additional properties)</i>
		Entire Plant	Architecture	climbing, bushy, creeper, twining etc.	
Epiphytic or lithophytic habit	<ul style="list-style-type: none"> plants growing in soil (not epiphytic or lithophytic) plants growing on other plants or on bare rock surfaces (epiphytic or lithophytic) 	Entire Plant	Preferred substrate	epiphytic, aquatic, lithophytic, terrestrial	
Habit (aquatic herbs only)	<ul style="list-style-type: none"> free-floating rooted in substrate with leaves mostly submerged rooted in substrate with leaves mostly floating on the water surface rooted in substrate with leaves mostly emergent above the water surface 	Root	Root attachment	free-floating, substrate-attached	<i>(appropriate terms for these states not yet present in the Prometheus ontology – but could be added)</i>
		Leaf	Aquatic position	floating, submerged, emergent	
Seasonal longevity	<ul style="list-style-type: none"> annual, biennial, or ephemeral perennial 	Entire Plant	Lifespan	annual, biennial, ephemeral, perennial	
Leaves (woody plants)	<ul style="list-style-type: none"> evergreen deciduous or semi-deciduous 	Leaf	Lifespan	deciduous, semi-deciduous, evergreen	
Structures for spreading vegetatively	<ul style="list-style-type: none"> none (plants not spreading vegetatively) underground bulbs, corms or tubers etc rhizomes, stolons or root-suckers detached aerial stem parts, or proliferous flower heads 	Entire Plant	Reproduction	vegetative	<i>(list of alternatives, or "not")</i>
		Bulb	Presence	present, absent	
		Corm	Presence	present, absent	
		Tuber	Presence	present, absent	
		Rhizome	Presence	present, absent	
		Stolon	Presence	present, absent	
		Root-sucker	Presence	present, absent	
		Detached aerial stem parts	Presence	present, absent	
		Inflorescence	Type	proliferous	
Chlorophyll in stems or leaves	<ul style="list-style-type: none"> present (plants green or grey-green) absent (plants colorless, white or yellowish) 	Leaf-chlorophyll	Presence	present, absent	<i>(uses structural hierarchy to identify which chlorophyll)</i>
		Stem-chlorophyll	Presence	present, absent	
		Entire Plant	Color	(list of colors)	
Nutritional strategy	<ul style="list-style-type: none"> neither carnivorous nor parasitic (normal plants) partially or totally parasitic on other plants carnivorous 	Entire Plant	Habit-Lifestyle	carnivorous, parasite, partial parasite, etc	<i>(any combination of states including "not")</i>
Trap structures (carnivorous plants only)	<ul style="list-style-type: none"> submerged or underground bladders pitcher-traps sticky glands or glandular hairs on leaves/stems trap like irritable leaf blade segments 				<i>(appropriate terms for these states not yet present in the Prometheus ontology – but could be added)</i>

Notes on Table on previous page: Examples based on public postings to tdwg-sdd@listserv.nhm.ku.edu on 2004-03-17; available in TDWG-SDD list archive. Left-hand columns based on the Lucid key "The Families of Flowering Plants of Australia", provided by Kevin Thiele; right-hand columns based on reply by Trevor Paterson.

121. Summary statement: The Prometheus description model has very special requirements on the information model. It elaborates and modifies the concepts of the Nemisys/Genisys model. It is implemented and tested. The extent to which this model is specific to certain kinds of data needs to be assessed as experience with the model grows.
122. Summary statement: The Prometheus description model provides for the definition of a subset of all possible object-part/property combinations for data entry. For different projects, different sets of "enabled" object-part/property combinations may be defined. The union of all enabled selections is roughly equivalent to characters in character or character state matrix models.

Relational characters revisited

As mentioned in the discussion of the character decomposition models, a number of characters typically used in biological descriptions have a different structure than part + property + value. Taylor (1995) was the first to introduce special data structures for "relational characters", and both Nemisys/Genisys and Prometheus are addressing the problem. The following cases may be distinguished (Table 35):

Table 35. Cases that may be termed a "relational character".

	Situation	Examples depending on two parts	Examples depending on two properties
1	A single measurement is by necessity dependent on two object parts/properties	Distances or angles-between two parts	(no example found)
2	A measurement primarily depends on one part/property but is modified by an additional specification	Body width at excretory pore; spore width at septum	Separate measurement of the width of septate/aseptate (or hyaline/brown) spores (compare Table 13, p. 73)
3	Multiple measurement may be made independently and then used to create another character	Relative size of sepals and petals, comparative weight or lightness/darkness of object parts ("frontal area lighter than surrounding")	length/width ratio, relative concentration of two chemicals in a part (chlorophyll a/ b)

The first row in Table 35 represents the most typical "relational characters". The second case may either be treated as relational characters, or handled through modifiers (p. 189). The third case is in principle a calculated character (see p. 72). It is perceived as a relational character, if the direct recording of the "calculable" result is more typical than the recording of both original and calculated data. For example, length/width ratio are more readily perceived as a calculated character (length and width being typically also measured separately) than "sepals to petal length" expressed categorically as "sepals shorter", "same size", "sepals longer".

Not shown in the table is perhaps the most critical case of what might be considered a "relational character" (but which the cited decomposition models do not consider so): characters indicating the presence or multiplicity of object parts that may be present on multiple other parts (hairs, leaves, etc.). In the case of "number of basal leaves", "number of stem leaves", "number of leaves in inflorescence (bracts)" one may tend to prefer to express this by creating specific subtypes of leaves (basal leaves, stem leaves, bracts). However in a case like "number of hairs on coxa", "number of hairs on femur", or "... on profemur", "... on mesofemur" it seems quite natural to consider the character being dependent on the two object parts, the relation of which it describes.

Integrating these cases into character decomposition models requires additional structures (compare Table 10, p. 64 and Table 33, p. 118 for Nemisys/Genisys).

The Prometheus description model (p. 21) uses a data structure for relational characters called “relative modifiers”. On p. 196, this mechanism is described. It is criticized here for being considered a modifier, but otherwise it does support characters depending on exactly two parts (similarly to those in Nemisys/Genisys), with the added advantage of supporting operators (equal, smaller, greater, etc.).

The topic of relational characters urgently needs further study. It is clear that it is a potential source of much complication in models, and a simple, flexible solution that allows to express various forms of relations is preferable. Whether the model of concept hierarchies discussed further down (p. 125) provides sufficient support is currently unknown.

123. A character may depend on more than one object part or on more than one property. A possibility to express this, either in descriptive terminology or in descriptive data, is desirable.
124. Some “relational characters” may be viewed as calculated characters. The multiple parts involved may then simply be discovered by analyzing the characters involved in calculations. However, often only values for the calculated, but not the base characters are available. The model should thus support analysis of multiple part-relations even if only the calculated character values are present.

Multidimensional character decomposition

The models discussed so far propose to decompose all characters (i. e., “defined variables for descriptive data”) into an object part and a property. This decomposition is only a subset of all potential decompositions. It is helpful to consider the steps that are involved in the process of recording a character value for a physical object (Table 36). The dominant dimensions influencing the definition of a character are:

- **Object constraints.** A character may be specific to a taxonomic group, a temporal period, a sex or life cycle stage (see p. 218 and 217 in the section “Secondary classification resulting in description scopes”), conditions of observation (see also “Dependencies on circumstances of identification”, p. 175), experimental conditions, preparation status, etc. *Examples:* “winter”, “larval characters”, “fresh material”.
- **Object part** (also called “*structure*” in other models). A character applies either to the entire object or to a part of it. The part(s) to which a character refers may be part of the character label (e. g., “flower color”) or may be implied (e. g., “stamen number”, referring to “stamens per flower” and thus a property of flower). If no specific part is mentioned (or implied through the property), one will assume that a character refers to the entire object (the entire organism). *Examples:* inflorescence, flower, anthers, pollen-sac, pollen, cell wall.
- **Measurable concept** (= “property” + fundamental measurement method). All characters refer to a property (such as color or shape) and a method to measure it. Measurable concepts are thought here as being defined without regard to specific circumstances depending on taxonomic group, object part, or instrumentation. *Examples:* color, shape, length, aggressivity, speed, presence of object parts.
- **Measurement method (operating instructions).** Where measurements are difficult, or a high degree of scientific accuracy is called for, measurements may be governed by specific measurement operating instructions. Instructions may be required even in relatively simple cases, compare the example of Figs. 24-25, p. 72). In many analytical situations, detailed and documented operating instructions are even required to make results admissible as evidence at a legal court (e. g., if the ownership of a plant breeding variety is disputed). An example for a model decomposing methods is Germeier & Frese (2001).

- **Instrumentation.** This designates tools and associated procedures that are required to obtain values for measurable concepts. It is convenient to include human senses (i. e, the absence of technical instrumentation) in this concept. *Examples:* “span of hand”/“cm ruler”; “unaided human vision”/“hand-lens”/“light microscope”/“scanning electron microscope”; or “human sense of smell”/“chemical gas chromatographic analysis”. In current descriptive terminologies instrumentation is often implied, especially where instrumentation is simply the unaided human senses. However, instrumentation does not necessarily default to these, e. g., “fungal spore size” will default to “light microscope with micrometer scale”. Occasionally considerable experience with a taxonomic group or property is required to understand an implied instrumentation correctly, and it is highly desirable to be able to make this information explicit in descriptive terminology.
- **Information representation (data type).** After measuring (which may include several layers of primary data transformations), secondary transformations may be applied to the character value to conform with the format required by the data storage method. This includes conversions from complex data types or quantitative values into categorical data.

Decomposing a character definition into these dimensions will often allow a reuse of relatively few fundamental definitions. At the same time, the dimensions provide a classification (and potentially a hierarchy) that helps in achieving manageable and well structured character sets.

Table 36. Character decomposition based on the steps required to observe and record a value for a character.

Step	Comment
Choose the object	Seemingly trivial, but the object may not belong to a class that can be described by the methods defined in the terminology at all. In identifications this must always be considered as a fundamental error source. Also, often temporal, geographic, or other constraints (such as life cycle stage or sex) must be observed to correctly observe a character value.
Optionally choose a part of the object	This is highly problematic insofar as no straightforward method to do this exists. The process is heuristic in that “sub-identifications” using general part-characteristics are required to name and choose the correct part
Apply the abstract observation method	To do this, one may need to follow both very general measurement instructions (especially use of instrumentation like hand-lens or microscope), as well as taxon- or part-specific instructions (Examples: Is “length” defined along an anatomical axis or is it always the larger of two perpendicular measurements? Are appendages to be included or not?)
Convert values into the form required by the information model	Measurement may be in different units (cm, mm), color may be measured spectrographically but requested as color-space values (e. g., sRGB)
Apply the data entry methods of an application implementing the model	Trivial when humans use an existing user interface, but especially in the case of automated data recording this needs consideration.

In many small objects, issues of observation and experimental conditions, operating instructions, and instrumentation are much more relevant than might be assumed from discussions based on examples from vascular plants or insects, where clearly the issue of object part hierarchy is dominant and where “field characteristic” observable by the unaided eye are often in abundance.

Similarly, a temporal (seasonal, developmental) decomposition is usually very relevant in biological objects (discussed later in detail on p. 162). Nobody so far seems to have proposed to extend the property/structure decomposition model by Diederich & al. (1997) by adding instrumentation/method and time/development as further dimensions to a character decomposition. In many taxonomic groups this seems to be a welcome addition.

Like object-parts and properties, methods are often hierarchically structured. Occasionally the structure may even be best viewed as independent parameters: Table 37 shows an example where the method itself is broken down into parameters. Doing so in a general descriptive information

model would probably overcomplicate the model; a method hierarchy instead of dimensions would probably fulfill most requirements.

Table 37. Examples of related characters that are distinguished by three parameters of a single method used.

Character	Object	Basic Property	“Basic” Method	— “Method Parameters” —		
				medium	temp.	duration
1. Growth rate (OA, 20 °C, 7 d)	Culture	Size	Growth/Petri dish	Oat Agar	20 °C	7 days
2. Growth rate (OA, 30 °C, 7 d)	Culture	Size	Growth/Petri dish	Oat Agar	30 °C	7 days
3. Growth rate (OA, 20 °C, 14 d)	Culture	Size	Growth/Petri dish	Oat Agar	20 °C	14 days
4. Growth rate (OA, 30 °C, 14 d)	Culture	Size	Growth/Petri dish	Oat Agar	30 °C	14 days
5. Growth rate (MA, 20 °C, 7 d)	Culture	Size	Growth/Petri dish	Malt Agar	20 °C	7 days
6. Growth rate (MA, 30 °C, 7 d)	Culture	Size	Growth/Petri dish	Malt Agar	30 °C	7 days
7. Growth rate (MA, 20 °C, 14 d)	Culture	Size	Growth/Petri dish	Malt Agar	20 °C	14 days
8. Growth rate (MA, 30 °C, 14 d)	Culture	Size	Growth/Petri dish	Malt Agar	30 °C	14 days
9. Color (OA, 20 °C, 7 d)	Culture	Color	Growth/Petri dish	Oat Agar	20 °C	7 days
10. Color (OA, 30 °C, 7 d)	Culture	Color	Growth/Petri dish	Oat Agar	30 °C	7 days
... etc.						

Method “Growth/Petri dish” = “Growth rate measurement in mm, cultivated in 90 mm Petri dish under conditions given in parameters”.

125. For many taxonomic groups the character decompositions beyond object part and property are desirable. Examples are experimental conditions, measurements methods and instrumentation, and information representation (e. g., quantitative versus categorical representations).

Concept hierarchies

The desire to decompose characters not only into an object-part and a property hierarchy, but also into multiple other dimensions (conditions, measurements methods and instrumentation, information representation), obviously greatly complicates a character decomposition. Alternative methods of expressing this information in secondary hierarchies may be considered. However, one may also consider whether the character decompositions proposed in Nemisys/Genisys (p. 21) and Prometheus description model (p. 21) may not be represented differently as well.

The model proposed here (and currently tested in SDD, p. 20) is to maintain the concept of a character as an independent basic entity that is defined in the terminology, but provide for multiple concept hierarchies superimposed upon the character list (Fig. 37). One concept hierarchy would be the part-of hierarchy proposed in decomposition models, another hierarchy would represent the properties (Nemisys/Genisys properties are proposed as a flat list, but (a) is this only a special case of a hierarchy, and (b) deeper property hierarchies may be desirable, compare Table 10, p. 64). Relations between concepts define the concept hierarchy, and relations between characters and concepts define the object part, property, method, etc. pertaining to the definition of a character (Fig. 38). In this model, extensions to further concept hierarchies can be made without redesign or a special effort, using established structures in the information model.

Concept hierarchies are related to general character hierarchizations (as in DELTA or DiversityDescriptions) or specialized ones. For example, Dmitriev (2007) associates characters with a specialized part ontology (“Characters.Morph” referring to extensible definitions in table Morph) and a fixed organism stage hierarchy (“Characters.Type” with system-defined values “ n = nymphs, m = males, f = females”; allowing combinations such as “ mf ”).

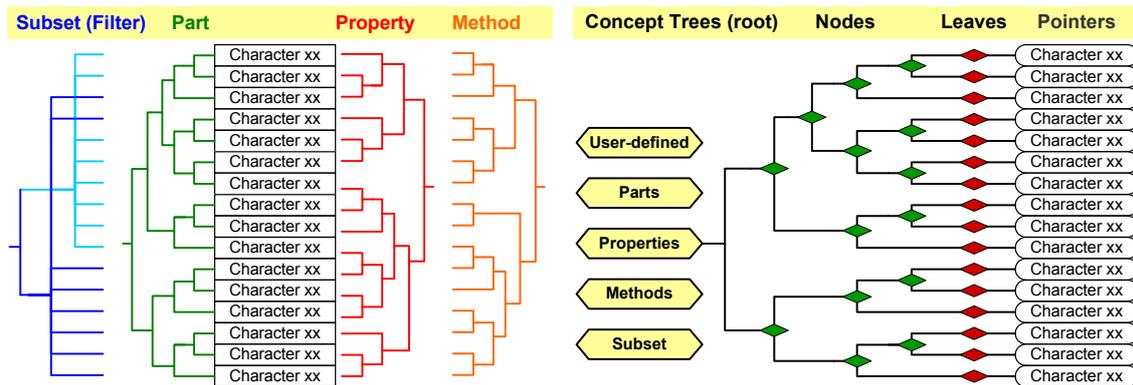


Figure 37. A flat character list may be organized under different aspects through multiple concept hierarchies.

Figure 38. Multiple concept hierarchies (trees) may be defined for a character list. In SDD, each tree (here “Properties”) contains inner concept nodes and terminal character nodes (also called “leaves” in informatics, each pointing to a single character).

One advantage of this approach is that in principle more than one part-hierarchy may exist. Although standard concept hierarchies will be desirable in most cases, this may occasionally be desirable:

- In large, heterogeneous taxonomic groups, homologous structures may be named or arranged differently in different subgroups (compare section “Problems with specialized, context-dependent names for object parts”, p. 157).
- A descriptive data set may have been developed under one concept hierarchy, but shall now be migrated to a larger data set with slightly different concepts.

The concept trees mentioned so far express truly semantic knowledge about characters and may be the basis for ontological reasoning. Concept trees may, however, also simply be used to express semantically opaque concepts in the form of user-defined trees or subset/filter definitions. Examples are the equivalent of headings and subheadings in descriptions (supported, e. g., by DELTA, DiversityDescriptions, or CBIT Lucid3). Such an arrangement may be any mixture of part, property, method concepts, or even yet completely different concepts like “importance for identification”. Similarly, concept hierarchies may be used to express the character subset definitions (present in DELTA or DiversityDescriptions). Subsets (compare Fig. 226, p. 334) act like views in a database, selecting a subset of characters for a particular audience, applicable under particular conditions (e. g., seasonal or geographical restrictions), or those characters applicable to a given taxonomic group (for large data sets, encompassing diverse groups).

The relation between characters and concepts may be differently constrained. In character decomposition models like Nemisys/Genisys or Prometheus exactly one property and object part must be defined for each recordable data value (but property or part-terms without recorded data may exist). One consequence of this is that existing data sets available in DELTA, NEXUS, or CBIT “LIF” formats cannot be supported. The SDD model therefore considers concepts on characters an optional feature, i. e., characters may exist without a defined property/part decomposition. Furthermore, SDD concept trees allow that a character is placed multiple times in a single concept tree. This feature enables appropriate handling of characters that are related to multiple object parts, e. g., where a measure expresses a distance between two parts. Treating such characters in a strict object decomposition system requires arbitrary decisions.

Interestingly, it is quite possible to handle data based on a character decomposition model in the character-plus-concept model. The concept hierarchies can be directly imported and characters could be created automatically wherever data are encountered (data types must be defined

under both kinds of models). The most problematic item is a label (or “name”) for these new characters. However, combining the labels for object part, position, and property will always create an intelligible (even if not always conventional and intuitive) character label.

To better understand the relation between character decomposition models and the concept hierarchy model presented here, one may visualize the two dimensions of the character decomposition models as spanning a matrix that is sparsely filled (Fig. 39). This illustration could equally well be used for SDD and the Prometheus character decomposition model. In SDD the black circles represent the actual characters, in Prometheus they represent so called “pro-forma” definitions (Pullan & al. 2005) of those potential data entry points, for which actual data entry is requested. The central difference is that in SDD data storage is based on the black dots (characters), which may even exist independently from the matrix, whereas in Prometheus data is potentially possible for any cell of the matrix (i. e., for any combination of a property and part definition). The black dots are then the result of a secondary selection process to create a view. Views can be defined in both models: in SDD it is another concept tree, in Prometheus it is the result of a separate selection mechanism (multiple “pro-forma”, compare Fig. 40).

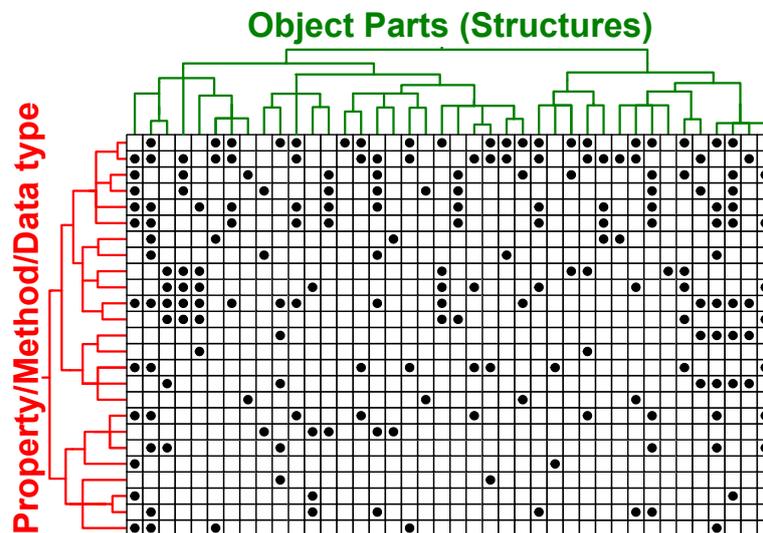


Figure 39. A sparsely filled matrix based on 40 object-parts (top) × 25 observed properties (left). 200 out of the 1000 potential combinations selected for data entry are shown as a black circle.

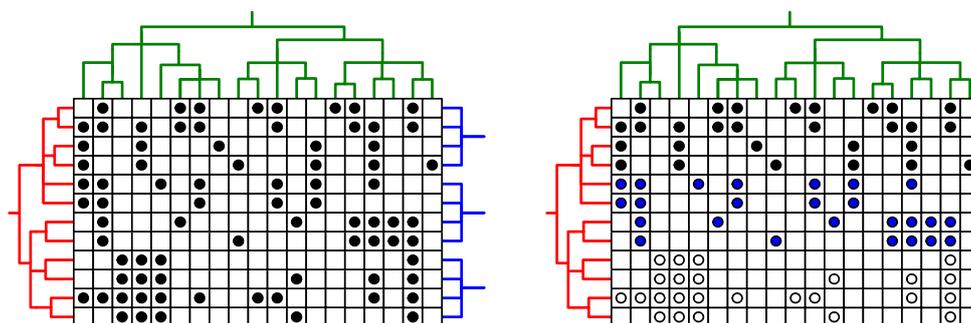


Figure 40. In the SDD model (left), views (for data entry, report generation, sorting, etc.) may be added as another concept tree (right side of left diagram). In the Prometheus model (right), a separate selection mechanism (called “pro-forma”) is used multiple times (represented here through differently shaded dots, and combined into a single diagram). To simplify the illustration, only non-overlapping views that are congruent with part and property hierarchies are shown. However, this is not a constraint of the model.

Table 38. Comparison of the character decomposition and concept hierarchy models.

Issue	Decomposition model (e. g., Prometheus description model)	Concept hierarchy + character model (e. g., similar to SDD)
Multiple compositional (i. e., object-part hierarchies)	One hierarchy is central to the model and the basis for data storage or retrieval. This hierarchy must be centrally managed and accepted by all participants in a federation. Additional secondary hierarchies based on different data structures may be added; only these can be federated.	All hierarchies are symmetric using the same structures. None is relevant for data storage. In a federated model, different participants may choose different hierarchies.
Multiple property hierarchies	Similar to part-hierarchy (but in current Prometheus probably flat rather than hierarchical)	As above; all concept hierarchies function in the same manner.
Multiple method hierarchies	Not supported in Prometheus, but extending the decomposition model with a further dimension is possible.	As above; all concept hierarchies function in the same manner.
Multiple views	Based on a selection process defining a subset of matrix cells. May be federated.	Based on a concept tree, defining a subset of characters. May be federated.
Creating a new terminology	Well-defined part and property hierarchies need to be set up before starting data entry. For well understood groups this may be done centrally in funded projects; for small groups this may become a problem.	An “ad-hoc mode” is explicitly supported: characters may be introduced by defining a type and labels, without including them into any organizing concept hierarchies.
Adding new “characters”	Not required. Any matrix cell may be used for data storage. Federation only depends on the defining part, property, and property value hierarchies.	This is a separate process. Creating a character creates an ID used when storing or retrieving data and requires some metadata (type, a simple label). IDs must be managed globally, but when using GUIDs federations may independently add new characters.
Adding new concepts	In the defining part and property hierarchies this may require central management. Similar to characters, GUIDs might allow federations to add new parts and properties independently.	Uncritical, for coded data only local to a single hierarchy. However, concepts may be used for natural language markup as well, which creates a similar situation to Prometheus and is managed through GUIDs.
Revising semantics of concepts	Very problematic. In a federated situation any matrix cell may contain data. Changing the concept for a part would require identifying and reviewing all existing data (for all properties/methods) in all databases.	Uncritical, data do not depend on the concepts. The analog in SDD is the revision of a character, which would, however, involve only a single character at a time. On the other hand, in combination with the support of the ad-hoc definition of characters, poorly defined characters that have to be revised are much more likely.
Extracting ontological information	Presumably relatively easy since the model is based on explicit ontological concepts	Possible, but less reliable. SDD introduces some mechanisms that let authors express that a hierarchy may be read as ontological information. Full ontological information is expected to be associated with the concept definitions, rather than with concept usage in the concept trees.
Supporting relational characters (p. 122)	Special support exists for two-part, but not two-property relations. Properties depending on two parts must be defined asymmetrically.	Characters may be associated with multiple property- as well part-concepts. Relations are always symmetrical.

Both the “character decomposition” and the “character+concept hierarchy” models have advantages and disadvantages, some of which are compared in Table 38. A major disadvantage of the “character+concept hierarchy” model is the lack of methods to express more than a general association between concept and character. For example, a (constructed and hypothetical!) character: “ratio of profemur length to distance from front to middle leg insertion at the body” could be associated with:

<i>Part hierarchy:</i>	<i>Property hierarchy:</i>	<i>Method hierarchy:</i>
body, front legs,	distance between parts,	field methods,
middle legs,	length of part	hand lens,
profemur		stereo microscope

The complexity of this example cannot be adequately solved in decomposition models either (compare Table 38, last row), but for simpler examples with exactly one property and two parts the decomposition model may be less ambiguous.

In general, a number of reasons exist why the association between a character and parts (organs, morphological structures) may be ambiguous:

- If an object part is dominant, its properties tend to be viewed as properties of the entire organism. Examples: the color of fern leaves or the fruiting body of a fungus is generally considered the color of the entire fern or fungus, even though the fern roots or the fungal mycelium are usually differently colored.
- If the border between object parts is not easily perceived (though perhaps well-defined in theory). Example: the hypocotyl of a plant may be considered part of the “root” as well as part of “shoot”.
- If object parts are distinguished morphologically, but evolutionary forces generated convergent appearance. Examples: cladodes (or “cladophylls”, flattened stems resembling and functioning as a leaf) or rhizomes (stems resembling and functioning as roots). Even trained botanists, lacking the time to do adequate developmental or anatomical studies, may misinterpret such concepts. Authors of the data set may decide to include characters in a place where users expect them, rather than where they properly belong. Although this problem can be solved by distinguishing between part-of and kind-of relations, the data offer no validation that the content author understood or cared about these problems.
- If the compositional hierarchy allows multiple logical arrangements. Example: A “pedicel” (i. e., stalk bearing a single flower) may with equally good reasons be treated as “child of flower” or as “sibling of flower” and “child of inflorescence”.
- Finally, and highly relevant to the problem of identification, the relation of an observable character and a body structure may only be observable with great difficulty in the field. Parts of legs of small insects may be colorful and form a good field characteristic, but detecting which part of the leg exactly has which color may require a stereomicroscope. Similarly, in many sitting or swimming birds the exact relation between coloration and tail versus wing feathers hardly matters and is difficult to ascertain.

In principle these problems are all solvable, but in practice one data set may be optimized for phylogenetic analysis and another for routine identifications, leading to a certain ambiguity when attempting to retrieve ontological information from concept trees that are superimposed on a character list.

Whether the character decomposition is more or less amenable to handle these cases is not immediately clear. The simple model (Table 32 or Fig. 39) has similar problems (e. g., providing exact information about the part/property relations in the ratio example requires substantially more complex data structures in the character decomposition model as well, see p. 117).

Character-plus-concept hierarchy models often fulfill the same requirements as character decomposition models. As an example, the feature of Prometheus to directly score character states for an object part (compare p. 119) shall be discussed. In Prometheus, after selecting an object component, a state may be scored from the full list of all applicable states. Provided that the list of states is not overly long, this kind of scoring come close to natural language descriptions, where the character or property is often implied in the term (compare the example on p. 39). In Prometheus, the property is not stored, but implied through the relationship defined between states and categorical properties (“state groups”) in the terminology.

If a non-decomposition model is supported by a compositional concept hierarchy (e. g., as in SDD), the same user interface may be generated by browsing through the compositional concept hierarchy, and at a given node request a list of all applicable characters and states. It is now a question of the implementation, whether all characters applicable to this node and all subnodes, or only for the current node are displayed (the latter is possible desirable). Similarly, the implementation may display the categorical states organized by character, organized by another concept (e. g. property), or directly in a sorted list. Selecting a state from the sorted list works identical in the user interface to the Prometheus model. Only the data are stored by retrieving the char-

acter-variable associated with the state (implied information), and storing the state as character data for the current description.

An interesting aspect of multiple concept hierarchies is that combined hierarchies may be created algorithmically, thus often removing the need to also create “operational” hierarchies for practical purposes (see Figs. 41-42).

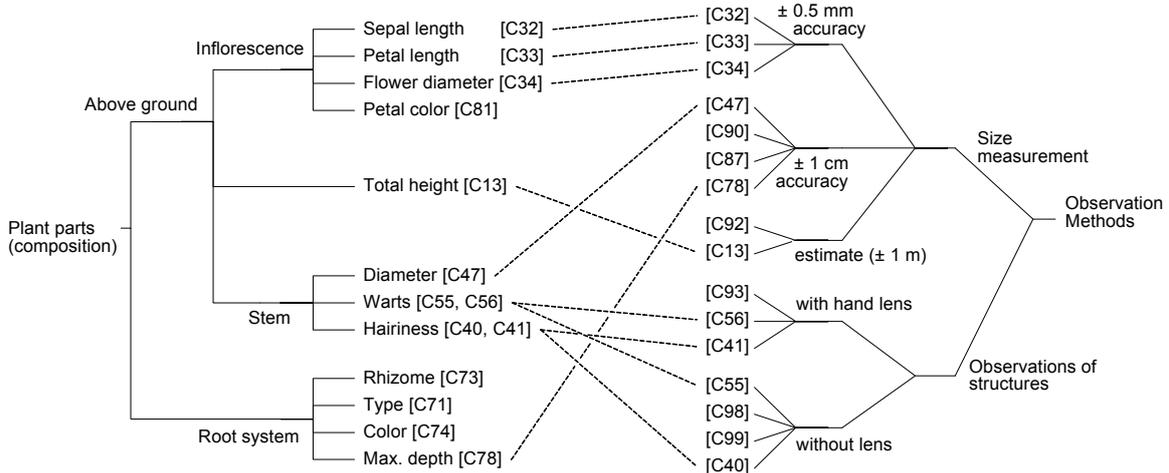


Figure 41. Two concept hierarchies (compositional and a methodological) associated with a flat character list (represented by character IDs in square brackets). Compare Fig. 42 for further information.

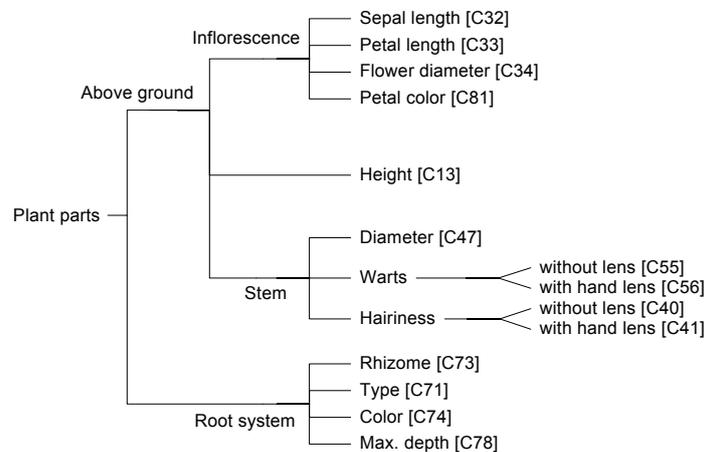


Figure 42. Multiple concept hierarchies (from Fig. 41) may be combined, providing additional information wherever the primary hierarchy contains multiple characters at a concept.

126. Concept hierarchies that are superimposed on a flat list of character may be a desirable alternative to strict character decomposition models.

127. The combination of concept hierarchies with a flat character list is desirable when the support of existing (“legacy”) data is a requirement. Concept hierarchies may be modeled as an optional part of the information model, whereas strict character decomposition models require decomposition information to be available to handle descriptive information. Concept hierarchies provide a large amount of the organizational and semantic advantages of character decomposition models without breaking compatibility with existing data.
128. Multiple concept hierarchies are desirable to express – in addition to object-part and property classification – also aspects of methodology, instrumentation, or simply arbitrary character subsets/filters.

4.12. Descriptive ontologies

The choice between “character decomposition” and “character plus concept hierarchy” discussed above is highly relevant to data exchange models like SDD. Although both models are equivalent with respect to many requirements, they introduce strongly divergent data representation models. This makes future migration of data difficult and applications may require a redesign rather than evolutionary development. A sound basis for the decision is therefore necessary (compare Table 38, p. 128). To a large extent, the answer depends on whether well-defined and stable concept hierarchies (semantic ontologies) for object parts (structures), properties, and methods or instrumentation, and development points, applicable to all organism groups from mammals to viruses, can be developed in the coming years. The following sections therefore discuss the problems the author sees in this respect. Many of the problems presented can be solved in principle, but often no immediate solution is known or has been tried. Work on the problems presented would be highly valuable. The goal of enlisting difficult cases without necessarily proposing solutions is to prepare a foundation for future decisions, helping information model designers in achieving a balance between abstraction (which at some level may hinder communication about a problem), complexity and functionality of a model being evaluated.

The concepts discussed here largely have analogs in explicit ontology languages such as OWL (McGuinness & van Harmelen 2004). However, because of the desire to continue to use UML to illustrate the examples, the following discussion will largely use the terms and diagrams used in software development. The UML diagrams are to be read as illustrations of a problem, not as proposals for a descriptive information model. Most diagrams are drawn under a naïve perspective which would result in a non-generalized software model applicable only to one taxonomic group (compare section “Level of abstraction of descriptive information models”, p. 42).

Object composition

Compositions (part-of relations) of concepts occur in modeling primarily if some object can be subdivided along the physical dimensions of space, mass, or time. The composition of a biological organism may contain parts like “head, body, legs”, but also “cell wall” may be composed of “proteins, chitin, β -1,3-1,6-glucan”. Space and mass-based abstract composition concepts are independent. In the body composition they fall together, in the chemical composition they are largely independent (the composition of a fluid is exclusively mass-based), and for a composition of distance segments the concept of mass is irrelevant. Space-based concepts include a notion of order and adjacency, mass-based only a summation of parts. Similarly, time-based compositions may or may not have an ordering and duration aspects. A process (e. g., an observation method in science) often includes a clear notion of order and duration of component processes (e. g., a fungus is cultivated for a given period, the cell wall destroyed, DNA extracted, DNA purified, PCR performed). However, the sequence of component processes may also be so variable and repetitive as to become irrelevant (e. g., “compare object with color chart”). Similar to spatial composi-

tions, temporal component processes may be nested (e. g., using a microscope while performing some operations).

Part-of relations inform that concepts belong together. A useful property of Part-of relations is that they are transitive (a part-of b , b part-of $c \rightarrow a$ part-of c). However, it is difficult to do reasoning about the properties of the objects having part-of relations. For compositions involving mass it is known that the total mass is the sum of the mass of the parts. For most properties, however, such deductions depend on secondary information. If a is part of b , neither $\text{color}(a) = \text{red} \rightarrow \text{color}(b) = \text{red}$, nor $\text{color}(b) = \text{red} \rightarrow \text{color}(a) = \text{red}$ is generally true. If the surface color is black for the insect body and red for the head, it is intuitive that the entire object is a mixture of red and black. However, if it is known that the fat body is yellow, the additional knowledge is necessary that this is an internal structure not visible in undissected insects.

The following sections will only discuss the physical arrangement and part-of structures. These are of primary importance in biological descriptions and are used as the basis for current character decomposition models (p. 116). However, observation or experimental methods also have a composition (part-of) structure.

Morphological object composition

Many objects are a composition of other objects. Fig. 43 shows an object that consists of one rounded square, two right-angled triangles, six circles, and one equilateral triangle. Aspects of object composition are:

- **Containment:** The entire object contains all objects listed. The rounded square further contains the two right-angled triangles.
- **Multiplicity:** The right-angled triangles occur twice, the circles six times.
- **Adjacency:** All objects are adjacent to the rounded square. The circles are adjacent to the right-angled triangles. No other objects are adjacent.
- **Ordered sequences:** The darker circles are between the lighter circles.
- **Relative position:** The circles are at opposite sides of the rounded square.

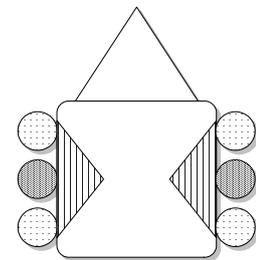


Figure 43. Abstract example of an object composed of other objects (components).

Fig. 44 gives an example for a morphological composition hierarchy for selected parts of the human body. Note that some composition details may have to be resolved by consensus, e. g., whether the foot starts below the ankle or whether the ankle is part of the foot; this is further discussed in section “Competing classifications of object parts” further down.

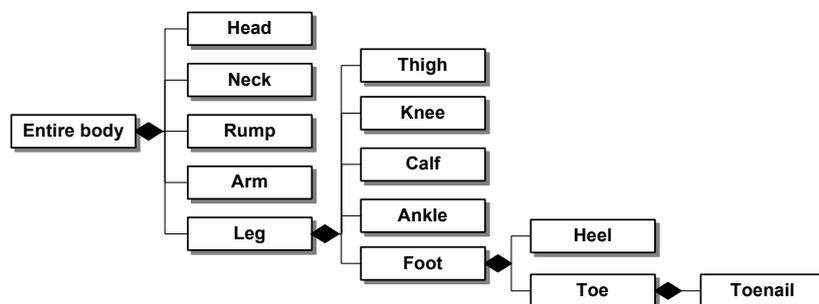


Figure 44. An excerpt from a morphological composition hierarchy of human body parts as a UML class diagram.

(Note: The terms “morphology” and “anatomy” are often used interchangeably. In this treatment, however, the term “morphology” refers to the outside organization or composition of an entire

object and the term “anatomy” to the inner organization that may require dissecting or disassembling the object.)

129. Morphological object composition includes aspects (multiplicity, adjacency, order) that are not immediately included in a part-of hierarchy. Support for these aspects is desirable.

Anatomical object composition

Morphological objects (composite or atomic) usually have a separate inner, anatomical composition. The outer and inner compositions often cannot be organized into a single hierarchy, i. e., the inner composition cannot be presented nested inside the atomic outer-composition objects. For the object from Fig. 43, a hypothetical “vessel” anatomy with a main vessel and branching side vessel is shown (Fig. 45). Similar situations occur for nerves or blood vessels in animals or the vascular system in plants. Fig. 46 gives another example for an anatomical composition hierarchy for selected parts of the human body.

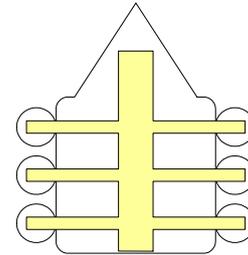


Figure 45. A “dissection” of the object from Fig. 43, showing an inner, anatomical composition hierarchy that is aligned with the outer composition hierarchy.

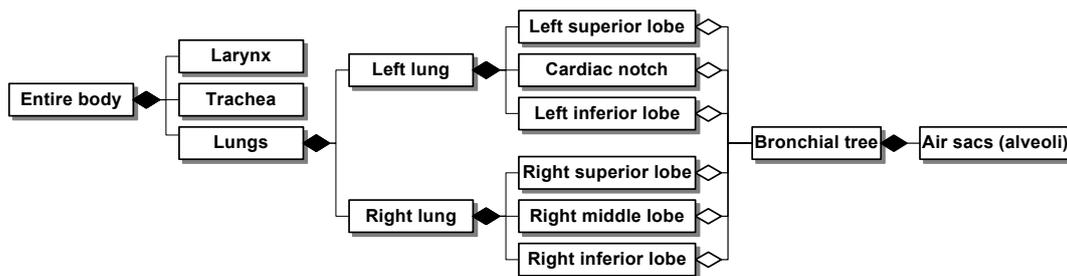


Figure 46. An excerpt from an anatomical composition hierarchy of humans showing a detailed hierarchy for lungs. Parts of the bronchial tree (and the attached alveoli) occur in all lobes of the lung (shown as shared aggregation, open diamond).

Morphological and anatomical compositions are separate hierarchies, but not independent. It may be desirable to be able to express which anatomical parts are contained inside a morphological object, even if an anatomical part stretches through multiple morphological parts. In the example of the human body (Figs. 44 and 46) the two hierarchies are so far related only by using “Entire body” as their base class, which does not allow one to determine which anatomical objects may be found in the neck. In Fig. 47, additional relations between morphological and anatomical objects are drawn in the form of dependency relationships. Both larynx and lungs may be associated uniquely with a morphological object, whereas the trachea (windpipe) runs through both the neck and the thorax. Relations like that between “Neck” and “Trachea” could perhaps be modeled by creating a new relationship type, a “partial-part-of” relation.

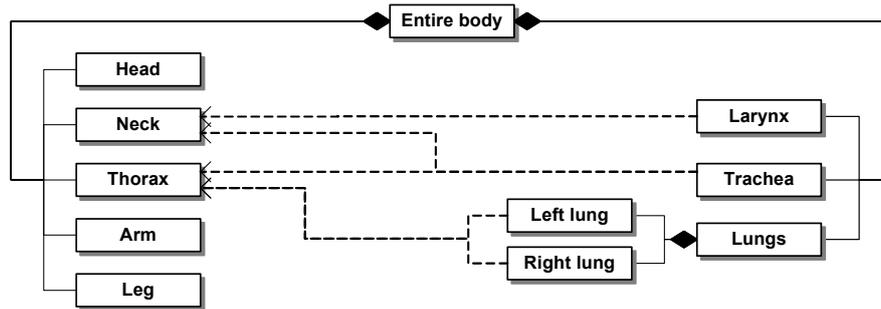


Figure 47. Interference between morphology and anatomy at the example of the human respiratory system shown as dependencies (dashed arrows).

130. Anatomical (inward) composition hierarchies are not necessarily nested inside a morphological (outward) composition hierarchy. They can therefore not be displayed in a single tree and support for multiple composition hierarchies is a requirement.
131. Mechanisms to express dependency relations between multiple composition hierarchies may be desirable.

Object decomposition

Decomposing objects, i. e., recognizing that an object is a composition and identifying the parts of this composition, is not always trivial. In many cases the decision whether to treat something as a separate object part (participating in an object composition), or whether to treat it as a property of the main object, depends on tradition rather than explicit rules. For example, most botanists will consider hairs on stem, leaves, etc. an attached structure (that would be modeled as a part in a composition). However, a dentate leaf margin will usually be considered a shape property rather than a larger number of separate teeth objects added to the margin (Fig. 48). Similarly, structures responsible for a color effect may either be considered a composition or implicit in a property (Fig. 49).

Pullan & al. (2005) analyze “striated area on petal apex: present”, concluding that it is a complex combination of “structure (petal apex), property (presence) and state information (striated)”. Other authors may express the same information as: “area on petal apex: striated”. Again, the fundamental problem is whether a part (which may then have a name) is recognized, or whether

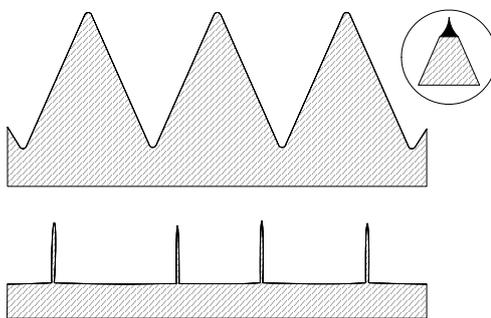


Figure 48. Teeth of a dentate margin (top) are usually not considered components (but implied in the property), whereas hairs on a margin or surface are often considered component objects. However, such teeth may have a substructure, e. g., a differently colored mucronate tip (circular insert).

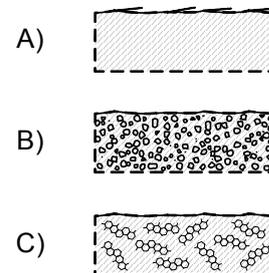


Figure 49. Object color may be due to surface structures creating physical colors effects (A), pigments (B), or soluble molecules (C). All may be viewed as a composition or not.

simply the anonymous region at the apex of the leaf is described through its properties. Rules how to systematically design a terminology are desirable for such cases.

Unfortunately, whereas in the previous example the second solution (using region instead of part) seems to be preferable, in many other cases no general “region” description is available as an alternative. In the case of complex butterfly or moth wing patterns it is often not possible to describe a component of the wing pattern without reference to its properties. The color of the third wavy band cannot be described as “structure X: wavy and red”.

Part of the rules governing object decomposition may be to consider something a part in a composition rather than a property if it has further object parts and properties. However this rule does not reflect the current customs in biological terminology. For example, teeth not only have properties that are directly shape-related (length, width, or angle), but may also have independent properties: different hairs may be found on the teeth, leaf veins may protrude at the tip, or the tip may be differently colored and mucronate (abruptly ending in a sharp point, insert in Fig. 48).

When designing a descriptive system that intends to make consistent use of object compositions, the fact that logical object decomposition rules and biological practice often disagree has to be taken into account. It is important to provide a mechanism to map property expressions (leaf hairy, margin dentate, fruiting body unilocular or multilocular) to object compositions (and their multiplicity) to allow comparisons and to provide readable and intuitive descriptions.

In some cases, independent properties may cause interactions that influence decisions about object decomposition. In Fig. 50 the left object is clearly recognized as a simple hexagonal shape. However, if the striping pattern has additional darker lines added to it at appropriate positions, partitioning the object into two triangular and one rectangular object is tempting. Such problems of human perception are often embedded in biological descriptive terminology. For example, it will be difficult to define an unambiguous decomposition of contours contained in butterfly wing patterns that does not depend on a small taxonomic group.

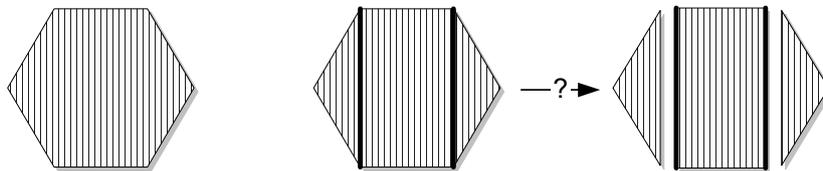


Figure 50. Ambiguous object decomposition.

Whereas in the previous example the preferred decision would probably be not to decompose into parts, in the next example decomposition is expected. Fig. 51 shows three “pseudocreations” with increasing reduction of head and neck shapes. Ultimately, the recognition of a head is based only on the presence of eyes. Pre-existing knowledge about possible head and neck reductions is essential to allow such an object composition. This knowledge is often missing when people try to identify organisms whose fundamental arrangement is new to them. When developing a key covering both insects (tagmatized into head and thorax) and spiders (single cephalothorax) the treatment of parts in a character decomposition model involving a mandatory part-information would be a serious problem. In insects the eyes are part of the head, in spiders they belong to a region of the cephalothorax.

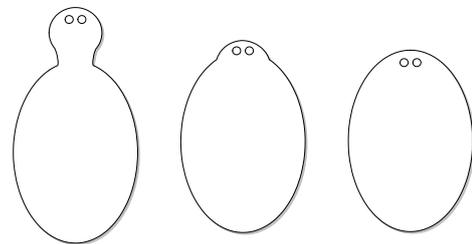


Figure 51. If head and neck shapes are increasingly reduced, the recognition of “head” and “body” is ultimately based on other features (and knowledge that a reduction exists).

132. Whether physical objects should be considered atomic or a composition depends on perspective and conventions, and may depend in complex ways on interaction with other compositions and properties. It is desirable to add mechanisms that help in communicating the perspective and conventions between designer and consumer or a descriptive terminology.
133. An object composition may often be considered a property of the parent object. The information system needs mechanisms to relate (or “map”) property and object composition expressions.

Competing classifications of object parts

In general, the decompositions (or “partitionings”) of biological organisms into parts are bound to a guiding principle considered optimal for a given purpose. The system may be based on morpho-anatomical aspects (as already discussed), but also on functional, biochemical, ontogenetic, life history stages (including multiple generations), phylogenetic, or similarity aspects. Different schools (and most notably terminology in different languages) frequently use different classifications.

Different decomposition systems may result in partially congruent concepts, i. e., a given part may occur in different systems. The names for such object parts under different classifications may or may not be identical. Where they are, they can no longer be used to infer the classification system. For example, a rhizome (an underground stem) may be classified as part of the root-system (functional concept) or stem (morpho-anatomical concept). While a petiole (leaf stalk) is commonly considered part of the leaf, the peduncle (flower stalk) is truly part of the stem system, but its length will commonly be noted together with other characteristics of the flower (or perhaps with the branching of the inflorescence, Table 39). Similarly, spines, thorns, and prickles have exact botanical definitions, but are in practice used interchangeably under a similarity or functional concept rather than a morpho-anatomically one. Note that in this case “spine or thorn” is a generalization (kind-of) concept, which generalizes multiple object components (part-of concepts; see also “Generalization of object parts (compositional concepts)”, p. 153).

Table 39. Examples of ambiguous or competing classifications of plant parts.

	True morpho-anatomical classification	Common alternative classification	Notes
Petiole (leaf stalk)	part of leaf kind of ?	part of leaf	
Peduncle (flower stalk)	part of stem kind of stem	part of flower	Other classification: “inflorescence” (combining flower and stem characters)
Spine (from entire leaf)	kind of leaf	“spine/thorn”	e. g., Cactaceae
Spine (from stipules)	part of leaf kind of stipules	“spine/thorn”	e. g., <i>Robinia</i> or other Fabaceae, many Euphorbiaceae
Spine (from petiole)	part of leaf kind of petiole	“spine/thorn”	e. g., in <i>Fouquieria splendens</i>
Thorns (from stem/shoot)	part of stem kind of stem	“spine/thorn”	in <i>Bougainvillea</i> the thorns are modified inflorescences!
Prickle derived from stem	part of stem kind of epidermis	“spine/thorn”	e. g., <i>Rosa</i> (rose, the “thorns” are prickles), <i>Smilax</i> (catbrier)

A classification may be useful in the majority of cases, but inappropriate in certain organisms. Fig. 52 presents examples where a common leaf classification is difficult to apply in certain taxonomic groups where leaf morphology varies continuously.

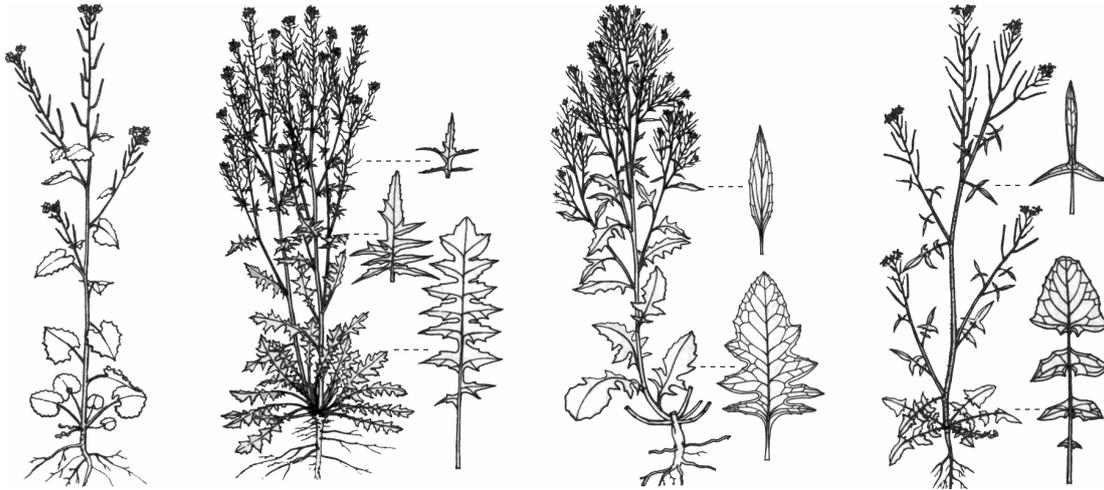


Figure 52. Examples of species with gradually changing leaf shapes, defying a strict classification into ground leaves/rosette, stem leaves, and inflorescence leaves (bracts). From left to right: *Alliaria petiolata*, *Sisymbrium austriacum*, *S. volgense*, and *S. orientale* (simplified after Rothmaler & al. 1985).

The questions whether something is considered a composition or a property (see previous section), which classification system to use, and whether a classification system for object parts is appropriate are a major source for instability in descriptive terminology. They generally cannot be resolved by logic, but require consensus and conventions. This consensus often depends on a given taxonomic group, but may also depend on a school of thought.

The major problem with this is that while the context can be described for descriptive terminology and descriptions using this, it is usually unavailable during identification (when the class name is not yet known). If identification depends on correct decomposition, or correct application of a composition hierarchy it may fail. Although this problem is in principle shared by both character matrix (p. 104) and character decomposition models (p. 116), the character decomposition models may suffer more if opinions about object decomposition evolve, and (at least current) decomposition models are unable to deal with multiple decomposition concepts.

134. Multiple morphological concepts and corresponding object composition hierarchies may exist. It is desirable to support alternative concepts of object parts and composition hierarchy.
135. The conventions whether something is considered a property or a composition, or which composition hierarchy should be preferred often depends on context, especially taxonomic scope.

Describing object-part composition

How can object or class-specific compositions be recorded as part of a description? How can descriptions of composition be compared with one another? The traditional solution is illustrated in Fig. 53. The terminology contains a list of potential part concepts and for each of these a “part-is-present/absent character” is defined. This solution is used both by character/character state models (DELTA, CBIT Lucid, SDD, etc.) and character decomposition models (Nemisys/Genisys, Prometheus description model).

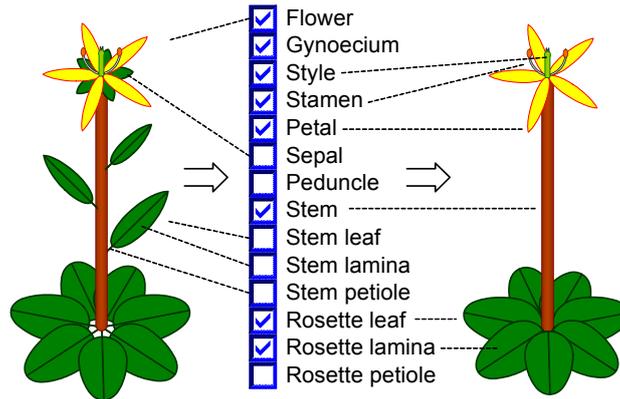


Figure 53. An abstract, generalized list of plant parts (left) is filtered (or mapped) through a list of present/absent characters to describe an actual plant species (right).

In DELTA or CBIT Lucid information about a composition hierarchy is available to humans (through prior knowledge, character labels or headings), but not to machine-reasoning. In SDD it is expressible through a machine-interpretable terminological concept hierarchy (p. 125), and in decomposition models (p. 116; Nemisys/Genisys, and Prometheus description model) it is an explicit part of the data storage model. In these models, the flat list of object parts from Fig. 53 is replaced by a general object composition hierarchy (Fig. 54, left side). By applying the information from “part-is-present/absent-characters” (provided these character are recognizable through metadata), a realized tree can be created (Fig. 54, right side). This process is perhaps more similar to the flat character model than may be expected; Pullan & al. (2005) describe it as “parts of the ontology that correspond to the form of the particular specimen/taxon being described are flagged as being present”. This may be due to the need to record not only presence, but also multiplicity (discussed in the next section, see especially p. 145).

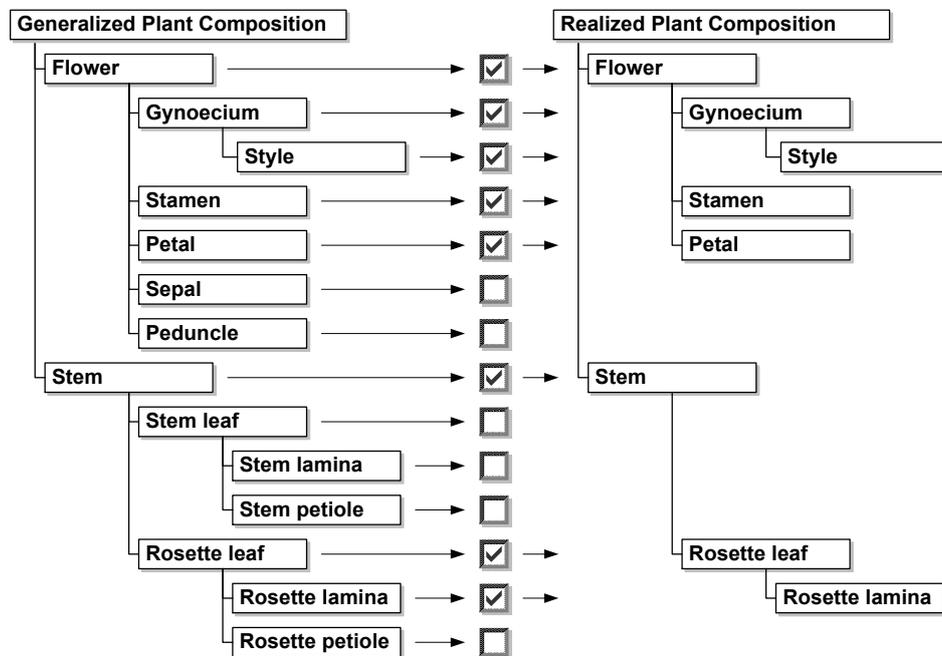


Figure 54. In description models providing an explicit compositional hierarchy (SDD, Nemisys/Genisys, Prometheus), hierarchy information may be inherited by the actual composition for a specific taxon (compare Fig. 53).

DELTA, CBIT Lucid, or SDD do not provide metadata whether a character is a “presence/absence character” that informs about realized composition. In contrast, in explicit character decomposition models the combination of a part with the “presence/absence” property may be used to deduce this (depending, however, on the “correct” use of the model – an iridescent layer may be present/absent, but not iridescence itself).

Rather than relying on implied deductions from a composition hierarchy, DELTA, the Nemisys/Genisys outline, and SDD provide character applicability definitions (compare “Character applicability rules”, p. 76), which often are due to presence or absence of object parts. The character dependency mechanism is able to infer from “petal is scored as present”, and “petal presence depends on flower present”, that “flower must be present”. Although parts of the desirable inference from composition knowledge are thus covered, it is not possible to deduce the composition with safety. Reasons for character dependency other than optional composition exist and only humans can make appropriate deductions.

A part-hierarchy like the one shown in Fig. 54 is intuitive and very useful for flowering plants. However, the more structurally diverse a taxonomic group is, the less intuitive will such a hierarchy be. Defining a generalized part-hierarchy for diverse taxonomic groups like Chlorobionta (ranging from microscopic green algae, over mosses and ferns to flowering plants) or Chordata (including primitive tunicates, cephalochordates, and vertebrates from fishes to mammals) is a challenging task. The results may be too abstract to be usable without additional annotations.

It seems a valid question whether the object composition hierarchy should be in the terminological domain at all, or perhaps rather in the description domain. The composition hierarchies (“part-of”) could be stored as a special form of descriptive data, similar to the absence/presence/multiplicity information. The relations shown in the right-hand side of Fig. 54 would then be part of descriptions, rather than inherited from the generalized composition hierarchy in the terminology domain, and may look like “petal is part of flower (5 times), stamen is part of flower (2 times)”. The information in such description-compositions could easily be compared, and, if appropriate generalization techniques are developed, the object composition expressed in descriptive data for a genus could simply be a generalization of the object composition of the species within the genus, the family composition a generalization of the generic compositions, etc. Some problems with this may be foreseen, however:

- Many relations would *a-priori* be known to be impossible (e. g., roots have neither leaves nor flowers). Burdening data entry with choosing the correct ones makes data entry slower and much more error-prone.
- Many relations are constant. The relation may only be missing, if a part is missing.
- It would be cumbersome to decide for every plant species whether the flowers are attached to the stem or to the roots.

Both problems may be alleviated using a combination of an inheritance mechanism working down the tree (i. e., composition information for lower taxa might be deduced from higher taxa, compare “Deductive inheritance”, p. 100) and carefully designed editing software, which arranges the information already in the hierarchy inferred from the higher taxa (which in turn may inherit from sister taxa), but enables methods to contradict this inherited information where it is not applicable in the special case. Another problem is that

- the concepts inherent in the definition of parts often already implicitly contain a hierarchy. In the example in Fig. 54, no choice for a different composition exists at all. The rosette (leaf) lamina must be part of the rosette leaf, the rosette leaf attached to the stem, etc. The actual part concepts as used in the figure are always a combination of a general part concept (leaf) with a position in the composition hierarchy.

An attempt to differentiate between a general part concept and its position in the composition context is shown in Table 40. It is clear, that much of the intuitiveness and clarity of terminology would be lost. For certain kinds of leaves (rosette, stem leaf, bracts) the solution may actually be preferable, avoiding definition problems with intermediate cases (compare Fig. 52 on p. 137).

However, distinguishing the various kinds of parts of a flower (gynoecium, anthers, petals, or sepals), all of which consist of one or several leaves (Fig. 55), by position is not only highly inconvenient, but in fact impossible if the flower is incomplete (sepals reduced rather than being converted to tepals, stamens or gynoecium missing in unisexual flowers). Tubular and ligular flowers may or may not be distinguished by position within the flower, they are morphological types and the distinction by place is only secondary.

When considering the practical use of descriptive data in identification tools, it seems dangerous to resort to definitions that require knowledge of the phylogenetic and ontogenetic origins of a part and which are difficult to verify on unknown objects. Most of the leaves in Table 40 and Fig. 55 would not normally be considered a leaf in the context of identification. It may be promising to add classifiers to the terminology of object parts marking all object parts which (1.) can be identified out of their composition context or which (2.) require a composition context, but this is always the same (as in stamen/carpel). For instance, petals can usually be recognized by their color, shape, and lack of sclerotization as well as by position. Current terminology probably still has to be changed (sepals and petals that are very similar but greenish-leaf-like are not normally considered tepals, petals are not considered tepals if sepals are known to be reduced, etc.), but this may be more acceptable than a more radical approach.

Table 40. Examples of related object parts distinguished by placement or otherwise.

"Normal" part name	"Generalized" part name	"Position"
1. Rosette leaf	Leaf	Rosette
2. Lower stem leaf	Leaf	Lower stem
3. Upper stem leaf	Leaf	Upper stem
4. Bract	Leaf	Inflorescence
5. Petal-like bract ¹	Leaf	Flower
6. Sepal	Leaf	Flower, outer ring
7. Petal	Leaf	Flower, 2nd ring
8. Tepal	Leaf	Flower, outer ring
9. Stamen (part of androecium)	Leaf → microsporophyll	Flower, outside Gynoecium
10. Carpel (part of gynoecium)	Leaf → macrosporophyll	Flower, innermost
11. Tubular flower (disc floret)	Flower	(undifferentiated)
12. Tubular flower (disc floret)	Flower	Central if ligular flowers present
13. Ligular flower (ray floret)	Flower	Marginal if tubular flowers present

¹ As in *Cornus florida*

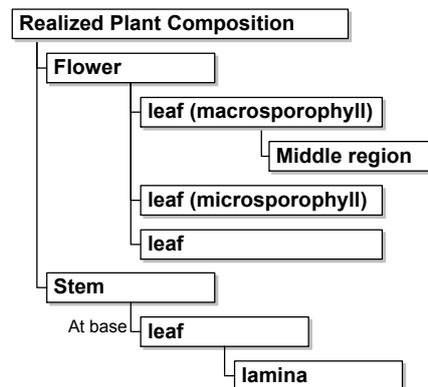


Figure 55. Hierarchy attempting to use only generalized terms.

The problems outlined are not intended as a proof that an alternative system of recording plants structure and composition cannot be found or would not be profitable. In contrast, it is probably highly desirable to study the problems involved in depth and try to develop practically

usable solutions. However, no existing model seems to have achieved to overcome the heuristic nature of part definition and recognition.

One path that may be worth attempting is to create a model where organism parts are handled similar to taxa, i. e., with descriptions that allow one to recognize them. Many part concepts are compositions in themselves and rely for their identification on recognition of the constituent components. The entire organism would simply be a special level, which normally would not allow further aggregation. In special cases (sessile colonial organisms like corals) even this would not be prohibited, i. e., the system could without modification go to a descriptive level above the individual organism.

Identification would then start to recognize, based on properties, some starting point for the part-identification. From there on, the system could support the recognition of further parts, if it has a concept of part composition and perhaps even knows about adjacent parts (the composition hierarchy itself informs only about sibling parts, compare “Adjacency”, p. 151). Identification may even start with paired information, such as: “I have a part – which has this color, shape, and size and occurs five times, connected to some other part – which has this color, shape, and size and occurs two times. What may this be?”. Such a combined part-identification might limit the possibilities substantially and may already allow the unambiguous identification of the parts, plus presentation of a list of species that fit the given descriptions for the identified parts.

- 136. The object-part-composition of individuals and classes may be expressed as part of the description (using characters or properties, depending on the description model).
- 137. The classical requirement is to record in descriptions whether a part is present. The hierarchical relations of the part composition are left to the terminology domain.
- 138. Whether a generalized object-part-composition hierarchy indeed belongs into the terminological domain or may be better placed in the description domain remains an open problem and needs further research.

Multiplicity of objects in compositions

Cardinality and multiplicity

Knowledge that an object has object parts (a flower has sepals, petals, anthers, and gynoecium) does not necessarily include information about the number of parts participating in a given composition. This may be an actual number in an instance (i. e., something has exactly n child objects) or it may be a constraint in a class definition (i. e., at least n and at most m instances of the child objects may be present, where $n \leq m$). In the first case it is always fixed (or unknown, see below), in the latter it may be fixed (e. g., “6 circles” in Fig. 43, p. 132), variable (e. g., “1-20 circles”), or unknown.

UML distinguishes between “**cardinality**, the number of elements in a set” and “**multiplicity**, the range of allowable cardinalities that a set may assume”. In the terms used so far, the cardinality refers to the number in an instance object, multiplicity to the potential range of numbers for a class of objects. It seems plausible to assume that these terms should in biology be used for object (individual, specimen, etc.) and class (population, species, genus, etc.) descriptions. As demonstrated in “Aggregation within individuals” (p. 93), however, aggregation (and therefore the concept of multiplicity) occurs in individuals as well. For example, a single specimen may have between 6 and 8 petals per flower. The term cardinality would only refer to the description of each individual flower. To avoid introducing differentiated terms in this discussion, the following text will always use the term *multiplicity* when dealing with descriptive knowledge that a composition exists with a fixed or variable number of parts.

Also note that questions of cardinality or multiplicity of classes are discussed with respect to the descriptive information to be expressed, not with respect to elements of the generalized descriptive information model such as DiversityDescriptions or SDD, expressing this information.

One may assume that composition and multiplicity are questions both of actual descriptions and descriptive terminology. However, similar to the discussion about object composition itself (see “Describing object-part composition”, p. 137), biology is so variable that it is rarely useful to express such knowledge as part of the terminology. In contrast, it is highly informative to express that all insects (except for a few highly reduced members) have six legs. This, however, is simply a class description. Rather than obtaining constraints or plausibility checks from definitions in descriptive terminology, editing applications may desire to check newly entered data against available descriptions of higher classes and warn if discrepancies are found.

139. In addition to object composition, the multiplicity of a composition must be supported in the information model.

140. It is not required to support composition and multiplicity information as part of the definitions of object parts in terminology.

Expressing multiplicity through instance cardinality

One may be tempted to express information about object composition and multiplicity 1:1 in an object-oriented programming (OOP) model, where each physical object and object part is represented by an instance of a programming object, and where multiplicity is represented by the number of instance objects in a collection.

Firstly, with such an attempt it would be difficult to represent classes where the number of parts in an object composition is expected to be variable. Object-oriented design is based on the (correct) assumption that a single object composition will have a defined state (i. e. a cardinality) and a corresponding exact number of parts at any given point in time. It can correctly model changes over time (e. g., a caterpillar becoming a butterfly imago and the number of wings changing from 0 to 4).

However, in descriptions commonly variation of individuals is summarized, both over time and between individuals belonging to the same class. Such class descriptions are not limited to higher taxa: genetic polymorphisms exist even at the lowest population level and environmental differences create different phenotypes for the same genotype.

Whereas in object-oriented programming, composition mechanisms (arrays, collections, lists, etc.) may be static or dynamic in class objects, the cardinality is always fixed at a given moment in time in object instances. That is, a “count()” function will always return a defined count of the members of the composition, not a range that might be interpreted as the range of variability.

One possible solution is to represent the variability through a set of instance objects derived from a common class. To express a range of “1-6”, six different object instances with 1, 2, 3, 4, 5, and 6 parts would have to be created to represent each possible object within the range. Unfortunately, all permutations would have to be created if a composite object contains several ranges. The number of objects to be instantiated quickly becomes impractically large (three ranges “a = 1-6”, “b = 11-20”, “c = “500-1000” would require $6 \times 10 \times 500 = 30\,000$ parent objects with 512 to 1026 child objects, i. e., a total of 1 223 096 250 parent and child objects!).

On the other hand, using the composition mechanisms provided by OOP languages does have tempting advantages. In insects with six legs creating the legs as instances in memory is unproblematic in terms of computing efficiency and has the advantage that it appropriately informs where properties are different in the legs (Fig. 56).

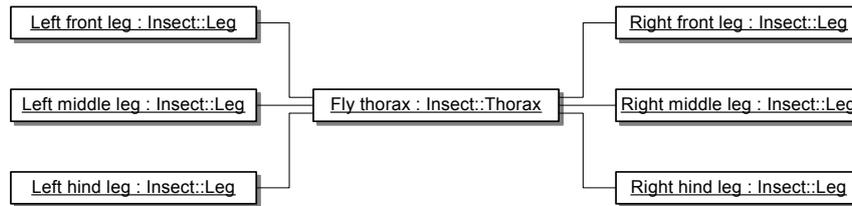


Figure 56. Six instances of the leg class are associated with a thorax instance of a particular insect. The UML object diagram corresponds to the class diagram shown in Fig. 57 where the multiplicity of the thorax-body composition is exactly ‘6’.

141. Object-oriented practices to represent multiplicity/cardinality in a composition cannot easily represent variability of cardinality in a description (e. g., “3-7 leaflets per leaf”).
142. Representing variability of multiplicity/cardinality in a composition through a collection of instances may require a huge number of instances, making this probably impractical.

Categorical multiplicity

In many object compositions, the number of parts may be so high that in practice they will not be counted and instead expressed as “much” or “many”. Examples from biology are the number of hairs on an animal or a plant or the number of cells in an organ. Therefore, only the lower range of multiplicities is fixed (to ≥ 0 , i. e., negative multiplicities are not possible), whereas the upper side is open (UML “0..*”). Again, although the multiplicities in a class definition support this in object-oriented languages, it is not directly expressible in object instances, requiring the use of special methods when attempting to express object descriptions through instances.

Furthermore, the data recording method may include a rule that above a certain number counting is considered impractical or inefficient. Instead of simply expressing this as “much”/“many”, estimates may be made into which of multiple categories the count is most likely to fall. For example, multiplicity may be recorded as an ordinal categorical expression of amount (e. g., “few”/“some”/“many”). Data for a given object (e. g., stamens in a flower) may be mixed, descriptions of objects or classes with a low count expressed quantitatively, descriptions with high counts through one or several categories.

In biology, the accepted recording method and the point where categories are used rather than exact counts may depend on the taxon group as well as on the specific character. For example, in most genera of flies the number of hairs on the forehead may be recorded only in broad categories, whereas in some genera the difference between 20 and 22 hairs may be diagnostically important.

The case of multiple categorical ranges presents not only difficulties when expressing counts of parts through instance composition, but also when expressing them through numerical data types. Whereas a single “positive infinity/overflow” value may be part of the definition of such a data type, multiple categories require more complex solutions.

In Fig. 57 an attempt is shown to mix composition multiplicity with a categorical expression, deriving a new class “HairUncountable” with an additional property “CategoricallyCounted” from the general class “Hair”. Instances of either class may be used to describe the presence of hair objects attached to the various parts of the insect body. However, the example shows, that this solution introduces a new problem: Now multiplicity is significant between some instances (e. g., “Forehead” and “Hair”), but insignificant in others (e. g., “Forehead” and “HairUncountable”, the information being here contained in the attribute “Categorically counted”). Doubtlessly, a reasoning algorithm may be developed that correctly interprets these situations, but the complexity involved gives rise to doubt whether this is the best solution.

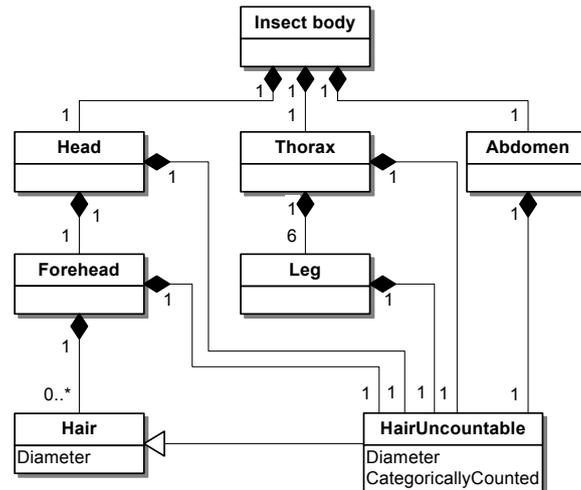


Figure 57. UML static class diagram with an attempt to model an insect as a simple composition. All parts may have an uncounted quantity of hair (expressed in the categories “none”, “few”, “many”). Furthermore, the quantity of hairs on the forehead may be expressed either categorical or may be counted, perhaps in taxa where this is diagnostically significant.

Further differences may be seen when additional properties of these classes are considered. In Fig. 57, the class *HairUncountable* inherits the attribute “diameter” from *Hair*. However, in “*Hair*” the “diameter” values directly represent the entire hair population (i. e., the diameter of each observed hair is recorded separately), whereas in the case of “*HairUncountable*” diameter can represent an aggregated statistical measure (e. g., average, compare “Standard aggregation methods”, p. 85) of the sample of “many” hairs. To obtain this aggregated measure, a sample of hairs for “*HairUncountable*” has implicitly been measured, which may be recorded in a separate collection of sample *Hair* objects to *HairUncountable*. These “sample objects” would be identical with the “*Hair*” class, except that the multiplicity of the relation indicates sample size instead of quantity in a composition relation (Fig. 58). Again, the different semantics of the multiplicity of an object relation cannot be expressed in a general object-oriented modeling mechanism and would have to be documented and implemented separately.

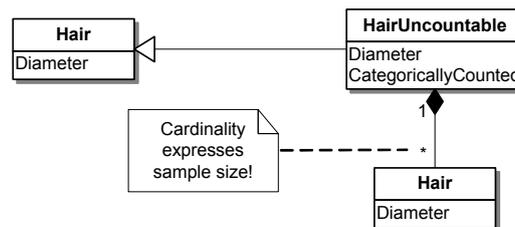


Figure 58. UML static class diagram extending the model from Fig. 57 to include sample objects for measuring diameter of hairs in the case that the quantity is measured categorically.

143. In addition to quantitative expression of multiplicity in object compositions, also categorical expressions such as “many” or “ ≥ 20 ” must be supported.
144. This issue is not a question of class versus object descriptions; the need for categorical multiplicity ranges arises even in individual objects where the composition is in principle countable.
145. Often more than one category is used (e. g., “few”/“some”/“many”).
146. Object composition multiplicity may be a mix of quantitative (1, 2, 3, ...) and categorical expressions.

147. Whereas a single category “beyond countability” may relatively easily be supported by quantitative data types, multiple categories with more or less well-defined ranges require more complex data structures.
148. Expressing multiplicity of object composition through a mixture of instance composition and values of categorical properties requires complex reasoning algorithms, interpreting values of instance properties as well as the instance multiplicity itself differently, depending on whether categorical multiplicity properties are present or not.

Uncertain or unknown multiplicity

Multiplicity ranges for compositions may either express knowledge of variability or lack of knowledge. The exact number of parts in a composition may be insufficiently known due to observation problems or poor recording of data. This may have been recorded without further qualification as a range (e. g., “10-20”), or the uncertainty may have been made explicit using modifier terms (e. g., “probably 6, perhaps 7”).

Uncertain multiplicity is not in principle different from uncertain attribute values (compare “Certainty modifiers”, p. 207). However, although normally presence is implicit in multiplicity (compare critique in “Basic property types”, p. 62), uncertainty of presence and multiplicity may need a separate mechanism. Presence may be certainly known, but multiplicity may be uncertain. Uncertainty of presence (e. g., “probably present”) can also be interpreted as a range of “0 to 1”. However, the statement “probably 1..20” does not tell whether 20 is uncertain and presence is certain or whether this may also include “0..20”. This may be expressible if upper and lower limits of the range can be qualified separately with certainty modifiers. The statement “certainly 1 to probably 20” would clarify this. However, the expression of “probably 6” is difficult to express in this way.

Related to “uncertainty” is the issue of coding status values, which allow one to express that an attribute has not yet been coded, that it is irrelevant to code it, etc. (see section “Coding status”, p. 74). Similar to the mechanism of certainty modifiers, it would be problematic to introduce separate mechanisms for properties and multiplicity.

149. Methods to qualify multiplicity in object compositions as being uncertain or express that multiplicity is unknown are required.

Describing object multiplicity

The problems with using OOP composition mechanisms to express the number of parts in biological object compositions strongly suggest that the number of parts should be expressed in a class attribute instead. Such a solution (Fig. 59) is analogous to the one already presented for object composition hierarchies (Fig. 53, p. 138). Fig. 59 illustrates that the definition of multiplicity is usually relative to the parent object (1-3 leaves per stem, 1 lamina and petiole per leaf). This reference to a parent is not always unambiguously clear in biological terminology (e. g., are “bristles above eyes” in a fly to be counted per head or per eye?) must be explicitly defined.

In principle, expressing object multiplicity this way implies presence/absence and removes the need for a separate filter working on structural composition (compare “Describing object-part composition”, p. 137). One minor problem with this in current DELTA-like information models is that character applicability rules (compare p. 76) typically do not support quantitative characters as controlling characters, and thus expressing dependency requires secondary categorical characters. This may be a primary reason why current descriptive terminology often defines has separate presence/absence and multiplicity characters.

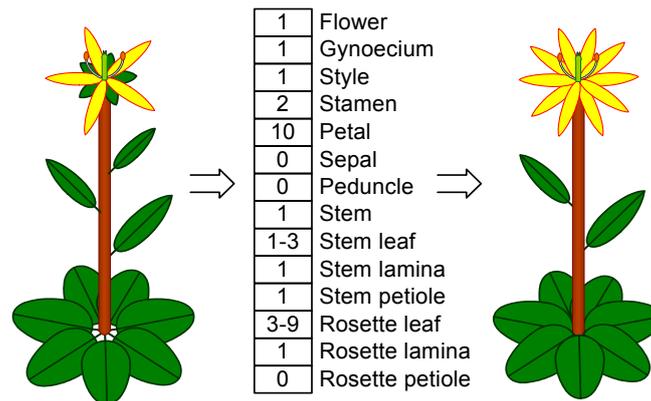


Figure 59. Multiplicity may be expressed through quantitative properties (extending the presence/absence filter shown in Fig. 53, p. 138). Some properties could remain Boolean (Gynoecium, lamina, petioles) because the compositional ontology may restrict their allowable values to 0 and 1 for all taxa.

Fig. 60 shows a conceptual OOP model to express multiplicity (ignoring the issue of multiplicity ranges) through a separate “Quantity” attribute on each class. This approach ignores the multiplicity of the class composition relations itself (compare Fig. 57). One advantage of using a class attribute is that this may be left undefined and that mechanisms to express coding status (p. 74) or certainty (p. 207) may be applied to it just like to other attributes.

In certain cases it may be desirable to support a mix of the attribute and the relation cardinality solutions (Figs. 60 and 57, respectively). For example, if multiple legs of an insect have different properties, multiple leg instances may be appropriate. The total multiplicity of parts could be calculated by summing the quantity attribute of all objects of the same class. Special methods would have to be defined if some leg objects have coding status unknown for the “quantity” attribute or uncertainty is indicated in a subset of objects. The “quantity” attribute would require a metadata item so that it can be recognized as an attribute associated with a specific composition relation. One way to do this would be to put the attribute on the relation itself, i. e., using association classes for composition relations.

An interesting aspect of such a model is that the aggregation operation may also be meaningful along a generalization axis. Example: fore-wing and hind-wing are both a kind of insect wing. If the fore-wing has 3 wing spots and quantity 2, the hind-wing 5 wing spots and quantity 2, the total number of wing spots on the fore-wings is 6, and 16 spots occur on all wings of the insect (using the generalization).

To extend this model to also cover categorical and range multiplicities, additional attributes are needed. In fact, quantity may have to be a collection (or set) of quantity ranges to express variability such as expressed in the categories: {“none”, “1”, “3”, “5”, “7 to 10”, “11 to 20”, “more than 20”} or through range predicates: $\{1 < x < 3, 5 < x < 10, 11 < x < 100, 9 < x < 12\}$.

Note that this model has not yet been further pursued in any information model discussed in this thesis. Currently presence or quantity information in SDD is expressed as characters that are undistinguishable from other characters. However, an extension to make software aware of special properties may be added in a

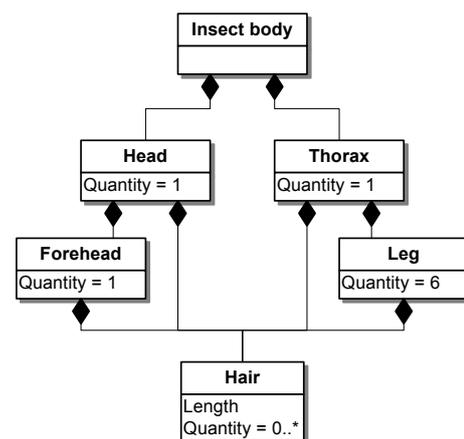


Figure 60. UML static class diagram similar to Fig. 57, but expressing multiplicity always through a “Quantity” attribute. Each quantity attribute may be further annotated using certainty modifiers.

future release. Many questions are open. For example, the handling of repeated objects where descriptions refer to different regions of an object, e. g., instances describing the base as hairy, the center as glabrous, the tip as hairy again. Depending on the model this may or may not be three instances of an object, but the total quantity may be one.

150. Multiplicity in object compositions may be expressed in attributes of child objects. These have special semantics and metadata to recognize them are desirable.
151. A combination of multiple child objects (if child objects differ) and multiplicity attributes may be desirable.

Spatial arrangement of objects in compositions

Absolute object orientation

Entire objects are usually viewed under an agreed orientation convention (front/back, top/bottom) that distinguishes between otherwise identical states. In Fig. 61 the rectangular shapes are oriented by the presence of additional features that define a front and a back. The recognition of absolute object orientation is often essential to successfully name the parts of composite objects.

In biology, absolute object orientation is achieved by convention and taught in introductory biology courses for organisms where it is not intuitive. The most important clues are geotropic orientation, the position of eyes and mouth-like organs, and levels of symmetry (see further below).

The terms top/bottom usually refer exclusively to geotropic orientation and are problematic if organisms with different geotropic orientation are compared (e. g., humans and dogs). Some organisms may have no clear geotropic orientation, and the orientation may be inferred from other members of their taxonomic group (e. g., soil insects, snakes living in shrubs and trees). In technical language, the preferred terms are therefore anterior/posterior = front/rear and (in most animals) dorsal/ventral = back-side/belly side. Since in animals anterior/posterior is usually defined by the position of mouth and anus, oral/aboral may be used instead of anterior/posterior. Right and left is the result of recognizing two planes of absolute orientation (anterior/posterior, dorsal/ventral).

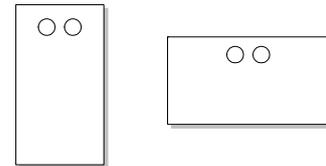


Figure 61. Objects obtain an absolute orientation (e. g., top/bottom) through convention.

Note that some commonly used terms referring to regions of an object are independent of either absolute orientation or even composition. Examples are “surface”, “edge”, “inside”, or “outside”, each of which may refer to a single, not oriented geometric object.

The identification of organisms from unusually organized taxonomic groups (e. g., horseshoe worms, Phoronida, or highly reduced plants like *Lemna*, *Tillandsia usneoides*, etc.) may fail because the overall orientation rules fail and specific acquaintance with the organization of these taxa is necessary. Since the fundamental orientation is recognized in the majority of organisms, it seems not necessary to devise a completely new identification and description terminology that is fundamentally independent of absolute object orientation. Instead, the possibility that no orientation is achieved during identification may be captured through special questions in keys. These could lead to a set of richly illustrated “fall-back” keys that deal specifically with problematic organisms.

152. Concepts to fix the absolute orientation of physical objects in space are an important means to facilitate object recognition.

Relative object orientation in compositions

Two objects acquire a relative orientation if they are placed adjacent to each other in a composition. In Fig. 62 the two composite objects on top are identical, whereas the addition of squares makes them distinct. Relative to the square, the circles are placed near or distant from the rectangle. In biology, the following terms often indicate relation orientation:

- **Close to attachment:** proximal, basal, at the base, origin, or center.
- **Distant from attachment:** distal, apex, apical, at the tip, or margin.

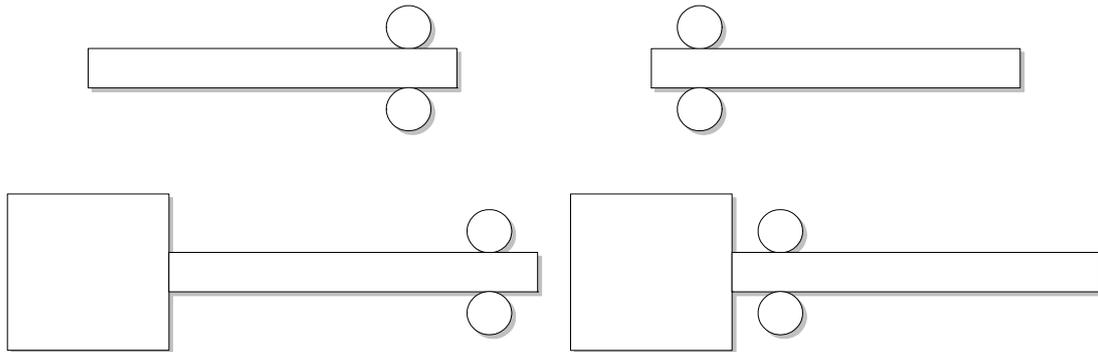


Figure 62. Parts of composite objects may have relative orientation. The two objects on top are identical, whereas the addition of a square adds orientation to the rectangle, differentiating between a distal (left object) and proximal (right object) placement of the circles.

Both absolute and relative orientation may result in problems in state typologies. In the case of oriented shapes, it must be decided whether the shapes distinguished by orientation should be given different names, or whether the shape and its orientation (or the point of attachment) is captured in a separate character. In biology, many pairs of shape terms (ovate/obovate, pyriform/obpyriform, claviform/obclaviform...) are distinguished only by 180° inverted orientation (Fig. 63).

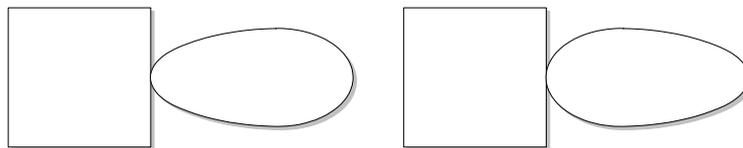


Figure 63. Orientation of shapes may result in different shape categories (e. g., ovate/obovate).

153. In object compositions the relative orientation of physical objects is an important concept to facilitate object recognition and should be supported in the information model.

Symmetry

Many objects have symmetries that add an additional level of orientation (Fig. 64). Most animals have a bilateral symmetry, for example, insects have pairs of eyes, antennae, wings, front, middle and rear legs. Other forms of symmetry found in organisms are radial (e. g., echinoderms), biradial (e. g., comb jellies), and spherical (e. g., single-celled algae, Radiolaria, or Heliozoa) symmetry. The concept of symmetry may be applied to the entire organism or to parts, e. g., to flowers of plants (where radial and bilateral symmetry are called “actinomorphic” and “zygomorphic” to confuse the zoologists...). Symmetry does not define a “right” and “left” side; these are part of absolute orientation (p. 147, above).

The geometric object displayed in Fig. 64 was described above as a composition with six circles attached to the rounded square. Recognizing symmetry, it may also be described as a composition with two symmetrical sides, with three circles attached to each side. Furthermore, since the circles have paired properties (shading), the object description may stress this and decompose the object into three pairs of circles, each pair with unique properties, and each pair split into two sides.

Note that the line of symmetry cuts right through the main rounded square and equal-sided triangle objects. Symmetry adds orientation, but is not well represented by an object composition hierarchy. It adds to the problems of the decomposition of objects into parts, since the side of symmetry (and of other forms of orientation) may be considered to be parts in the composition. However, whereas the symmetric equality of the patterns on the outer circles is informative, the equality of the pattern of the triangle on top is not very informative, because the symmetry cuts across a single object.

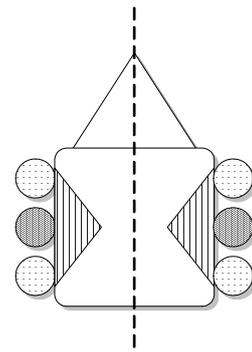


Figure 64. The geometric composite object from Fig. 43 has a bilateral symmetry.

Because of phenological and ontological variation, most biological organisms are not truly symmetric in a mathematical sense. Pictures of human faces where the right or left halves are mirrored often look like different persons. Biology therefore uses an abstract concept of “approximate symmetry” (Fig. 65, left). Singular deviations from symmetry may be recorded (the heart of humans being on the left side, the right pincer of lobsters being much larger) but do not destroy the recognition of a fundamentally symmetric organization (Fig. 65, right).

Biological terms usually applied to orientation involving symmetry are *lateral* for bilateral (or “zygomorphic”) symmetry, *central/peripheral* for radial or spherical symmetry. In botany the orientation terms *adaxial/abaxial* are often used relative to *axial* symmetry.

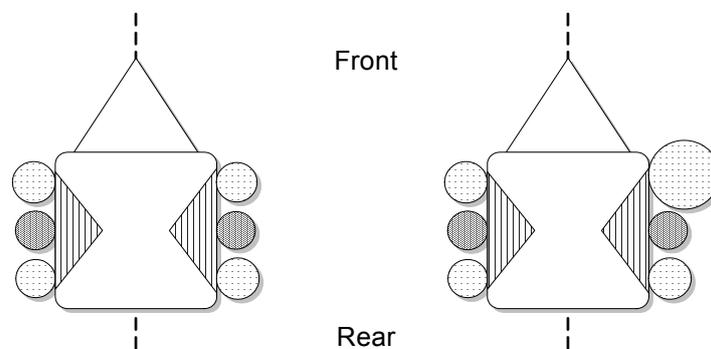


Figure 65. Minor variations (e. g. in the size of circles and lateral triangles, left object) may break “strictly” or “mathematically” defined symmetry. However, an abstract concept of “approximate symmetry” remains. Furthermore, a singular exception from symmetry (large circle in right object) still allows the recognition of a line of symmetry, although the whole would no longer be called “symmetrical”.

154. In object compositions, symmetry is an important concept to facilitate object recognition and should be supported in the information model.

Spatial gradients

A spatial gradient may be described as an oriented function on a surface or within a volume. The simplest case is a one-dimensional linear function oriented in some way on the object (absolute or relative). More complex cases involve two- or three-dimensional functions (e. g., chloroplast density in algae). However, although butterfly wings or spotted animal furs could be described by such functions, such “patterns” (compare “Pattern versus composition”, p. 165) would not normally be considered a gradient. Most likely, a form of oriented variation on an organism is considered a gradient only if the function is continuous and has no inner extremes.

The kind of change described by the gradient function may be the value of a property of the main object (e. g., lightness of color, increasingly dense pattern along an object, Fig. 66), it may be the cardinality of a child object in a composition (e. g., density of hairs along the stem of a plant), or it may be a property of such child objects (e. g., size of hairs on a stem; compare Fig. 67). The presence of a gradient itself can be expressed through a Boolean presence/absence property. However, where child objects in a composition are affected, special methods must be devised when using object-oriented composition models. Again, where machine-reasoning is intended, specialized reasoners will be required.

In the practice of biological descriptions gradients are not perceived as a major problem. Gradients are rarely considered diagnostic and are often documented in free-form text annotations or modifiers applied to the fundamental property (e. g., “hairiness: strongly hairy” with free-form text note “(stronger towards the base)”). A case where the available data recording in biology actually allows recording a gradient through its gradient function is not known to the author.

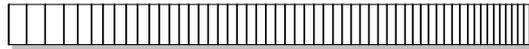


Figure 66. Density gradient (not considered object composition here).

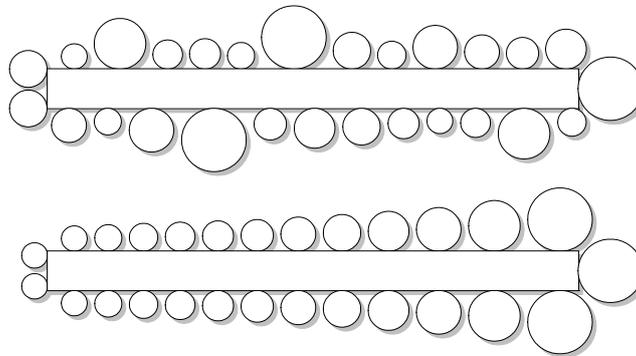


Figure 67. Rectangular parent object with multiple attached child objects (circles) of variable size. In the upper example circles of various sizes are randomly distributed, and descriptive statistics are well suited to describe the extremes, mean, and variability. In the lower example, the circle size varies on a gradient along the parent object. Standard descriptive statistical measures describe the size variation in a gradient inadequately.

155. Spatial gradients may interact with object composition, depending on properties or multiplicity of child objects (parts of main object).
156. Spatial gradients usually interact with absolute or relative object orientation; the data or terminology model must allow for this.

Adjacency

Humans often interpret object compositions as also expressing aspects of order and adjacency. Whereas the adjacency of parent and child objects in a structural composition may indeed be inferred, the adjacency of child objects among each other is less clear. Compositions like “plant = root + vegetative stem with leaves + inflorescences (= stem and leaves)”, or “insect thorax = thorax without legs + front legs + middle legs + rear legs” will intuitively be interpreted as expressing adjacency and order. However, if the pronotum (a part on top of the thorax, adjacent to the head) is added to the insect example, the assumption of order and adjacency breaks down. Adjacency is never expressed, but implied based on external knowledge about plants and insects.

Some information about relative location of objects can be expressed if a collection of parts is explicitly defined as a sequence (i. e. an ordered list, where the order is considered semantic and cannot be changed at convenience). This concept is supported in OOP languages or RDF, but not in standard UML or W3C xml-schema. If insects have three pairs of legs, the first pair is adjacent to the second, and the second to the third. However, sequences often are insufficient to express which object borders on another object. In the geometric example from Fig. 43, the triangles in the rounded square are adjacent to the circles, but not to the triangle on top. This is not evident from any hierarchical object decomposition or from a sequence. Another example from insects is: Antennae and eyes are parts of the insect head, but they may be adjacent or not (and this may be diagnostic).

A separate mechanism is required to express adjacency of objects. In UML this may be modeled by introducing a new stereotype “adjacent” for binary class relationships (Fig. 68, left). It may further be desirable to also record the orientation in which two parts are adjacent, causing the introduction of an association class (Fig. 68, center) to record attributes on the association (e. g., relative orientation: “basal”, “distal”, or “proximal”). This model supports the standard mathematical property expected for adjacency, i. e., it is reflexive ($A \text{ adj } B \rightarrow B \text{ adj } A$), but not necessarily transitive ($A \text{ adj } B \text{ and } B \text{ adj } C \rightarrow A \text{ adj } C$ or not $A \text{ adj } C$).

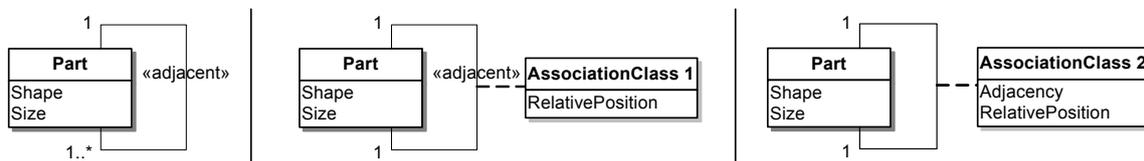


Figure 68. Three UML class diagrams attempting to model object adjacency. The left side shows a simple binary association introducing an “adjacent” stereotype (which is not available in standard UML profiles); the center adds an association class to record attributes of associations like “basal”, “distal”, or “proximal”; the right side records degree of adjacency in an additional attribute of the association class.

Modeling adjacency through an association implicitly assumes that two objects are either adjacent or not. In reality, however, a continuum between fully adjacent, slightly apart, neighboring, and completely apart exists. It will usually be impractical to measure exact distances of “adjacency”, but a categorical property with more than two values of adjacency may be more practical than a Boolean decision (Fig. 68, right).

Adjacency may be defined on classes with a multiplicity relation > 1 (for example, front, middle, and rear legs in Fig. 69), but often can be fully expressed only if the classes are specialized so that only relations with a multiplicity of 1 can be found (e. g., wing spots on an insect wing). A consequence of this is that adjacency is not a very practical mechanism to fully replace ordered sequences. Expressing any form of adjacency through adjacency relationships or association classes would require for the up to 200 equal legs of a millipede to create “leg 1”, “leg 2”, ... “leg 200” classes. It may make more sense to allow for a separate adjacency inference mechanism for ordered sequences.

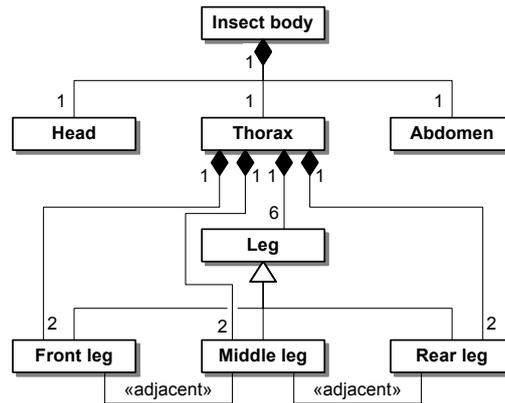


Figure 69. UML class diagram of the insect example introducing a tentative association of “adjacent” stereotype (not available in standard UML profiles). Legs can be included in compositions alternatively in their general (“Leg”) or derived (“Front Leg”, etc.) form.

157. Adjacency of object parts in a composition is an important concept that is desirable to be supported by the information model.

Location

In general, the location of parts in relation to the entire object or to each other is primarily expressed in a system based on absolute and relative orientation, symmetry, and the definition of adjacency or ordered sequences (see Table 41 for biological examples).

Table 41. Examples of location statements taken from biology.

Localization method	Example
Absolute orientation	top-most branches of tree
Relative orientation	root base (= adjacent to stem)
Relative orientation	center/margin of fungal culture
Relative orientation	wing tips
Relative + absolute orientation	upper surface at leaf tip
Relative + relative orientation	adaxial surface at leaf tip
Symmetry	lateral stripes
Absolute orientation	right pincer of lobster
Adjacency + absolute orientation	black patch above the eyes
Ordered sequence + absolute orientation	first 10 legs of a centipede

(The terms upper/lower and adaxial/abaxial surface are not true synonyms: rarely the adaxial may be the lower (geotropic) surface.)

Beyond this, location may be more precisely defined by relative (e. g., “at $\frac{2}{3}$ of length of parent object”), absolute (e. g., “1 cm from the margin”) distances, or angular (e. g., “at right angle”) measurements.

158. The concept of “object location” is a synthesis of absolute and relative orientation, symmetry, adjacency, and sequences.

Generalization of object parts (compositional concepts)

Composition versus generalization

In parallel to the object compositions discussed so far, a generalization hierarchy exists for the concepts referring to the physical parts of objects. Whereas composition hierarchies are expressed through “has-a/is-part-of” (or “contains/contained”) relationships, generalization hierarchies define “is-kind-of” or “is-type-of” relationships. In information modeling, this is generally called a classification or generalization/specialization relationship. Another term is “typological hierarchy”.

Generalization and composition hierarchies are often confused. For example, the taxonomic hierarchy of non-extinct (recent) organisms on earth is a generalization hierarchy (whether it is phylogenetic or based on arbitrarily selected generalizations). A species concept is a refinement of the more general genus concept, or the genus is a generalization of all species it contains. However, when illustrating the taxonomy, a tree will be drawn where genus and species are arranged in space (as a tree, as headings in a text, or as nested boxes). These visualizations may lead to the erroneous intuition that the species is a “part-of” the genus “container”.

An abstract geometrical example and associated UML class model for object generalization and composition is shown in Fig. 70. The composition and the generalization hierarchy are independent. Biological Examples of objects that occur multiple times on various parts of individuals are hairs, finger and toe nails, or various kinds of spores or leaves (e. g., Fig. 52, p. 137). Often more relevant are, however, the cases where related parts have different names and somewhat different definitions in different taxonomic classes. These parts remain comparable (e. g., for phylogenetic analysis or identification) only through generalizations.

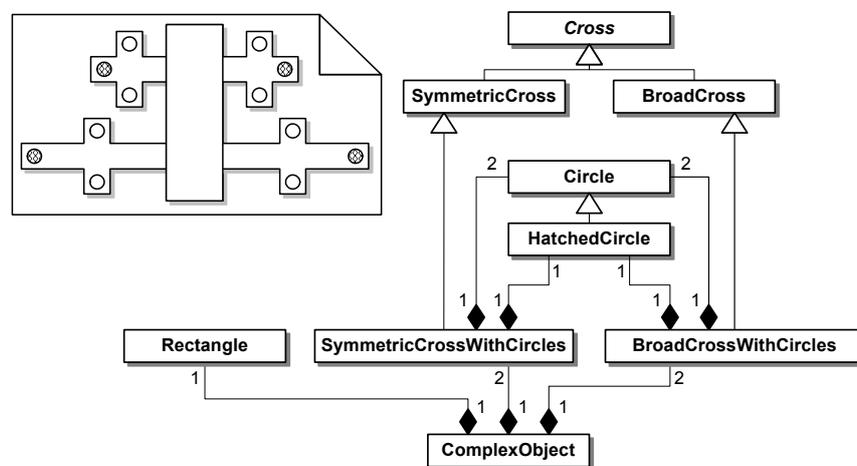


Figure 70. UML class diagram showing an object generalization (white triangles, top to bottom) and composition (black diamonds, bottom to top) model for the geometric object shown in the note shape.

Whether a hierarchy is a composition or a generalization may be ambiguous. An example is the anatomical composition hierarchy of the major organ systems of the human body (Fig. 71). The parts are distributed through various parts of the body, but because they are generally connected to each other it is plausible to assume that they form a composition (part-of relations). On closer examination, however, not all bones or muscles of the musculoskeletal system are directly connected, making it dependent on other parts not in the system. In the case of the endocrine system, the parts are not only often disconnected, but also belong to other organ systems: testicles (reproductive system) produce testosterone, kidneys (excretory system) produce renin, and the hypothalamus (nervous system) produces corticotropin-releasing hormone (CRH) and many

other hormones. Clearly the endocrine system is not appropriately described as a composition, where each part may be member of only a single composite object. It may perhaps be modeled as a UML aggregation (this would be depicted with a white diamond instead of the black composition diamond). Alternatively, a generalization hierarchy may be more appropriate: skin is a kind of integumentary system, liver is a kind of endocrine system (Fig. 71 right). However, the concept of “system” strongly suggests composition; “shoulder bone is a kind of musculoskeletal system” seems the wrong perspective. It may be desirable to create a mixture of the interpretations shown in the left and right side of Fig. 71, using composition for all unambiguously assigned elements, aggregation for parts belonging to multiple objects, and generalization for the endocrine system.

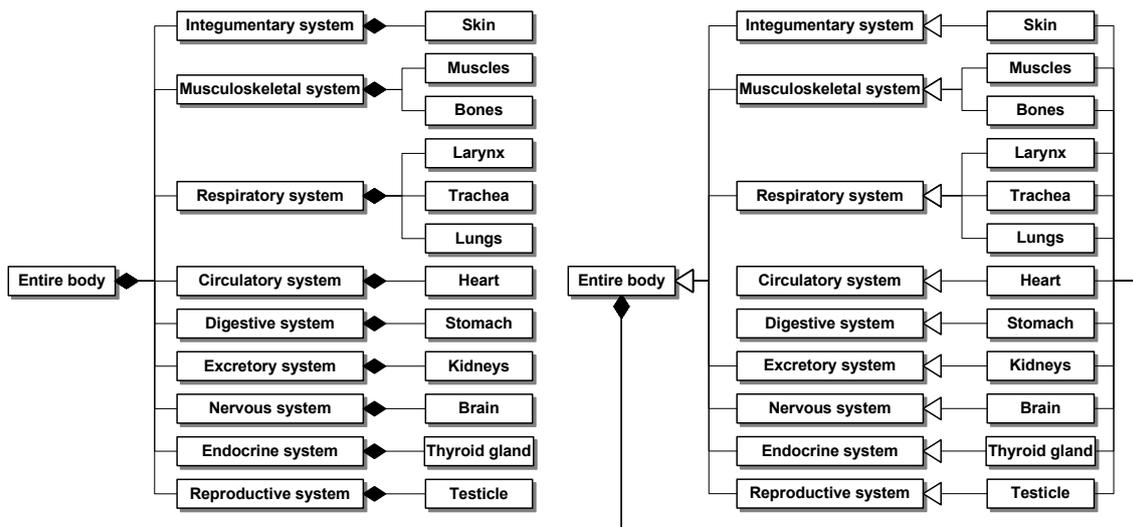


Figure 71. UML class diagrams showing the nine major organ systems of the human body with examples of organs for each system. The left diagram interprets the organ systems as an anatomical composition hierarchy. The right diagram shows primarily a generalization with a more general composition added. Note: multiplicities are omitted from this and some of the following diagrams.

In some cases it seems that composition relations (part-of) may also have a generalization (kind-of) quality. For example, in the case of a surface hierarchy (upper/lower surface, margin, Fig. 72), both “margin is part of the surface” and “margin is a kind of surface” make sense. Fig. 73 adds a specific leaf context to the surface terms (as subclasses of the general terms). It is now no longer entirely clear, whether the specific leaf concepts inherit their compositional qualities from the more general classes, or whether the general classes have two implicit concepts for which the same names are used.

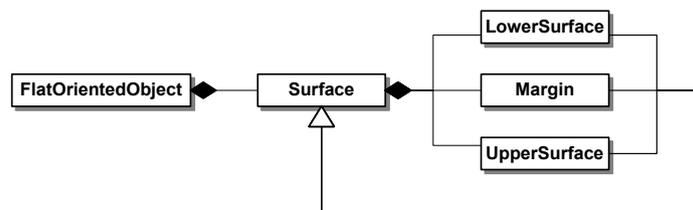


Figure 72. UML class diagram modeling the surface (e. g., of a plant leaf). Surfaces of flat oriented objects can be decomposed into a lower, upper, and marginal surface (composition, black diamond). Independently, each part is also a kind of surface (generalization, white triangle).

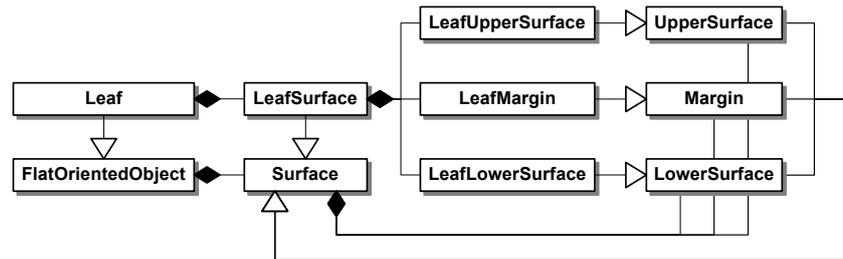


Figure 73. UML class diagram adding a leaf context to the surface abstraction shown in Fig. 72.

159. Generalization and composition are distinct forms of relations that have different properties and lead to different conclusions. For physical objects (parts of the described objects) the information model must support both a composition and generalization hierarchy.

Generalization concepts for object parts

Similar to the multiple composition hierarchies discussed above, multiple generalization concepts exist. Examples are morphological (e. g., “arm and leg to extremities”) and histological tissue types (e. g., epithelial tissues, muscle tissues, nerve tissues, and connective tissues). The following generalization concepts are of special interest in descriptive information models for biology:

- phylogenetic homology,
- functional similarity,
- appearance or morphological similarity (things look similar),
- compositional similarity (things consist of similar parts down to chemical compounds).

Generalizations may combine several of these aspects (Table 42); it is unclear how this can be expressed.

Table 42. Examples from biology for generalizations of object parts under three different generalization concepts.

Object part ¹ :	Function:	Phylogenetic origin:	
		Identical (homologous)	Not identical (analogous or heterologous)
Similar	Similar	pairing chromosomes; basal leaf and stem leaf → leaf, conidia of hyphomycetes and urediniospores of rusts → mitospores; different pheromones of sister species (with minor chemical modification)	conidia (“mitospores”) and meiospores → diaspores; leaf and phylloclade → leaf-like structure; mimicry systems like warning coloration (Müllerian or Batesian mimicry)
	Dissimilar	mandibles of normal beetles versus stag beetles (<i>Lucanus cervus</i>) → eating versus sexual attraction; tail feathers of pheasant versus male peacock → flying versus sexual attraction (function often changes morphology beyond immediate similarity, so these cases are rare)	plant twigs and walking sticks (Phasmida, using camouflage)
Dissimilar	Similar	human and elephant toenails; leaves of woodruff (<i>Galium verum</i> , needle-like) and of horse chestnut (large palmate)	eyes of mammals or snails; wings of birds, bat and insects; beak of woodpeckers and elongated third finger of aye-aye (<i>Daubentonia madagascariensis</i> , a squirrel-like primate), both gathering wood-boring insects; spines (derived from leaves) and prickles/thorns (derived from stem tissue)
	Dissimilar	Photosynthetic leaves and scales on a rhizome; flipper of seals and human arm. Middle ear of mammals: malleus (hammer), incus (anvil), and stapes (stirrup)	(pairs without generalization: perhaps animal eye and plant root)

¹ Compositional or structural similarity = morphological, anatomical, or chemical composition.

The phylogenetic homology of object parts is a central aspect of many evolutionary or phylogenetic studies. Identifying this homology is necessary both to infer a phylogeny based on descriptive data specific to parts (mostly morpho-anatomical data, but also organ-specific protein expression or gene regulation patterns), and to reconstruct individual characteristics based on a known phylogeny (based on other, e. g., molecular characters). In many cases phylogenetic homology is already embedded in common composition (and property) concepts in biology. However, the information model for descriptive data should not be restricted to data fulfilling homology assumptions. For example, although data expressed in NEXUS (which was developed for phylogenetic purposes) will usually contain homologous characters, NEXUS is also used in the context of Linnaeus II identifications, where this assumption does not hold.

Functional similarity generalizations without respect to homology are of interest, for example, when studying the geographical or ecological distribution of functional characteristics, when studying correlations of functions in organism communities, or when studying the evolution of functional characteristics by mapping them to a known phylogeny. Examples of functional generalizations used in the study of DNA sequences are: (a) transcribed or non-transcribed; (b) protein-coding, rRNA-coding, or non coding; (c) intron/insert, or exon; (d) conserved, variable, or hypervariable; (e) structural or regulatory; or (f) monomorphic or polymorphic (i. e., single or multiple alleles in population). Most of these classifications overlap considerably, and preference for a certain classification depends strongly on the purpose of the user of the data.

In some cases, the question whether a generalization is a homology or not, may even be debatable and depend on the perspective. For example, the genes within gene families (e. g., myoglobin, α - and β -hemoglobins belong to the globin gene family) are homologous insofar as they are assumed to be derived from a common ancestor gene within the genome, and the homology assumption is meaningful for the purpose of gene-phylogenies. However, for the purpose of phylogenetic inference of the organism, the different members of a gene family are not homologous.

Appearance and compositional similarity are the most important aspects for identification. This is only partly about achieving error tolerance by generalizing object parts likely to be confused during identification.

The difference between appearance and compositional similarity, as proposed here, is frequently blurred because the aspects often occur together. However, for example, color may be a result of dissimilar structures (object compositions) such as pigments or structures causing the creation of “physical colors” by means of interference (the colors in the wings of certain dragonflies or butterflies like *Morpho* are the result of the same effect that colors thin oil films swimming on water). Conversely, compositional similarity may be difficult to judge under the aspect of morpho-anatomical appearance. In molecular biology compositional similarity is the basis of the design of PCR primers or oligonucleotide probes for DNA microarrays. Both methods depend on the similarity of nucleotide sequences, not on their homology.

In contrast to composition hierarchies, generalization hierarchies can, in principle, always be joined into a single directed acyclic graph. However, when forcing the hierarchy to be a tree rather than a directed acyclic graph, information may be lost. Generalization trees can usually only be joined at the top, whereas in reality more direct generalizations would be possible.

160. Multiple generalization perspectives exist (e. g., phylogenetic, functional, morphological similarity, or compositional similarity) and must be supported in the model.
161. If generalization hierarchies support directed acyclic graphs, a single graph may incorporate the generalization hierarchies for multiple perspectives. However, for the clarity of expressions clearly labeled separate graphs may be preferable.

Problems with specialized, context-dependent names for object parts

Specialization is the opposite of generalization and after discussing the latter it may appear redundant to discuss problems of specialized part names. However, as already introduced earlier, one of the most fundamental problems of managing descriptive data is the reciprocal dependency between recognition of object properties and parts (compare Figs. 8-9, p. 37). In the context of object identification therefore an important difference exists between

- parts that can be recognized (arms and legs) and for which also one or several generalization perspectives exist, and
- parts that are initially recognized on a generalized level, but specific names or concepts are normally used in descriptions.

Descriptive terminology often uses terms for object parts that encapsulate knowledge that is difficult or impossible to obtain during routine identifications of biological organisms. In most cases the knowledge is not customarily obtained in a prescribed analytical process, but supplied in retrospective, after the organism itself has been recognized (“post recognition problem”). Until then, the specific terms for organism parts are used in the mind of the identifying person as “potential terms” that are used operationally only on a generalized level, i. e., only the properties of the generalized terms are used when testing concept hypotheses. The following situations may be distinguished:

- The correct term for an object part (or “structure”) depends on studying properties that are difficult to observe, either in the part itself (e. g., anatomical properties), or in other parts of the organism. Examples:
 - Spines, prickles, thorns (see Table 39, p. 136), or cladodes (i. e., phylloclades) and leaves are morphological concepts that may require anatomical or developmental studies for differentiation.
 - The difference between a leaf and a leaflet is recognized after the compound nature of the leaf is recognized first. In plants without stipules and axillary buds, this distinction often poses a substantial problem for the general public when identifying a plant.
 - A name is occasionally modified if multiple similar structures occur on the same organism. For example, asexual spores in fungi are normally called “conidia”, but are called “microconidia”, “mesoconidia”, and “macroconidia” if two or three kinds of conidia of strongly different size are produced. This can only be recognized if both conidial types are produced concurrently (which is not necessarily the case) or, more commonly, if the species is already tentatively pre-identified to genus level and the existence of multiple spore types in that genus is known.
- Highly similar object parts may have different names in different generations of a life cycle or sexual stage. Examples:
 - In most fungal groups sexual and asexual spores (conidia) are named differently. Ascomycetes often have multiple spore types. The spores resulting from the sexual process are called “*ascospores*”, the asexual (“vegetative”) propagules are variously called “*conidia*”, “*conidiospores*”, or “*mitospores*”. Sometimes a third class of spores, the “*spermatia*”, can be observed that are assumed to transfer the haploid genome to another individual for the purpose of mating. Occasionally, a single species may even have “*synanamorphs*”: multiple, differently shape asexual propagules, often generated in different conidiomata. These object parts can be classified under different aspects (Table 43).
 - Probably, the “world record” in stage-specific naming is held by the rust fungi (Uredinales, a group of obligate plant pathogens). According to their developmental stages, *spermatia*, *aeciospores*, *urediniospores*, *teliospores*, and *basidiospores* are distinguished. A secondary problem is that synonymous names are in current use (e. g., *spermatia/pycno-/pycniospores/spore state 0*, *aeciospores/spore state I*, *uredinio-/uredospores/spore state II*, *telio-/teleutospores/spore state III*, and *basidiospores/spore state IV*). As long as they use comparable concepts, they can be easily synonymized and standardized (e. g., following Kirk & al.

2001 as above); the concepts may, however, differ in complex ways (M. Scholler, pers. comm.). The principle problem of different spore states, however, expresses biological knowledge. Only some of these spore types are relatively easily distinguished morphologically (spermatia, basidiospores); others may be difficult to distinguish prior to identification (especially aeciospores vs. urediniospores). Interestingly, this problem can in practice often be avoided by using another character first: Many rust fungi form the different life cycle stages on different host plants. While this knowledge can be included into an authored branching key, it is difficult to express this in a general information model, allowing machines to recognize and handle these and similar situations of context-dependency.

- In some groups of heterobasidiomycetes producing repetitive ballistospores, sexual and asexual spores are fully identical and can be distinguished only by observing their origin.
- The name of object parts may be specific to a higher taxonomic group. Examples:
 - The sexual spores are called ascospores, basidiospores, and zygosporangia in Ascomycetes, Basidiomycetes, and Zygomycetes, respectively. Mycologists will readily recognize an ascospore if it is still embedded in an ascus (a typical feature only found in ascomycetes) and a basidiospore if it is connected to a typical holo- or phragmobasidium.
 - Insect larvae are called caterpillars in butterflies, larvae in other groups.
 - The term *haltera* is specific for the reduced hind-wings of Diptera. (Note: The problem of taxon-dependent part names is a general one. The relevant question in the context of the present discussion is whether objects with different names can be recognized in an identification context or not. For example, the difference between a silique of Brassicaceae and a pod of Fabaceae has compositional foundations that can be studied even in the field.)

These three situations may be summarized as a *character, life cycle, and taxonomy dependency* of the correct name for the object part. They require studying the entire organism to increasing depths: neighboring parts, recognizing life cycle stage, and recognizing the taxonomic group. (A related problem that often leads to similar problems in identification is that the correct name for an object part may depend on perspective and customs; compare “Competing classifications of object parts”, p. 136).

It would in principle be possible to replace all the terms given in the above examples with generalized terms. However:

- In most cases, specific terms that include morpho-anatomical, developmental, and phylogenetic knowledge are the accepted consensus in biology. They incorporate a high amount of knowledge (are most expressive) when used in descriptions and work in most cases when used in conventional, printed identification keys (p. 242) or diagnostic descriptions (p. 39).
- As discussed above, different potential generalization perspectives exist. For example, in the case of asco- and basidiospores the generalized term “meiospores” was proposed. This term is based on a combination of phylogenetic and ontogenetic perspectives. In an identification context, however, this term is operationally meaningless, since it is not practical to decide on the meiotic or mitotic status of a cell division. Indeed, the term “meiospore” was accepted only in the 8th edition of the “Dictionary of Fungi” and has been removed again in the following version (Kirk & al. 2001). Other functional generalizations like “ballistospore” (which may refer to sexual basidiospores or asexual conidia) are more suitable, because they are correlated with structural features that make them operationally useful in identification.
- A conflict of interest exists between data storage and specimen identification. The different spore generations of rust fungi are different objects and have properties that need to be recorded separately. Abandoning the specific terms would only lead to descriptive replacement phrases or data structures (e. g., “spores of monokaryotic aecial generation of spermatial function”, etc.), which do not solve the problem. Ultimately, generalization always addresses the tension between uniqueness (individual recognition) and comparison.

The importance of the perspective by which a generalization is informed can clearly be seen in the Plant Structure Ontology (PSO, Ilic & al. 2006). This ontology tries to integrate the different

taxon-specific vocabularies created for model organisms in molecular studies (like *Arabidopsis*, *Zea*, *Oryza*). PSO recognizes that silique and caryopsis are types (i. e. specializations) of fruit, but nevertheless treats all of “silique”, “caryopsis”, “grain”, and “kernel” as synonyms of “fruit”. This approach is a deliberate simplification guided by the gene-annotation use case of the PSO. Unfortunately, it implies that PSO is not usable for descriptive data, and a separate version has to be created for these purposes. When creating such an ontology, it may be desirable to annotate whether a specialization is easily recognizable in identification scenarios or not.

Some important differences exist between conventional printed keys and computer-aided identification tools using multi-access keys with respect to the names of object parts:

- Printed keys (branching keys, e. g., dichotomous) occasionally use appearance generalizations (e. g., “spiny structures present”), but often they do not. However, care for the problem is usually implied. It is highly unlikely that a question in an identification key is “do you have prickles, spines, or thorns?”, or that, when asking “spines present/spines absent” the user is expected to first study whether it indeed is a spine, and having found a thorn, follow the lead under “spines absent”. Instead, the question: “spiny or not” would normally be interpreted to imply that only spines may occur at this place in the key. Ultimately, humans performing the identification rely on this convention and provide generalization knowledge themselves, i. e., when asked for a spine they compare the object primarily against the generalized and not the specific description of a spine.
- In a multi-access key, identification can start with any character and no sequential context can be designed into the key. The need to explicitly use terms generalizing for appearance similarity is much more urgent here, e. g., supporting “I do not know which type of spore I have, but it has the following size, shape and color ...” Generalization hierarchies on top of the concept terms used for data storage may be a solution to this problem (compare “Concept hierarchies”, p. 125). If the concepts of object parts like the different spore concepts can be generalized into “spore”, it might be possible to also search for properties common to multiple members of the generalization (shape, color, size) and use them to support generalized questions based on data that are recorded in a more specific system. Other properties (e. g., type of conidiogenesis) may be specific to the precise kind of spore and not generalizable.

Table 43. Examples of competing classifications of spores and sporogenous cells in ascomycetes.

	Composition (depends on taxonomy and life history)	Classification according to Sexual function (depends on meiosis/mitotic state)	Similarity = observable morphological features
Ascospore	Ascus	Postzygotic sexual propagule	Spore
Conidium	Hypha or – if present – conidioma	Asexual propagule	Spore
Spermatium	Hypha or – if present – conidioma	Prezygotic sexual propagule (gamete)	Spore
Ascus	Ascoma (sexual fruiting body)	Sexual structure	Sporogenous cell
Conidio-genous cell	Hypha or – if present – conidioma	Asexual structure	Sporogenous cell
Spermatio-genous cell	Hypha or – if present – conidioma	Sexual structure	Sporogenous cell

Note: Classification according to the morphological composition hierarchy implies recognition of the surrounding tissue, sexual state (life cycle stage) and taxonomic group, whereas the direct morphological classification assumes no additional knowledge.

162. The recognition of object parts and the preferred name for these in biology depends in complex ways on character properties, developmental stages, and the taxonomic classification of the organism. In an identification context this information is often available only after identification success (“post-recognition problem”). Support for generalization concepts to overcome this problem is required.

Misinterpretation of object parts

One result of the problem to correctly identify object parts is that many parts of biological organisms are notoriously prone to misinterpretation by non-experts. Good examples are plants where flat and green stems function as leaves (cladodes, = cladophylls, = phylloclades) and true leaves are missing or not apparent (e. g., *Asparagus*, *Tillandsia usneoides*, many Cactaceae, and Asclepiadaceae), where the petiole has replaced the leaf blade and looks astonishingly leaf-like (e. g., as in many pacific *Acacia*), or entire inflorescences that look like flowers (e. g., as in Asteraceae, also known as Compositae). Other cases are more confusing and more difficult to recognize. For example, the flowers of *Euphorbia* are strongly reduced and unisexual, but the entire inflorescence (the cyathium) looks remarkably like a flower. As a result, characters will often be misinterpreted because the object part is misinterpreted (i. e., not the state). This applies to categorical data (example: flowers unisexual = true/bisexual = by misinterpretation) as well as quantitative data (example: length of flower stalks = peduncle; often the length of the stalk bearing the cyathium will be measured here).

The case of misinterpreted categorical characters can be more or less satisfactorily handled by the addition of a status modifier “By misinterpretation” to the state data. Such a modifier is implemented in the information model used by the CBIT Lucid programs (p. 21). It is also supported in the SDD modifier system (see “Misinterpretation hints through modifiers”, p. 209). The drawback of this solution is that while the cause of misinterpretation is based on a confusion of object parts, the solution works on the value (categorical state) level, i. e., all properties for the object part in all taxon descriptions must be appropriately re-coded. Furthermore, the reason for the suspected misinterpretation is not explained. The modifier does not distinguish between misinterpretations due to state misinterpretation (this may e. g., occur in the case of color states) and compositional misinterpretation as in the case of the *Euphorbia* example.

No equivalent solution is proposed for quantitative measurement data that may be similarly misplaced.

163. Object recognition may fail in predictable patterns; support for knowledge about common misinterpretations of object parts is desirable.

A botanical concept challenge

Similar to some of the data challenges developed in SDD, the following is a concept challenge. It is not solved here, but shown as an example that the creation of ontologies is not trivial:

Hairs on the lower side of a leaf will normally be considered to belong to “Plant → Leaf → Surface → Lower surface → Hairs”. If the hairs occur only on the veins of the leaf, this may or may not be observed, i. e., it may be observed as a separate fact, or the hairiness of the leaf will inherit from vein hairiness and may be recorded as such. If only one of two different hair types (e. g., “short velvety” and “long glandular”) occurs on veins, it seems advisable to differentiate two object parts: “... → Intercostal hairs” and “... → “Costal hairs” (*intercostal* = between the veins). Further, some may consider the veins to belong to the lower surface, because in species studied they are prominently visible only there. The classification may thus have an (optional) added step: “Plant → Leaf → Surface → Lower surface → Veins → Hairs”. Note that veins are internal anatomical object parts of leaves (they contain the vascular bundles) but also cause external morphological surface structures. In many plant species these are more prominent on the upper surface than on the lower surface. Where this is not the case a different classification may be preferred, e. g., “Plant → Leaf → Veins → Surface → Hairs”.

This problem is partly a problem of alternative object compositions (part-of relations). However, it seems that it may be solvable by introducing a combination of composition and generalization relations. Based on the simpler diagram Fig. 73 (p. 155), some additional composition and generalization concepts mentioned above have been added in Fig. 74. Only the general terms are

added, not yet the terms in the leaf context. Hairs have not yet been added at all. It is conceivable to model hairs through a property “hairiness”, although it is a composition, but this could create problem if the shape, structure, or color of hairs shall be recorded later on. The Prometheus description model introduces generalized parts, which may appear without composition in an *ad-hoc*-manner anywhere to deal with problems like hair. It may be noted that the diagram is yet incomplete: for the intercostal surfaces a composition with IntercostalAreas in addition to the generalization should also be added. One point of this exercise is to illustrate that it is not trivial to define both composition and generalization relations (an “ontology”) for a realistic set of component terms. These can probably be overcome by following modeling guidelines and testing the statements with logic processors for contradictions. Note that the Prometheus description model (p. 21) partly tries to circumvent the specific problems shown here by introducing a separate mechanism for region-terms that is outside the compositional hierarchy (and probably also outside the generalization hierarchy). This is further discussed under “Modifiers”, p. 189; compare also the related Fig. 102, p. 201.

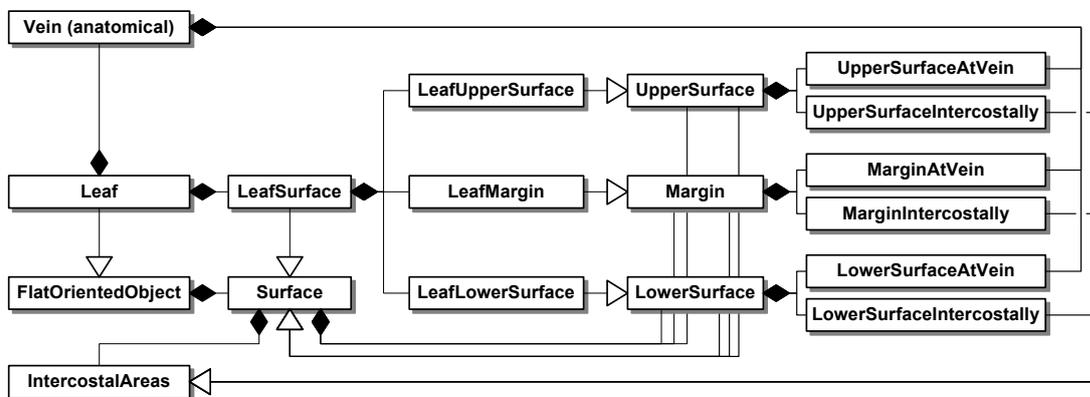


Figure 74. Further detail added to the ontology already shown in Fig. 73.

Taxonomic hierarchy

The “class hierarchy” most familiar to biologists, the hierarchy of biological taxa, has not yet been discussed. In principle, the various generalization aspects (phylogenetic, functional, appearance and compositional similarity) can all be applied to entire organisms as well as to parts of it. The difference between modeling the kinds of leaves occurring on a plant, and the kind of plants occurring on the planet is astonishingly small. Both recognitions are based on a combination of object properties and object composition. At first glance the major difference seems to be that entire organisms are autonomous entities and do not take part in further compositions of higher classes. However, many organisms are indeed part of larger entities and recognizing these may be essential for identification. Examples are colonial organisms like corals or ants. Occasionally, the distinction between individual and colony may be blurred; e. g., in certain colonial jellyfish or in *Dictyostelium* (where up to 100 000 individual amoebae aggregate to create a multicellular organism that is surrounded by an extracellular matrix and forms a sporocarp). Similarly, compositions based on symbiotic relations (mutualism, commensalism, or parasitism) or loose communities (“syntaxonomy”) of different species often play an important role in identification. Examples are lichens (algae plus fungi) or the use of host name in host-specific parasitic fungi.

Sexually or asexually reproducing individuals are the basis for evolutionary mechanisms. Insofar a homology of parts always depends on the phylogenetic taxonomic hierarchy. One consequence of this is that the taxonomic hierarchy often influences the concepts and names of object parts (compare “Problems with specialized, context-dependent names for object parts”, p. 157).

Exploring how far an information model can handle object parts and entire objects homogeneously in a single model may be a fruitful future study. However, in practice modeling the taxonomic hierarchy in a separate model will generally be justified. Taxa are historically been proven to be a central element in the understanding of biodiversity and biology in general, and are therefore governed by a specialized set of rules (nomenclatural rules such as ICBN or ICZN) and have special properties (e. g., taxonomic ranks in the conventional Linnean naming system).

164. The taxonomic hierarchy itself is a generalization hierarchy placing entire organisms in classes. Generalizations of object parts and entire objects are related. Even composition hierarchies of organisms exist. However, despite the similarities, a special data structure for the taxonomic hierarchy is desirable because of the special role taxa play in evolution itself and in the management of biodiversity knowledge.

Change of object concepts through temporal development

Another aspect that is particularly relevant to the description of biological objects is change over time (*develops-from* and *stage-of* relations). This may cause change to the properties of object parts (and consequently the concept of what a part is), but it may also affect the composition of objects. Temporal development in biology has four aspects:

- Behaviorally related changes (e. g., a turtle retracts head and legs under threat),
- continuous ontogenetic change during the development of individuals,
- discontinuous, named life cycle stages, especially if different generations of individuals are involved in a life cycle,
- phylogenetic change over evolutionary time.

In this discussion “changes over time” like deformation, discoloration, or decoloration as a result of specific preservation methods are excluded. These are treated under observation or measurement methodology, see p. 171.

It is interesting to note that UML prior to version 2.0 had no mechanisms to model such information. Mechanisms to model the dynamic behavior of programming objects over time (UML state, sequence, and activity diagrams) exist, but not a built-in modeling pattern to represent a static view of knowledge about potential change of objects.

Continuous ontogenetic change

Almost all object parts change over time during the development, maturation, or senescence of biological organisms. In many cases the changes (growth, slight deterioration when aging, etc.) are so regular and generic that they are not separately mentioned in descriptions. In cases, where unexpected or rare changes occur, the knowledge about temporal development processes is expressed in descriptions in two major ways:

- It may be embedded in the terminology. For example, bud, shoot, cotyledons, and leaves in a plant, larvae, pupae, imago in insects denote certain developmental stages. Some terms like “annual”, “biennial”, or “perennial” embed knowledge about the life cycle of a plant, which can be gathered with certainty only by long-term observation – but which can in many cases be inferred based on other characters (woodiness, development of root system, etc.).
- The changes of object parts or properties are explicitly described: “leaf shape rhomboid in young leaves, later elliptical”, “flowers blue when opening, later red”, or the wandering of eyes that occurs when larval flatfishes (flounder, sole, halibut, etc.) change from symmetrical shape to the asymmetric flattened shape they obtain when settling on the sea bottom. These explicit descriptions of temporal development can either be expressed as modifiers of categorical terminology, or as free-form comment text.

Named life cycle stages and generations

Many organisms have more or less distinct phases in their temporal development called developmental or life cycle stages. These may be an artificial classification of a continuum (e. g., embryo/new-born/infant/youth/adult, or seed/seedling/sapling/adult plant) or associated with distinct events like molting, pupation, change of habitat (free-swimming to sessile, different host organisms), or the changes in nuclear phase (e. g., haploid vs. diploid). The first of these cases is called “growth stages” by Pujar & al. (2006) and defined as “distinct morphological landmarks in a continuous developmental process”.

In many organisms different life cycle stages form no special problem for descriptions because – even though property values like measures of size, proportions, patterns, coloration, etc. may change – the fundamental property and object composition model applies to all stages. Examples are infants and young individuals of most mammalian and avian species, larval stages of ametabolous insects, or the shoot stages of most plants. Describing all in a single description may work in some cases (e. g., adult and senescent animals) but not in all cases. For example, caterpillar instars of a single species or seasonal generations of butterflies like *Araschnia levana* may display substantially different colors and patterns. To avoid overgeneralization, such generations and life cycle stages have to be treated in separate descriptions (requiring a life cycle annotation of the description scope, see “Secondary classification resulting in description scopes”, p. 215).

In other organisms, however, the life cycle stages differ so profoundly that the applicability of property and object composition models depends on the development stage. This is quite similar to dependency of such model on taxonomic groups (and occasionally different sexes). Examples are hemi- and holometabolic insects (e. g., changing from larva over pupa to butterfly), marine hydroids (Cnidaria: Hydroida; changing from swimming planula larvae over sessile polyps forming colonies to swimming, jellyfish-like medusae), anamorphic and teliomorphic stages of many fungi, or ferns (alternating between an inconspicuous prothallus in the haplophase and the diploid fern).

In the context of identification, determining the life cycle stage is usually a post-recognition feature, i. e., it requires identification at least to the level of a higher taxon from which some generalization can be deduced. In many examples, even a specific identification is required. Where different character sets are required to describe life cycle stages, a generalization method may be required for successful identification (compare “Problems with specialized, context-dependent names for object parts”, p. 157).

Phylogenetic change

As mentioned above (p. 153), the taxonomic hierarchy of non-extinct organisms is a generalization hierarchy. However, the development of lineages of evolutionary time (anagenetic evolution) is a part-of hierarchy. A paleontological species may therefore be a part of the evolutionary clade leading to a current species. Since this is difficult to ascertain (the paleontological specimen could belong to a side-line now extinct), these part-of hierarchies are irrelevant in practice.

For all three changes over time, the periods on the time axis (e. g., embryo–adult, larva–pupa–butterfly, Triassic–Jurassic–Cretaceous–Tertiary–Quaternary) form a sequence that can be arranged in a part-of hierarchy. The changes of parts that are correlated with this time axis (e. g., wing sacs to wings) are, however, best viewed as kind-of generalizations.

165. In addition to the common composition (part-of) and generalization (kind-of) relations, relations expressing change over time (ontogenetic, life cycle, evolutionary history) are desirable in descriptive information models designed for biological objects.

Properties

Definitions

The composition and generalization concepts of object parts discussed so far are one half of the character decomposition models (see p. 116). The other half are called properties in the Nemisys/Genisys models (compare Table 9, p. 63; basic property under data type aspect) and the Prometheus description model. In the following an attempt to discuss properties, methods, their interactions and generalizations is made.

The concept of properties (or attributes, the preferred term in UML and ER-modeling) in information science is relatively easy to understand. Variables and constants associated with any given class are considered properties of this class. The type of these variables may be simple, including value types like integer or Boolean, or it may be complex, including any other class. Ultimately, the composition of object instances is defined through class properties. In UML models, relational attributes are usually not shown in the list of class attributes, but only as end names of associations. However, as soon as programming code is created from these models, the associations are converted to properties or association classes containing the necessary properties.

Table 44. Examples of dictionary and UML definitions for attribute, feature, and property.

Term	Collins English Dictionary (CED 1992)	Merriam-Webster's Collegiate / New Oxford Dictionary (EB 2001)	UML 1.5 definitions (OMG 2003)
Attribute (noun)	[...] 2. a property, quality, or feature belonging to or representative of a person or thing. – 3. an object accepted as belonging to a particular office or position. – 4. Grammar. a. an adjective or adjectival phrase. b. an attributive adjective. – 5. Logic. the property, quality, or feature that is affirmed or denied concerning the subject of a proposition. – [From Latin <i>attribuere</i> to associate with, from <i>tribuere</i> to give]	1. a quality or feature regarded as a characteristic or inherent part of someone or something: <i>flexibility and mobility are the key attributes of Britain's army.</i> – 2. a material object recognized as symbolic of a person, especially a conventional object used in art to identify a saint or mythical figure. – 3. Grammar: an attributive adjective or noun. – 4. Statistics: a real property which a statistical analysis is attempting to describe.	A feature within a classifier that describes a range of values that instances of the classifier may hold. (Def. of Classifier is: A mechanism that describes behavioral and structural features. Classifiers include interfaces, classes, data types, and components.)
Feature (noun)	1. any one of the parts of the face, such as the nose, chin, or mouth. – 2. a prominent or distinctive part or aspect, as of a landscape, building, book, etc. [...] – 9. Linguistics. a quality of a linguistic unit at some level of description: grammatical feature; semantic feature. – [From Anglo-French <i>feature</i> , from Latin <i>facere</i> to make]	1. a distinctive attribute or aspect of something: <i>impressed with the safety features of the plant.</i> – 2. (usually features) a part of the face, such as the mouth or eyes, making a significant contribution to its overall appearance. – 3. Linguistics: a distinctive characteristic of a linguistic unit, especially a speech sound or vocabulary item, that serves to distinguish it from others of the same type. [...]	A property, like operation or attribute, which is encapsulated within a classifier, such as an interface, a class, or a data type.
Property (noun)	1. something of value, either tangible, such as land, or intangible, such as patents, copyrights, etc. [...] – 6. a quality, attribute, or distinctive feature of anything, esp. a characteristic attribute such as the density or strength of a material. [...] – [From Old French <i>propriété</i> , from Latin <i>proprius</i> one's own]	[...] an attribute, quality, or characteristic of something: <i>the property of heat to expand metal at uniform rates.</i>	A named value denoting a characteristic of an element. A property has semantic impact. Certain properties are predefined in the UML; others may be user defined.

Unfortunately, when studying the relation between the physical (e. g., biological) world and their description in information models, the definition of the term “property” (Table 44) in relation to the physical world becomes considerably more difficult. Intuitively, it makes sense to say “diameter and shape are properties of the head”. This intuition is certainly valuable. However, such a “property” (or “attribute”, “quality”, “distinctive feature”, “characteristic attribute” accor-

ding to the dictionary definitions) may be obtained by various methods, some of which result in comparable, others in non-comparable data. Both the measurement method and the storage method (data type and measurement units) seem to be tightly coupled with the abstract concept of “properties” themselves.

The discussion will first discuss some aspects of properties under an intuitive concept of “object property”, then study the dependency of descriptive data on measurement methods, and finally attempt to review the relation between property and measurement methods.

Property decomposition

In many cases a property may be complex in the sense that it is possible to express the same information in a combination of other, more atomic properties. It is not possible to generalize that for a given purpose always complex or always atomic properties are preferred. The complex property “inflorescence types” is generally preferred. On the other hand, for object shapes it is customary to decompose the complex shape into a generalized shape plus secondary shape properties. In the case of a square with rounded corners (Fig. 18, p. 65) most humans attempt to abstract the shape as a square for as long as possible, and add the information about the rounded corners separately.

An example from biology is the collection of leaf shapes shown in Fig. 75. Technically, all five shapes are different. However, they would all be described as the abstract shape “elliptical” plus information about the leaf apex (acute, rounded, obtuse to truncate), leaf base (rounded, cordate) and margin (smooth, dentate) described separately.

As discussed already under “Calculated characters” (p. 72), it is desirable to provide mapping or calculations to convert complex to atomic properties and vice versa.

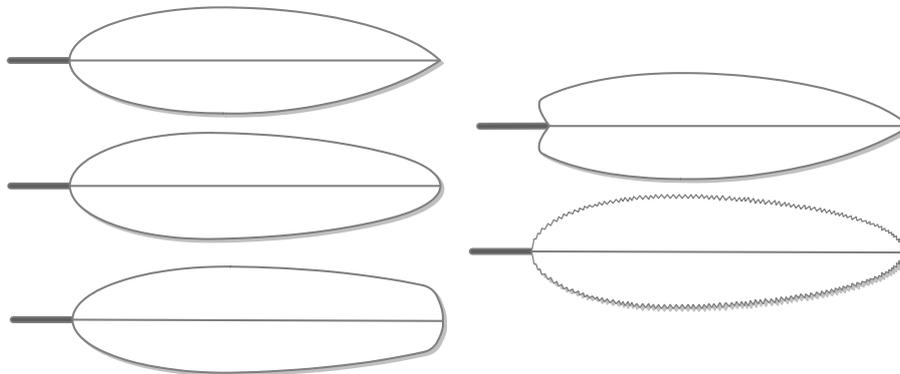


Figure 75. Five different leaf shapes, all of which might be described as the abstract shape “elliptical”, plus separate extension/variant information about tip, base, or margin.

166. What is considered a property is subject to conventions. Complex properties exist that may also be expressed as a set of more atomic properties. A conversion/mapping functionality is desirable.

Pattern versus composition

Any pattern is, by definition, a composition of other elements. In the strict sense, a pattern consists of an arrangement or design that is repeated in itself. In biology it is, however, customary to also speak of a pattern when referring to a unique arrangement occurring only once (e. g., a wing pattern), but being repeated on each individual of a species.

When modeling object descriptions, patterns create problems because it is customary (and efficient for human recognition) to give complex patterns “type names” like “striped”, “checkered”,

“hatched”, “dotted”. In biology, even typical wing patterns, e. g., of the moth families “Geometridae” or “Noctuidae” may be recognized.

Already in Fig. 43 the triangular pattern on the central object was considered an object composition, whereas the striping was considered a pattern. Fig. 76 shows several patterns that can either be described as object compositions or as named pattern categories. The left object is easily described as a composition of a white square with a central gray circle. The second object is more readily described as a square with a regular pattern of gray circles on white background. The use of the pattern approach can still be maintained in the third object (e. g., “alternating rows of gray and black circles on white background”). These named pattern descriptions are more intuitive to humans than an exact enumeration of circles, their color, and which circles are adjacent to each other at which angle (besides that the objects in pattern may easily be too numerous to enumerate them exactly, compare “Categorical multiplicity”, above). However, complex patterns that are a mixture of regular and irregular arrangements (like in the second but last object in Fig. 76) are not easily described.

Furthermore, it is difficult to give guidelines about when to use an explicit object composition and when to use a named pattern category. In the last object in Fig. 76 the circle object can be described as an ordered composition of black and white horizontal lines of specific length. From this description the fact that these lines appear as a circle with a horizontal striping pattern may still be deducible to computer algorithms, but completely opaque to human recognition.

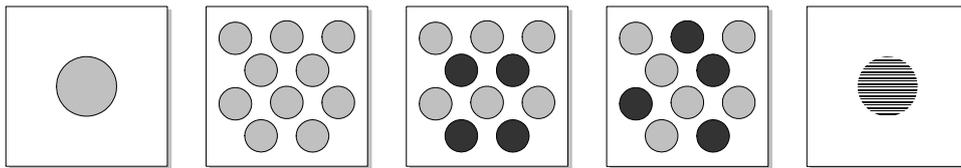


Figure 76. Patterns that can either be described as object compositions or as named pattern categories. The fourth pattern is not repeated in itself (see text for further information).

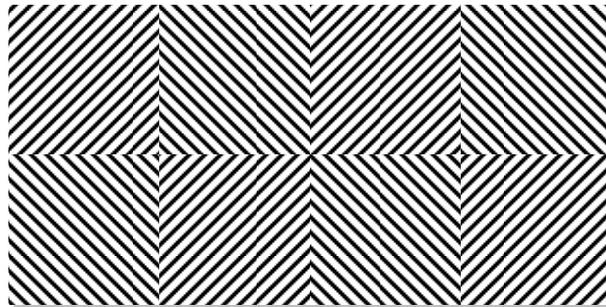


Figure 77. The concept of a pattern abstracts from concrete color to the ability to distinguish parts. Distinction may even occur through surface texture of parts of identical color (here illustrated with a fine line pattern that is intended to symbolize a texture).

A potential solution seems to be to define a library of named pattern types, each of which is defined as an object composition. These object compositions could then be used in the background to allow an algorithm to either give a pattern name for an object composition, or reversely evaluate similarities of patterns. However, a major problem with this approach is that two-dimensional patterns are created by objects of different colors, and that it would be impractical to define a type for each color combination. Similarly to the examples discussed in the section “Property decomposition” above, the concept of “pattern” abstracts a complex reality. The actual reason why components of the pattern are distinguishable (such as color, texture, see Fig. 77) and the absolute size and to lesser extent the shape of the pattern components is considered irrelevant, and only the relative arrangement of pattern components isolated. As a consequence, a “pattern

type library” could be a parameterized object composition where the properties of components (color, texture, etc.) are expressed through parameters that are defined in addition to the type. Although such a solution is conceivable, it seems difficult to implement in OOP languages and requires considerable custom code for analyzing, editing, and reporting descriptions using this concept. Further analysis into this problem is required.

The current approach method of categorizing patterns (used, e. g., in DELTA or NEXUS data sets) has the advantage of not introducing any new concepts or data structures that complicate the information model. It is, however, often unsatisfactory because it creates a dependency between character states of “object pattern” and “object color”. In the presence of a pattern it is customary to list all patterns present in the components of the pattern as object colors, a process in which information is lost (Fig. 78).

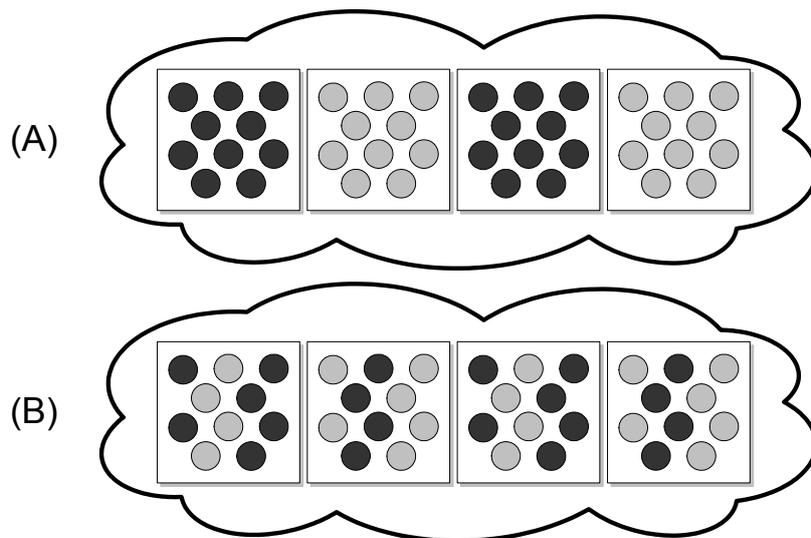


Figure 78. Two populations (clouds) with four patterned objects each. Recording pattern and color as independent properties prevents distinguishing between multiple colors as a result of population variability (A) and co-occurring in a single pattern (B). The class descriptions for populations would be: *pattern “dotted”, colors “white, gray, or black”*.

167. Patterns are especially problematic situations that may be modeled through properties or compositions. Patterns are highly relevant to the description of biological objects and adequate support for them is required.

Property interactions

The previous section already discussed the problems of property interactions in the case of pattern and color. Other cases exist:

1. When measuring the spore size (diameter or length/width), some spores may be halonate (they have a low-density gelatinous outer layer) or they may have appendages. The measurement instructions must define whether the halo or the appendages are to be included in the size measurement or not.
2. When recording length and width of two-dimensional shapes, two values may be redundant where length and width are by definition identical as in circular or square shapes (Fig. 79). The case of globose versus ellipsoidal shapes often occurs in the case of diaspores (seeds, fungal spores, etc.).

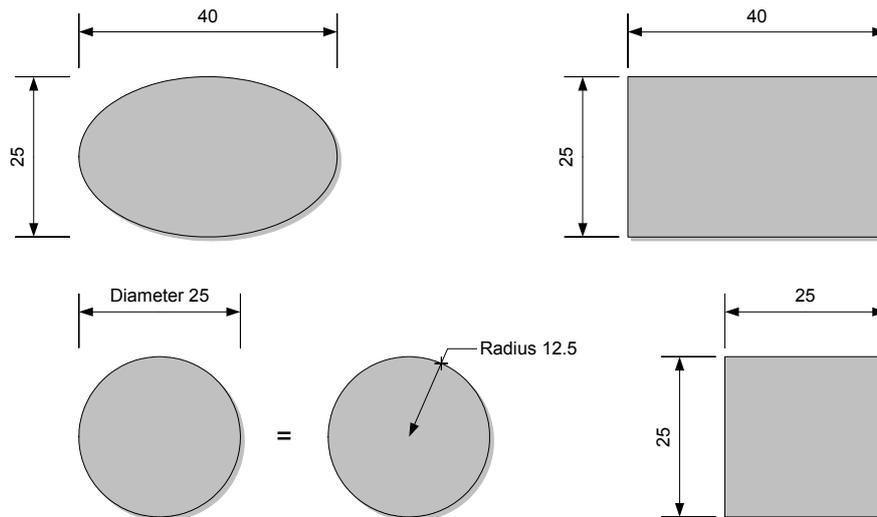


Figure 79. Example for a dependency of shape and size properties. For most two-dimensional shapes a length and width is recorded, but this is redundant where length and width are by definition identical.

Three models of property definitions are commonly found:

- | | | | | | |
|----|-------------|----|-----------------------|----|-----------|
| a) | 1. Length | b) | 1. Length or diameter | c) | 1. Length |
| | 2. Width | | 2. Width | | 2. Width |
| | 3. Diameter | | | | |

In the case a) a dependency from the shape property to the size property can be created such that diameter is applicable only when shape circular etc. In the case c) the length and width are always entered, regardless whether they are identical or not. No model is fully satisfactory. Provided that a system could rely on a rule that data are always either not coded at all, or coded completely, it is relatively simple to represent either of the above cases using the rule: If the width is missing, replace it with a reference to the length (which would return missing if the length is also missing). This assumption is implicitly made in the case of b). However, whereas in the case of fungal spores it is customary to always record both length and width, this is not necessarily true for all objects. For example, for very small petals only the length might be measured, whereas in larger petals both length and width should be measured, In this case the representation b) and the implementation rule given above would assume that small petals are always round with length = width. Furthermore, length and width may occasionally have an absolute orientation in which case width may become larger than length (see p. 169 and Fig. 81).

Informing about the conditions for comparability of the multiple properties in this case is not trivial. An interesting option is perhaps to introduce “label-dependencies” rather than property or character dependencies. For example, if shape is circular, property ‘1’ might be labeled “diameter”, otherwise “length”. Unfortunately, while improving the handling of case b), this does not cover the case of absolute orientation, where it may be desirable to compare a diameter with width as well as length.

3. A related problem occurs where properties or the object composition of sides (e. g., upper versus lower side) of an object may be homogeneous or heterogeneous. In Fig. 80 A) and B)

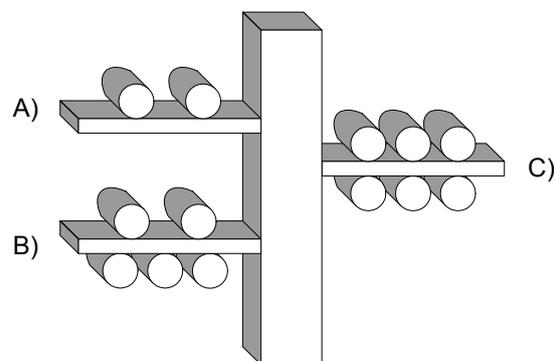


Figure 80. Orientation (upper/lower side) may be significant (A, B) or insignificant (C) when recording object properties (presence and density of circles).

upper and lower side differs and would be separately described. However, in C) both sides are identical and a separate description appears to be redundant. If the Fig. 80 is interpreted as a plant stem with leaves and the circles represent hairs, the case of C would simply be described as “leaves strongly hairy”, rather than “upper side of leaves strongly hairy, lower side of leaves strongly hairy”.

168. Depending on property values, other properties may or may not be applicable. Support for character dependency rules is desirable. This may be in the form of character applicability rules (compare “Character applicability rules”, p. 76) or more general property dependency rules (which would be applicable to multiple objects).
169. Values in different properties may be comparable or not, depending on other property values.
170. In addition to dependency definitions, analogous renaming rules (making labeling dependent on taxonomic scope) may be desirable; this requires further study.

Methods

Introduction

To assume that an object “leaf” has a single property “length” is simplistic. All measurements, whether recorded quantitatively or categorically, require the definition of a measurement procedure or method. For example, when measuring length and width, a definition of orientation is necessary and the start and end points of measurements must be defined (Fig. 81). In leaves, length may be defined as oriented along the axis of the midrib vein, whereas in spores the length is normally defined as the maximum diameter (with width and height oriented perpendicular to it, width being the next-largest extension). However, some spore forms have a clear orientation where the origin (point of conidiogenesis) and a tip can be recognized, and length may be oriented differently here to increase comparability of data.

In the case of quantitative measurements the measurement method usually defines a measurement unit (e. g., mm, cm, or inch) and may inform about precision and accuracy issues (e. g., the number of significant digits).

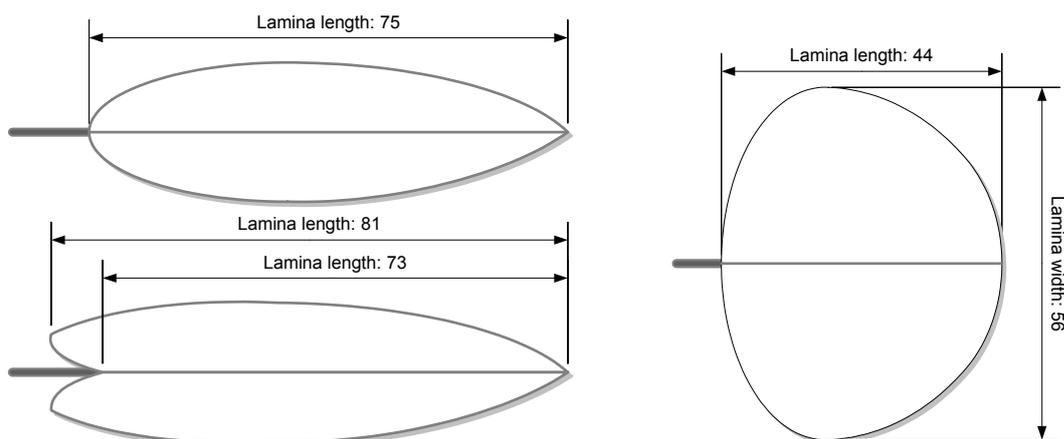


Figure 81. Measurements may result in different results depending on measurement procedures. A procedure to measure the length of the leaf blade may be defined as measuring the distance from end of petiole to tip of lamina or as measuring the longest extension of the leaf blade oriented along the midrib vein. Depending on leaf shape, these result in comparable (top) or incomparable (bottom) values. The example on the right side shows the importance of defining the orientation.

Clearly, to record a value for a property in a description a method or measuring procedure is required. In many cases this method is so well known that it does not seem to deserve a separate mentioning. However, the example just given shows that seemingly simple methods may require explicit method descriptions (compare also the earlier example of measuring total leaf length, Fig. 25, p. 72). Together the methods to record property values and the procedures and rules for object decomposition (see section “Object decomposition”, p. 134) form the description methodology. This may include the following aspects:

1. Specify the *conditions* under which an observation or measurement may be made. These are often generic and trivial: The fresh weight of a plant cannot be obtained from a dried specimen, colors cannot be recorded in fluorescent light. Non-trivial conditions are, e. g., whether an observation can be made on alcohol-preserved material or not, or conditions of cultivation.
2. Specify the tools to be used (“*instrumentation*”). For example, surface structures may be observed using the unaided eye, a hand lens, a light microscope, a scanning or transmission electron microscope, or a scanning probe microscopy – switching between completely different modes of observation, from light to focusable electron beams to magnetic force, electron tunneling, temperature or other forces at close distance.
3. Specify the *operating procedures*. For example, when measuring spores size, the required minimum optical resolution, correct handling of the microscope (e. g., use of Köhler illumination), the number of spores to sample, etc. have to be specified (in addition to “age of fungal culture”, “preparation in water or mounting fluids”, which may perhaps better be handled as conditions).
4. Specify the *conversion and recording procedures*. This includes questions of rounding (to the next mm?), handling of outliers, supported measurement units (international or local units?), statistical aggregation, conversion of measurements to categorical values where requested, etc.

The importance of measurement methods on interpreting descriptive data varies greatly. In many cases, procedures are sufficiently well known or standardized, so that the method can safely be guessed. The modeling of descriptive data has a strong tradition of using well-studied taxonomic groups like vascular plants or larger insects that do not need experimental data for identification and where the majority of characters is observed using the unaided eye or a “standard” 10× hand-lens. As a result, methods have so far not been systematically included in the discussion of descriptive information models. However,

- the more a taxonomic group requires methodically demanding observation techniques (as, e. g., in viruses, bacteria, fungi, or protozoa),
 - the newer the observation methods – being most likely under development – are,
 - the greater the time span is from which data are integrated into a single data set,
 - the greater the set of collaborators is,
 - the greater the taxonomic diversity of organism groups is,
- the greater the diversity of methods will be and the more desirable it becomes to be able to define or record method data. Many of these points apply to large scale descriptive data repositories for identification purposes. In such repositories the traditional close correspondence between secondary data in a descriptive data set and primary scientific publication is quickly becoming fragmented to a degree that makes it impractical to read all pertinent scientific background publication which may shed light on methodologies.

In the followings sections the four aspects outlined above will be discussed in more detail.

171. Even seemingly trivial observations require a definition of an observation method. Support for defining this method – either as a complex method description, or broken down into components such as conditions, instrumentation, operating procedures, and conversion and recording procedures – is highly desirable.

Measurement conditions

Measurement conditions are defined here as the state in which the object has to be at the time when a measurement is taken. This concept summarizes a number of (perhaps disparate) concepts like preservation method, experimental conditions, and measurement preparations. A distinction may be made between direct observations (under specified, but naturally occurring conditions) and observations under conditions set by experiments. An example for experiments performed to obtain descriptive data in biology is the growth of microorganisms under specified carbohydrate, temperature, and light conditions, measuring growth rate, color, etc. (Fig. 82, compare also Table 37, p. 125).

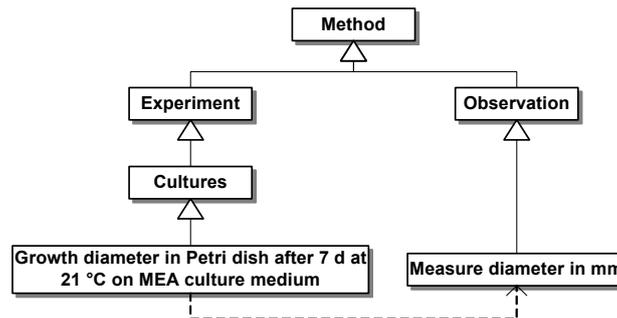


Figure 82. UML class diagram showing methods as a generalization of experiment and observation. The experimental method “growth diameter ...” can be interpreted as depending on the “measure diameter” observation method (dashed arrow). Instead of a dependency this could also be viewed as a composition, i. e., the experimental method includes the observation method, since every experiment contains one or several observations.

Unfortunately, the distinction between experiment and observation is often not clear. Are the different media used to observe cultural characteristics in fungi an experiment, or simple an observation under defined growth conditions? Is the preparation of a microscopic slide – perhaps including some form of squashing – an experiment? What if the preparation is first boiled in lactic acid or KOH (strongly modifying the results, but often still considered an observation rather than an experiment)? What if iodine solution is added, causing a blue amyloid reaction? Procedures such as these may either be considered observation methods (like the use of a microscope itself), or the experimental creation of “morphological” artifacts that are then observed.

A more stable distinction in biology may be between living conditions of the organisms and preparation for measurement after its death. The growth conditions may be experimentally controlled or environmental conditions may be specified. For example, when measuring typical plant size the method implicitly contains the knowledge to avoid the stunted plants from wind-swept mountain tops.

Conditions after death are very important due to the importance of long-term preservation of dead organisms for biodiversity research. Here one may want to distinguish between preparations methods that are closely related to a specific set of measurements and usually under the control of the observer, and preservation methods which are often under the control of the museum preserving the specimens and which may be incompletely recorded. The various conservation methods (like desiccation, freezing, alcohol, or formalin) may lead to various changes that influence subsequent measurements substantially (like discolorations, deformations, or destruction of surface structures).

Some methods may profit from “parameterized” definitions. An entire “family” of methods may differ only in a few parametric values, allowing the definition of more generic methods. For example, the method to measure growth diameter of a fungus could have the parameters “culture duration”, “cultivation temperature”, and “cultivation medium” (Table 37, p. 125).

172. The separation between experimental conditions, conditions under which material is sampled from the natural environment, and conservation or special sample processing conditions is not always sharp. It may therefore be desirable to support information in a generalized “measurement conditions” category.

Instrumentation

Instrumentation may be separated from the measuring methodology because the same instrument is often used for a large variety of measurements. An instrument normally comes with a set of default operating procedures and a domain of measurement possibilities. By modeling instrumentation as a separate dimension of methodology this information can be centralized.

Instrumentation is perhaps even more relevant for complex data types (representing color, shape, molecular patterns, etc.; compare p. 59), recorded or produced using specialized instruments, than for classical data types (categorical or quantitative measurements). In principle, instrumentation also applies to digital recordings of media data (images, audio, or video). However, here instrumentation is often fairly standardized and many relevant metadata may be embedded in media data (e. g., resolution, white balance method, camera model). Perhaps the most important aspect of instrumentation for images and videos that is not commonly supported is the combined scale resulting, e. g., from a combination of a microscope and a digital camera.

An important practical aspect of instrumentation is that instrumentation often is the decisive hierarchy during identification to determine which potentially useful characters are actually available, e. g., when in the field (Fig. 83). Presenting characters organized by the instrumentation required for observation (e. g., unaided eye, hand lens, light microscope, scanning electron microscope, transmission electron microscope, molecular studies, biochemical studies, cultural studies, etc.) will often be as useful as presenting them organized by the property that is observed (any color, any shape, etc.).

It will often be desirable to include higher categories (such as molecular laboratory for PCR or DNA sequencing, sterile equipment to culture fungi) on top of specific instrumentation concepts. In practice these categories may perhaps be viewed as generalized instrumentation concepts. In a careful representation of reality, however, they are probably a composition hierarchy. For example, both light and electron microscope are a kind of microscope, but only the light microscope will be part of the equipment in a “field laboratory” concept.

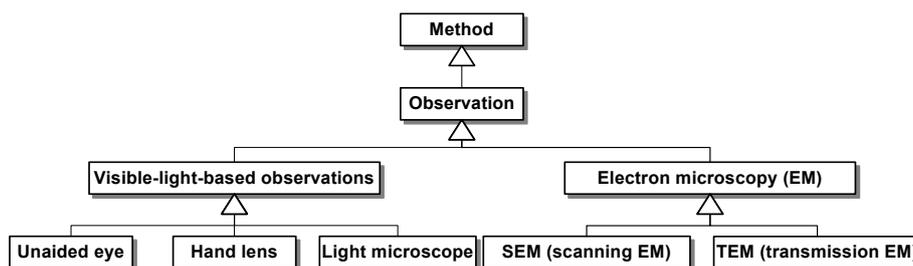


Figure 83. UML class diagram showing a selection from a generalization hierarchy of observation methods.

A complex relationship exists between properties and instrumentation. Properties can be observed by multiple instrumentation methods, and instruments can be used to observe multiple properties (Fig. 84).

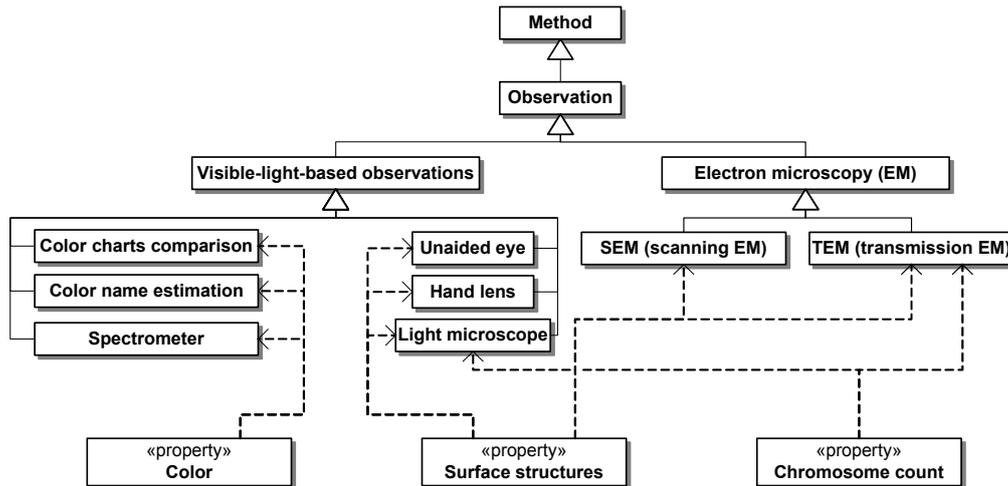


Figure 84. UML class diagram showing that properties can be observed only by specific methods (shown as dashed dependency arrows). The «property» stereotype is not part of a UML profile, but introduced for the purpose of this discussion.

173. Support for “instrumentation” concepts, a highly reusable part of methodology, is desirable.
174. The concept of instrumentation may include basic default operating procedures.
175. Generalization and composition hierarchies of instrumentation (such as “forestry field instrumentation”) are desirable.
176. Complex relations exist between instrumentation; it may be desirable to model these dependency relationships unless this leads to an overly complicated information model.

Measurement procedures

Often specific measurement procedures are necessary for each:

- property (e. g., size or color),
- class of objects or object parts (see leaf length example, Fig. 81),
- class of measurement conditions and instrumentation.

For example, color can be recorded as an approximate color name based on a memorized color (fuzzy color concepts), it can be observed by comparing a color with one of several standard color charts, or it can be measured with a spectrometer. In all cases, the recorded color may depend on the kind of illumination present (color temperature and spectral lines present). In the first cases, being based on human perception, the perceived color will be based on the surrounding color as well as on the color of the object itself.

Measurement procedures for a given property may include different instruments (Fig. 85). The combination of “color estimation” with “unaided eye” and “microscopic observation” results in two different specific methods (multiple inheritance). However, no form of color observation can be combined with an electron microscopical method.

The measurement of color depends both on preparation and instrument use. For example, six different fungal spore colors may be recorded for:

- an observation with unaided eye in situ (thin layer with hymenial hyphae as background),
- a spore print on white paper,
- a heap of dry spores observed with the stereomicroscope,
- a heap of wet spores observed with the stereomicroscope,
- spores observed in the light microscope at 400× magnification,
- spores observed in the light microscope at 945× oil-immersion with *Differential Interference Contrast*.

Another example for the interaction between instrumentation and property is that a “surface roughness” may be expressed in the same categorical enumerated values (e. g., “smooth”, “rough”) but the results of observations by different instruments may be incomparable.

In many cases, the distinction between instrument use and specific measurement procedures may be blurred. An instrument may come with subclasses of operating procedures; for example, in a light microscope for using phase contrast, using oil-immersion, and using a combination of these.

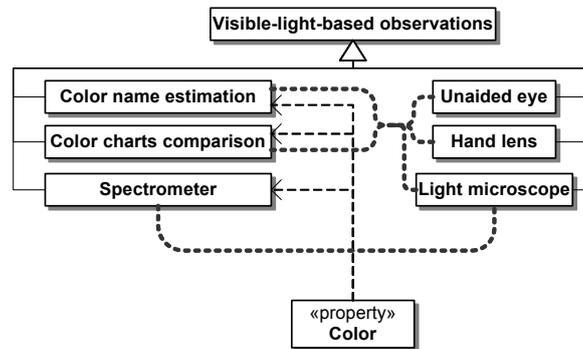


Figure 85. UML class diagram elaborating part of Fig. 84, showing that some methods may be combined to observe a property (shown as thick dashed lines; not a standard UML vocabulary). The first two color observation methods can be combined with all three optical methods, whereas the spectrometer can only be combined with the light microscope (and used on its own). Combinations produce new methods that inherit from multiple parent methods (not shown).

177. In addition to generic concepts of measurement conditions and instrumentation, support for measurement procedures specific to object parts, properties, and interactions with conditions and instrumentation is required.

Conversion and recording procedures

The initial measurement result will often not yet be in a form that can be recorded directly. This includes the normally trivial case of converting it into an appropriate digital format – but even this may require special consideration when automatic measuring instruments are coupled directly with a storage method.

A simple example where measurement and the desired recording format may differ are length measurements. In DELTA the measurement unit is part of the character definition, so that it is not possible to record length, width, and diameter (all considered different “basic properties” in Table 9, p. 63) using different measurement units such as μm , mm, or inch. A conversion is possible before recording them. Other information models (e. g., SDD) support different measurement units in multiple descriptions using the same character. This facilitates data integration from different sources (recording a value of “2 ft” as “60.96 cm” is rather confusing and suggests an entirely wrong precision), but adds complexity and requires consuming applications to be able to dynamically convert units to compare values.

A slightly more complex situation occurs when continuing the color example. Different measurement methods result in different kinds of data, such as a wavelength-intensity curve generated by a spectrometer, categorical color values according to “common sense”, or a category from a given color standard. A multitude of such color standards exist, some using codes (e. g., “5R 6/8” or “5.3R 6.1/14.4” in a Munsell color chart, Munsell 1977, Munsell 1992; “8C6” in “Methuen”, i. e. Kornerup & Wanscher 1967), others names (e. g., “Artemisia green” in Ridgway 1912), or a

combination of numbers and names (e. g., “89. Olivaceous Buff” in Rayner 1970). All five color standards listed above are in current use in mycological studies.

The color chart standard used during recording may be considered a form of instrumentation and it is possible to define specialized properties (all derived from abstract color) for each measurement method. This may be desirable if the recorded data should be as faithful to original data as possible. However, given the complexity that is involved in managing and comparing a multitude of color standards, it may be desirable to standardize recorded data to a single standard. In some cases, manual mappings are available (e. g., Methuen “8C6” and “25C4” are equivalent to Ridgway “Terra cotta” and “Artemisia green”). However, to improve comparability, it may be desirable to map them all to a common quantitative color space model (e. g., the CIE XYZ color model, see http://en.wikipedia.org/wiki/International_Commission_on_Illumination).

Conversion and recording procedures can be viewed as analogous to operations that constrain the modification of class attributes. This is a common development pattern in many OOP languages, and is supported through property methods in .NET or setter/getter functions in Java (especially in the JavaBean pattern). However, since OOP does not deal with the multiple layers of methods required to go from the physical world to reading and writing in a given information model, this analogy is incomplete. On an abstract level, the entire methodology including conditions, instrumentation, operating procedures and conversion and recording procedures may also be viewed as analogous to these OOP property methods.

178. The form of data returned by a measurement method may differ from the form expected in data recording. Support to define and document conversion and recording procedures is desirable.

Dependencies on circumstances of identification

Whether a given property of an object can be observed depends both on the object itself and on the circumstances or conditions under which the observation occurs. Knowledge about this is often embedded in the identification process and data structures supporting this. Examples:

- Identification keys to higher plants may ask “flowers presence?”; a negative answer may lead to a subkey using only non-flowering, vegetative characters (rather than excluding all plants that do flower at some point during their life cycle).
- A key may ask for the season or the presence of leaves if it provides both a summer key based on leaves and a summer key based on buds.
- If identification depends on the availability of certain methods (hand-lens, stereo microscope, light microscope, or availability of chemical reagents for color spot tests used for fungi or lichens), separate keys may be provided (commonly a “field key” and a “full key”, but more flexibility in defining such method dependencies would be desirable).
- Different keys are provided to work with dead, preserved material and fresh or cultivated material.

Because computer-aided multi-access keys (see p. 251) enable their users to ignore characters that are inapplicable to the observation conditions, the importance of some method-dependency mechanism is most urgent for computer-aided branching keys (see p. 247). However, the user experience of multi-access keys will also improve, if fewer inapplicable characters have to be ignored.

The dependency of the applicability of a property on other information is similar to character (or property) dependencies discussed in “Character applicability rules” (p. 76) and “Property interactions” (p. 167). One may model the conditions as controlling “states” of an “observation-condition” controlling character (Fig. 86). This could allow a reuse of the character dependency model for method-dependency.

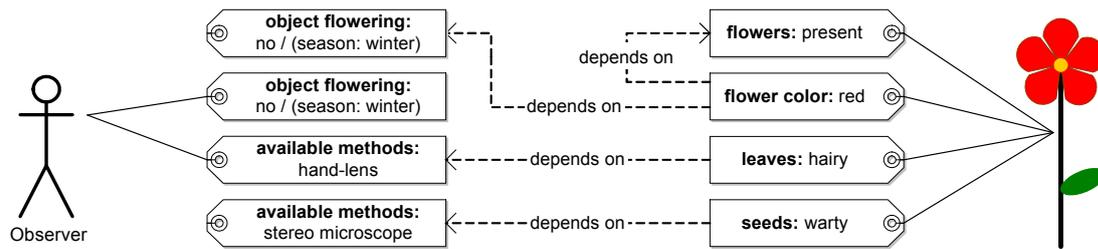


Figure 86. Dependencies of object properties on “states” of the observation situation are structurally very similar to dependencies among different properties (flower color depends on flower presence). Seed wartiness and flowering color are inapplicable in the current situation.

Initially it seems that such observation-condition characters would only be used as controlling characters during identification, but never actually for recording descriptive data. However, they may turn out useful under two aspects:

1. To some extent they may simplify the management of data sets (e. g., informing whether an attempt to study autumn or winter characters of a plant was made). However, scoring a “condition-character” gives no measure about the degree to which condition-specific information has been collected. The same information, plus the distribution of recorded condition-specific characters should also be available by analyzing the data set.
2. In a scenario where the relevant data behind identification events are recorded (e. g., “IdentificationBank/ID-Base”, see use case “Recording identification data”, p. 295 and “Future relevance: A proposal to record identification data”, p. 369), the information in such observation-condition-characters may be worth preserving.

Whether it is advisable to follow such a character-dependency approach, or whether to deal with method dependencies through method concept hierarchies could not be clarified in the current work.

179. Dependencies between the observation circumstances, conditions, instrumentation, etc. and characters available for comparison and identification is an important aspect of the use of descriptive data. It is essential for branching (e. g., dichotomous) keys, and beneficial for multi-access keys.

180. It may be desirable to record “observation circumstances/conditions” as part of archiving identification data. This may point to modeling the method dependency as a character dependency based on special “observation condition” characters. (Alternatively, method dependency may be modeled through the method concept hierarchy/ontology.)

Relations between properties and methods

A precise definition of “properties” was initially purposely avoided (p. 164). It is possible to define properties as abstract, high level concepts of real world objects. To manage the comparability of values in a digital format, however, knowledge of measuring and storing methods is required. Some of these methods (object conditions, instrumentation, recording methods) are reusable. They are therefore not necessarily nested inside the property and may best be represented by composition (Fig. 87). On the other side, a number of method components are directly connected to an abstract property concept (such as “color”).

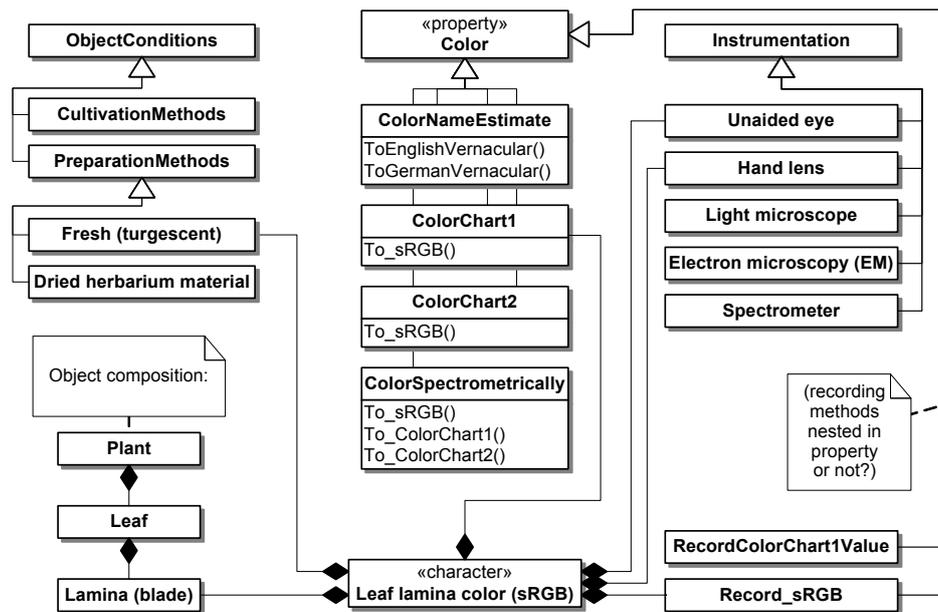


Figure 87. UML class diagram attempting to outline a possible property/method/object-part/character model for the color example. Most classes show neither attributes nor operations; an exception are the specific color measurement classes in the center of the diagram showing examples of conversion operations. The «property» and «character» stereotypes are introduced for the purpose of this discussion.

Comparability does not depend on measurement procedures alone. For example, recording object color as RGB-polygons based on color-calibrated digital images is in principle a fairly direct and reliable measurement method yielding easily comparable data. However, depending on the instrumentation involved, the comparability may suffer: Whereas the color of spores is highly comparable between measurements using the naked eye and a stereo microscope, measurement through a light microscope are not easily comparable with these two. Measurement conditions (e. g., cultivation media, growth time until measurements, preparation or preservation procedures) further influence the comparability of the color measurements. Thus, even measurement methods returning directly comparable raw data may need exact or approximate conversions to obtain comparable data.

The delimitation of *abstract property* concepts seems to be rather difficult. Perhaps an abstract property should generalize all measurement methods that result in potentially comparable value domains. Different abstract properties should be incomparable. Thus, color may be expressed in different formats (categorically, quantitatively in various color spaces) for which conversion functions (mappings) could potentially exist. Similarly, distance seems a good property concept, regardless of the measurement unit (μm , cm, inch) used. However, treating size, length, width, and diameter as independent properties (as in Table 9, p. 63) violates this intuition.

In physics a technique called dimensional analysis attempts to reduce physical properties to their fundamental dimensions (length, mass, time, etc.), to verify the comparability of data. The desire to define high-level properties seem to be related to this, although treating only those properties with a separate dimensional footprint as separate seems to result in impractically abstract properties, at least for all properties that are dimensionless.

Using the framework of Fig. 87, the “basic property” proposal (Diederich 1997, Diederich & al. 1997, Diederich & al. 1998, Table 9, p. 63) can be interpreted as implicitly combining a very abstract form of property concepts with all measurement concepts except the definition of a part from the part hierarchy (Fig. 88). (Figs. 87-88 are intended as examples to discuss the conceptual problems – they should not be interpreted as proposals for descriptive information models).

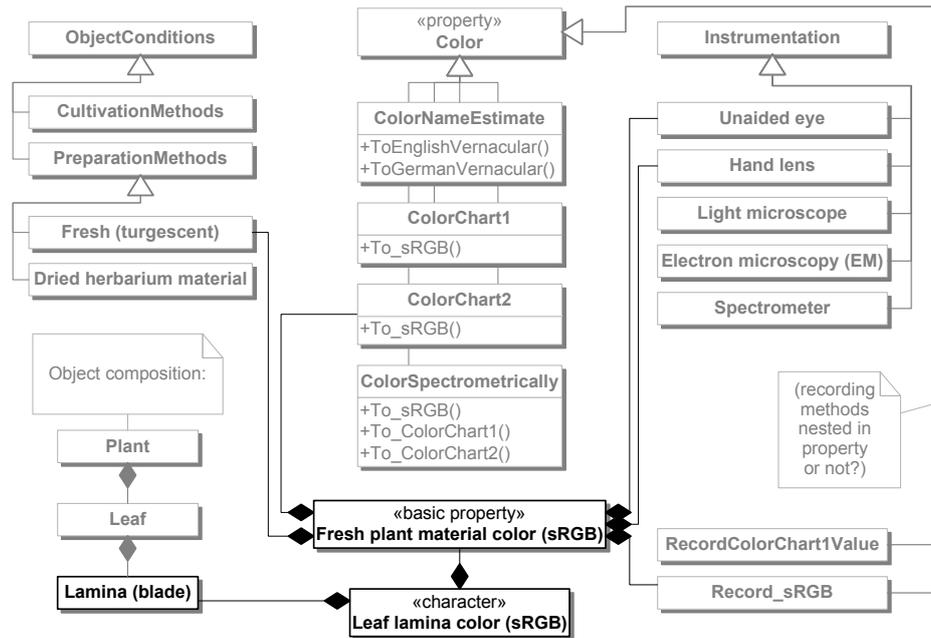


Figure 88. Modified version of Fig. 87 (identical parts are grayed out), adding an interpretation of the “basic property” concept as combining all concept parts of character concepts except the object composition part.

Returning to the overall topic (“Description storage models”, p. 104), a comparison between the distribution of information between terminology and descriptive data storage of the character plus concept hierarchy model (p. 125, SDD) and character decomposition (Nemisys/Genisys, p. 117) model may help to clarify the options how ontological information about object parts, properties, and methods may be realized in actual description models (Fig. 89). Whether a “Part-Property-Method decomposition model” as proposed in Fig. 89 is a successful pattern for descriptive information models requires further analysis and testing with complex data. In this thesis the more conservative first model (characters with superimposed concept hierarchies) is generally preferred.

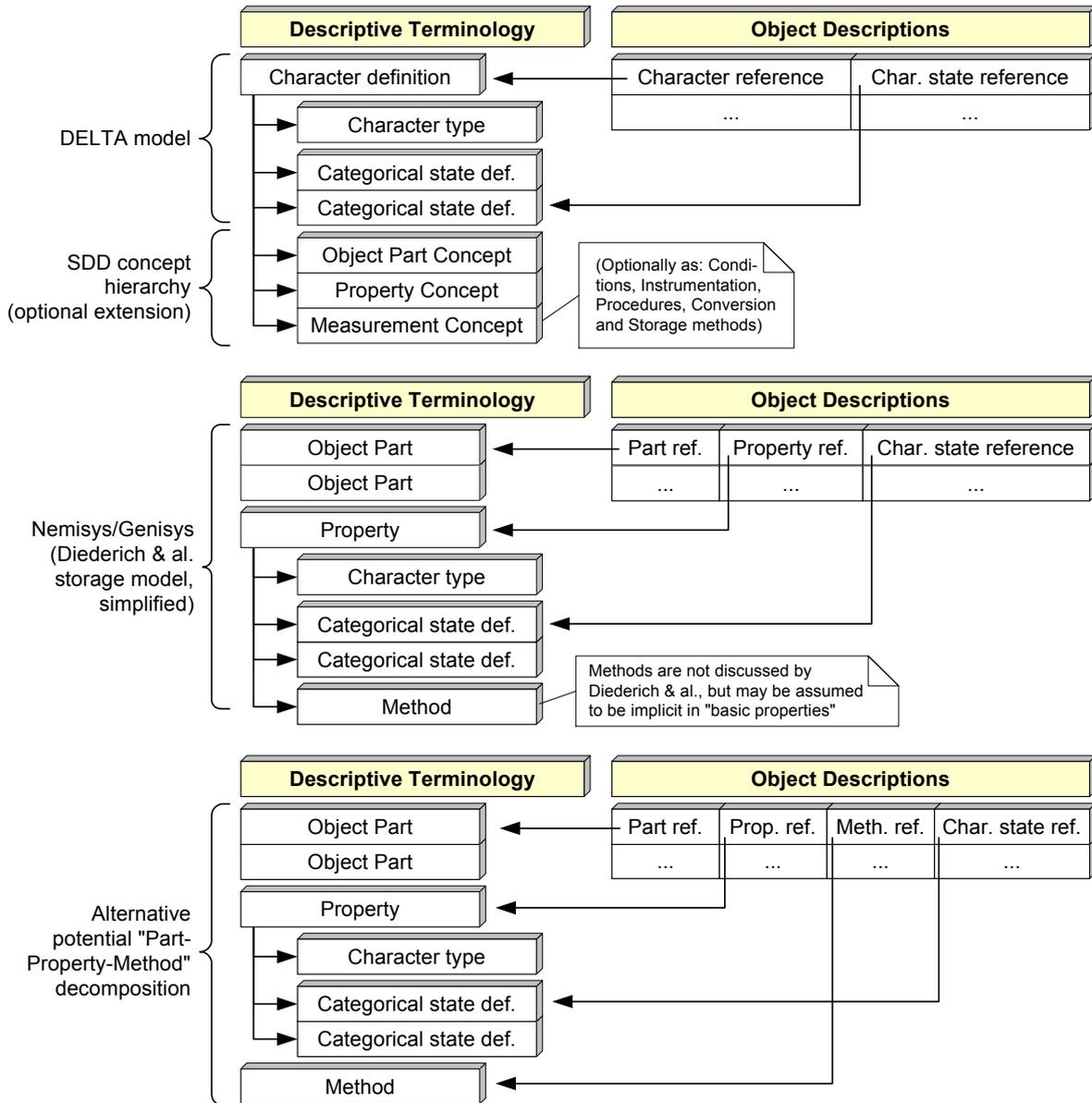


Figure 89. Simplified comparison of a character plus concept hierarchy model (DELTA, SDD), a character decomposition model (Nemisys/Genisys, p. 21), and a potential *part-property-method* decomposition model.

181. Abstract properties and methods interact in complex ways that should be addressed in the information model. Whether a “Part-Property-Method decomposition model” or multiple concept hierarchies superimposed on fixed character concepts are preferable needs further analysis and testing.

4.13. Federation and modularization of terminology

(This and the following two sections are studied in particular detail and therefore presented as separate sections at the end of the fundamental requirement analysis.)

Introduction

As discussed above in the section “Structured descriptions and the concept of terminology” (p. 42), it is unrealistic to assume that a single terminology could be developed that would satisfactorily cover descriptions of large and diverse groups like plants, fungi, or insects. Similarly, the terminology for different methods (e. g., morpho-anatomical, physiological, behavioral, molecular) is typically independently developed, and new terminology for newly developed methods is constantly being introduced. Consequently, existing software for the management of descriptive data enables individual researchers to define the terminology required for their studies.

However, having a large number of independently developed terminologies impedes data integration (as planned, e. g., for the IdentifyLife project, IdentifyLife 2005). A terminology for descriptive data is a kind of schema and it is in principle possible to integrate and correlate any number of terminologies by developing schema mappings and ontologies. Unfortunately, a diversity of opinions exists which term and concept is scientifically the correct one and truly congruent descriptive concepts are rare. As a result, the schema integration will be a laborious task that ultimately leads only to fuzzy relations.

Furthermore, the development of a correct, practical, and stable terminology for feature-rich groups requires substantial effort and time. Examples of attempts to develop a “standard terminology” for vascular plants are the TDWG Descriptors subgroup convened by R. Pankhurst up to ca. 2000, the www.plantontology.org effort (e. g., Pujar & al. 2006 for stages, Ilic & al. 2006 for structure), or the Prometheus description model (p. 21).

It is therefore desirable that the information model for descriptive data itself provide mechanisms to collaborate and share common terminologies. Requirements and models for this are being explored in the following.

182. The information model should support management and curation of the descriptive terminology independently of the descriptive data itself.
183. It is desirable to enable curation of different parts of the terminology by different organizations, in different systems.

Managed federations

The simplest case of federating descriptive data systems is that several projects voluntarily agree to share common resources. One server might supply the terminology, another server images, and several other servers descriptions. This model is especially attractive where institutions form a close collaboration that has a supra-institutional project management. The parts of such a managed federation could either be considered a single project, in which only the physical location of data is federated, or as separate projects in which different people are responsible for the federated project parts.

An information model that supports managed federations is believed to require little extra effort. All parts may simply be structured composition, e. g., in terms of XML documents, the various federated parts are simply included fragments combined to a document. Each description from one of the description servers would be accompanied by the terminology obtained from a central terminology server and by resource objects from the image resource server. The overall management would have to provide mechanisms guaranteeing that each part of the federation fulfills its responsibilities. For example, when improvements in the terminology are required, all description services may have to be informed and take appropriate actions in updating their data,

and the image resource service must provide services for depositing resources required in new descriptions.

The applicability of managed federation models is, however, limited. While optional centralization is desirable, compulsory centralization is not. As Berners-Lee points out: “Traditional knowledge-representation systems typically have been centralized, requiring everyone to share exactly the same definition of common concepts such as ‘parent’ or ‘vehicle’. But central control is stifling, and increasing the size and scope of such a system rapidly becomes unmanageable.” (Berners-Lee & al. 2001). To increase the overall interoperability and the productivity in creating digital descriptive data, it is desirable that description providers may unilaterally decide to use public terminologies without having to enter into a management agreement with the providers of these terminologies. This situation differs from managed federations in that the provider of terminology does not know about the consumers of the data, but must still adhere to rigid design and versioning principles in providing a terminology that can be used as a “standard”. Much of the following discussion addresses this situation. Managed federation projects will, however, also benefit from mechanisms intended to support unmanaged federations of terminology.

184. Supporting managed federations is desirable. This may require some data items supporting management procedures. These are, however, difficult to specify because they strongly depend on local management practices.

Terminology modules

One desirable model for sharing descriptive terminologies seems to be to support a combination of local and external terminology definitions. The local terminology provides flexibility in the case of dissent or new developments, the shared external terminology standardizes descriptive data and simplifies data integration.

To encourage the widespread adoption of standard terminologies it seems further desirable to provide for the concurrent inclusion of multiple external terminology “modules”. Limiting the design of the information model to a single external terminology (Fig. 90 a), would impose an all-or-nothing constraint. The competition between different terminological definitions will be much improved if it is possible to link a description project to multiple terminologies, picking the best part of each (Fig. 90 b). Standardization of terminology would then be the result of voluntary choices and agreement on convergence due to evolutionary processes.

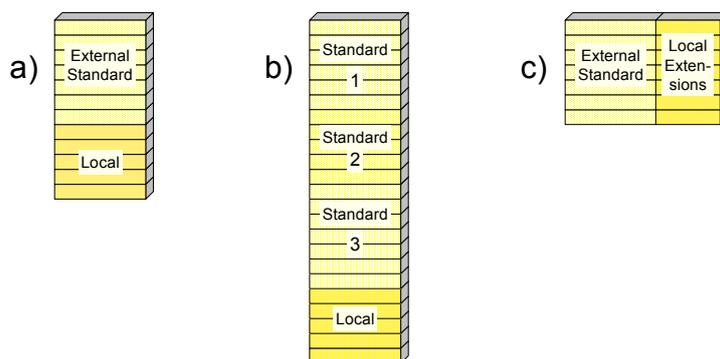


Figure 90. Options for federating, modularizing, and extending descriptive terminologies. a) The project uses a single external standard terminology plus locally defined extensions. b) A modular design integrating multiple standard terminologies and local extensions providing additional terms (characters, etc.). c) The project uses an external standard terminology, each term of which has been locally extended, e. g., to support other languages.

If multiple terminology modules are introduced to the model, the locally defined terminology could either automatically become another module usable by other projects (symmetric design), or it could remain distinct from a terminology module intended for shared use. Only terminology modules explicitly designed as a reusable standard would then be available federation. The advantage might be that such projects presumably are more careful regarding publishing, versioning, and evolving or refactoring their terminologies.

Some potential use cases involving federated terminologies are shown in Fig. 91 (see p. 23 for information on use case notation). An important point is that besides accepting external terminology modules, it may also be desirable to define the relations between one terminology (perhaps a local one) and another. Many terminological definitions in independently developed terminologies may be sufficiently identical for the purpose of data integration and comparison.

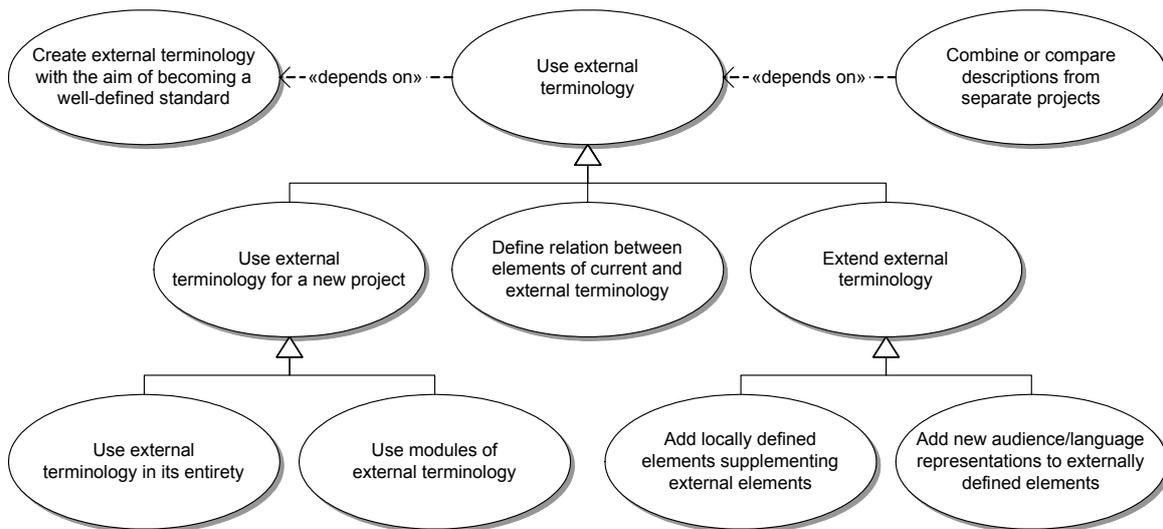


Figure 91. UML use case diagram for some use cases involving external (federated) terminologies. The «depends on» stereotype is not available in standard UML use case diagrams but has been introduced here.

185. It is desirable to support a combination of locally defined and multiple externally defined (standardized) terminology modules.
186. It is desirable to distinguish between locally developed terminology modules proposed for external use, and terminology modules that are considered to be too instable or poorly developed for such use.

Extending shared terminology definitions

Extending external shared terminology modules with local definitions may occur through the definition of additional terms (Fig. 90 a, b), or through extending terms imported from the external standard terminology (Fig. 90 c). The latter case occurs, e. g., when additional language or audience representations are added. Obviously, extending the terminology objects bears the danger of changing the semantics in a way incompatible with the concept of the original term. A discussion about which components of a terminology object may or may not be changed is thus advisable.

The components of terminological objects may be classified into *definitional* (or ‘essential’), *presentational*, and *assumptional*. Only a small part of a terminology is strictly definitional. Examples are the measurement scale of a character or a frequency value range for a frequency

modifier term. Strictly presentational components are the sequence of characters or the wording definitions for natural language reports. An example for an assumption is whether states within a character are assumed to have an inner order (ordinal scale) or not (nominal scale). The assumption is not definitional insofar as the character and its states may be reliably recognized without it. However, it has substantial influence on the outcome of statistical or phylogenetic analyses, and different researchers may want to base their analyses on different assumptions.

The major part of terminological definitions, such as labels and definitional text for concepts, characters, states, etc. are, unfortunately, a mixture of definitional and presentational components. This is most obvious in a multilingual situation. The English representation may be seen as an “international” definition and the other languages as presentations for non-English speakers. However, to those speakers, the other language representations are the only means of being informed about the definition and their coding of data will depend on the local representations, not on the “international” definition.

Whereas few problems arise when centralizing strictly definitional parts of the terminology, it is desirable to be able to locally change (extend or even override) presentational and assumptional parts. The major problem is that no method exists to express semantic definitions of terms independent of language. Even though ontology languages like OWL (McGuinness & van Harmelen 2004) map concepts to language-independent URIs, they still express the ontological concepts only of a specific language. Very few terms in two languages have exactly the same circumscription and can be used interchangeably. For example, the term “bright” may be translated to German: “hell, glänzend, blank, leuchtend, strahlend, klar, durchsichtig, heiter, klug, munter, fröhlich”, all of which have circumscription matching only partly with the English term – “strahlend” may also mean radioactive.

As a consequence, the most central part of the definition will always also include presentational aspects. It would be highly undesirable to centralize all labels and definition text and consider them unchangeable. However, whereas purely presentational or assumptional parts may require changes that contradict the original definitions, the mixed definitional/presentational parts may only require extensions by providing additional languages. If the standard terminology provides English labels and definitions, local copies may add German, Chinese, French, Japanese, Spanish, etc. representations, but may not be permitted to change the centralized English representation locally.

187. It is desirable to support extending external standard terminology modules with local information. The essential definition should not be changed, but it may be extended through labels or definition text in the local language. Furthermore other information affecting presentation or assumptions for analysis purposes may be desirable to extend or change locally.

Terminology modules and class hierarchy

It is conceivable to create a hierarchy of terminology modules (i. e., sets of terminology elements) that follows a taxonomic hierarchy (Fig. 92). Although the model is attractive, it has several limitations:

- Phylogenetic classification is an area of active research, and the taxonomic hierarchy in many biological groups is not stable. Changes in a taxonomic hierarchy that defines usable terminologies would be difficult to implement once thousands of researchers would use such a central terminology system on the internet.
- The characters that are desirable at a higher level for the purpose of identification are not necessarily phylogenetically informative. A purely phylogenetic design of the taxonomy-dependent hierarchy is therefore not possible. For example, the vegetative stage of a fern like *Marsilea quadrifolia* L. may easily be confused with a flowering plant. Thus, even if leaf size

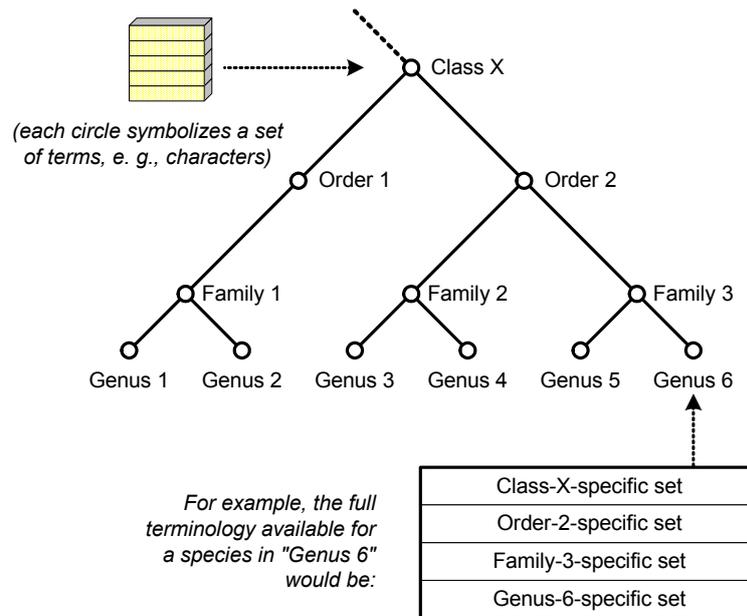


Figure 92. Example for a hierarchy of terminology modules that follows a taxonomic hierarchy. Additional taxonomic ranks may be present above, below, and in between those depicted here. Note that even species-specific terminology would be conceivable, e. g., to distinguish infraspecific taxa.

and shape are too variable to be used for phylogenetic purposes, it is desirable to have them at a very high taxonomic rank, to support vegetative identification without prior knowledge of taxonomy.

- The scientific process of revising taxa is a bottom-up process. The most urgent need for terminology and digital descriptions is present at the level of genus or family. It would be unproductive to postpone using advanced computer-supported description software until the taxonomic tree is stable and the terminology modules for the higher taxonomic levels have been agreed upon.

Despite these limitations, a hierarchy of terminology modules designed for taxonomic groups is desirable and should be supported in the information model. At the moment, however, the taxonomic hierarchy should not be a required element in the organization of the terminology. Instead, it may be used to label and organize terminology modules that are then manually selected and combined in a project. Judicious use can limit the danger that may result from changes in parts of the phylogenetic classification that are poorly understood. For example, it may be desirable to skip a poorly defined order rank and duplicate a few characters in multiple family terminologies.

Similar to taxonomy-specific standard terminology modules, terminology modules specific to methods or instrumentation could be defined and standardized. The complete terminology for a descriptive project could then be a combination of terminology modules (Fig. 93) plus local terminology extensions.

Class X:	Morphology	Anatomy	Ultrastructure	Physiology
Order 2	Morphology	Anatomy	Ultrastructure	Physiology
Family 3	Morphology	Anatomy	Ultrastructure	Physiology
Genus 6	Morphology	Anatomy	Ultrastructure	Physiology

Figure 93. Combining multiple terminologies ("character definitions") can also be useful to combine characters defined for different methods and add them to the current project as needed.

188. It is desirable to express the scope of terminology modules relative to the taxonomic hierarchy. Application may use this information to manage availability of terminology items for different taxa.

Models to support multiple distributed terminologies

Three basic approaches to connect local descriptive data with standardized terminologies can be distinguished:

- The **namespace model**, in which the standard terminology resides entirely on the internet and is only referenced in the local terminology. A local cache may be present, but no local changes or extensions are possible (Fig. 94, right side).
- The **template model**, in which a standard terminology is copied to a local terminology and can then be changed. Provided some kind of identifier remains unchanged in the copy, the identity of origin may then be used for data integration. However, without human control the local changes may substantially change the semantics of the terminology up to the point where data integration is no longer sensible (Fig. 94, left side).
- The **declarative model**, in which the terminology is defined locally, but the developer declares that the definition of a given term (character, state, etc.) follows a published standard. This may be achieved by citing a standard identifier or reference, version, plus a specific code for each term from the standard.

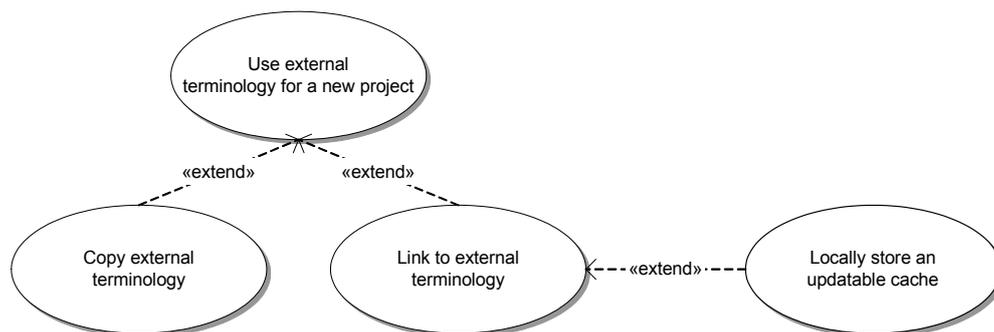


Figure 94. External terminology may be copied or linked, the latter optionally with a local cache. Compare also the general use case diagram (Fig. 91).

Namespace model: Standard terminologies could reside on multiple servers on the internet and could be used directly from there (Fig. 95). This is similar to the use of multiple XML namespaces (with a schemaLocation as a resolution method) in a single XML document. Given that online internet connections may be expensive, unreliable, or even unavailable (e. g., on a notebook in the field), a mechanism to locally cache external terminologies is desirable.

Using a namespace model, a standard terminology module would always be included in its entirety. This may be acceptable if each standard is split into small modules (e. g., separate modules for methods/instrumentation) so that the amount of unnecessary terminology that may confuse users is minimal. Alternatively, the information model could provide a local mechanism that allows defining subset views on the standard terminology.

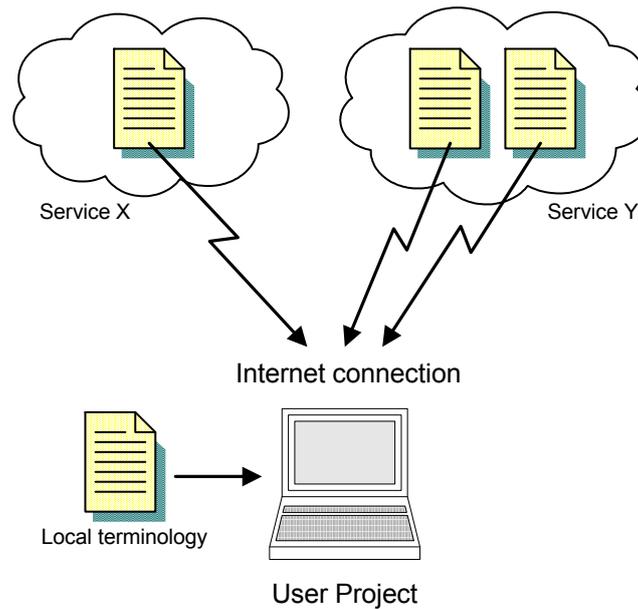


Figure 95. Network namespace model for federated terminologies. Multiple standardized terminologies are stored on the internet and used directly from there. Only terms not defined in a standard are stored in a local terminology.

The disadvantages of the namespace model are:

- The standard terminology modules would have to be available before the work on a project begins. It is difficult to combine this model with locally defined terminologies. If local terminologies overlap with only recently developed standard terminologies, the descriptions have to be ported to make use of the new standard terminology. The model itself provides no mechanism to do this gradually.
- A similar problem may arise, if a new version of a standard is published. A new version that is not fully backwards compatible would not replace an existing standard, but would be added as a new namespace. Changing the referenced standard itself is feasible only in very limited circumstances, since any substantial changes would invalidate the descriptions that use the older definitions.
- Standards could be published only digitally. This may be acceptable if all programs use a common exchange standard format. As long as multiple formats are used, and given the ongoing importance of printed publications as long as digital publications are too unstable to guarantee retrieval at a future date, this is, however, undesirable.

Template model: If one or several standard terminologies are used in a new project, they can be copied from templates that are available from a library of standard terminologies (Fig. 96). To trace the definition of a character back to the standard template it originates from, an explicit mechanism such as a Globally Unique Identifier (GUID) is required to remain unchanged in each term.

Once a template is copied into a local terminology, it can (and usually needs to be) changed. In these cases great care must be taken that the changes do not lead to situations where the human-readable definition in character or states contradicts the semantics of the original definition in the standard used as a template. The developers of terminology are ultimately responsible that the terminological concepts perceived by users using the terminology for coding and identification remain sufficiently similar to the concepts defined in the standard terminology template.

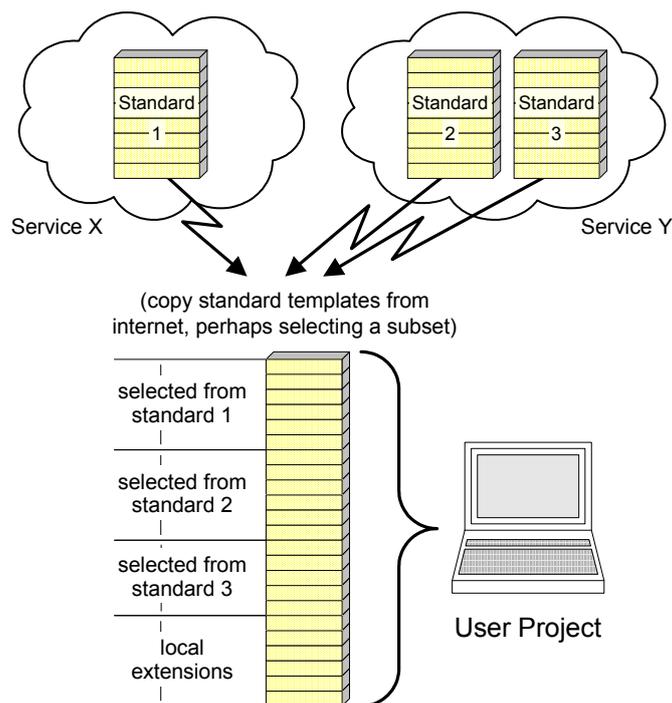


Figure 96. Template model for federated terminologies. Several terminology modules are copied from templates and can then be changed similar to local definitions.

Declarative model: In this model any terminology is primarily developed locally. Wherever possible, the developer adds an explicit declaration that the concept of a local character or state conforms to the concept of a character or state in a standard terminology (Fig. 97). The declaration should consist of an identifier or a reference for the standard, the version of the standard, and a reference to the individual term. These elements may be combined, so that a single Globally Unique Identifier (GUID) may include the information on the term, the standard it is contained in, and the exact version of the standard. The standard could be identified through a URL, or through text citing a printed publication. The advantages of this approach are:

- Unlike in the namespace model, external standard terminologies are not required to have a specific format (e. g., SDD).
- The external standard may even be a conventional, printed publication. Printed and digital standards could exist side by side.
- A smooth transition of existing data sets towards increasingly standard-conforming data is possible, since the declarations can be made individually for single terms (character, states) rather than being restricted to entire terminologies.
- The process explicitly supports the process of migrating from existing terminologies to newly developed standard terminologies, or from older to newer standard versions.

Disadvantages are:

- No automatic discovery mechanism for possible relations to standards is anticipated. The machine-readable data-integration mechanism depends entirely on human comparison of local and standard concepts.
- Developing a local terminology for a given group involves substantial work and often many revisions to correct for initial errors in the terminology.

These points can be addressed by combining the declarative model with a template model, copying a ready-to-use terminology module, but maintaining the publicly visible declarative reference. If the designers of the terminology detect that they are changing a term in a way that the local and the standard concepts differ, they may remove the declarative reference to indicate this.

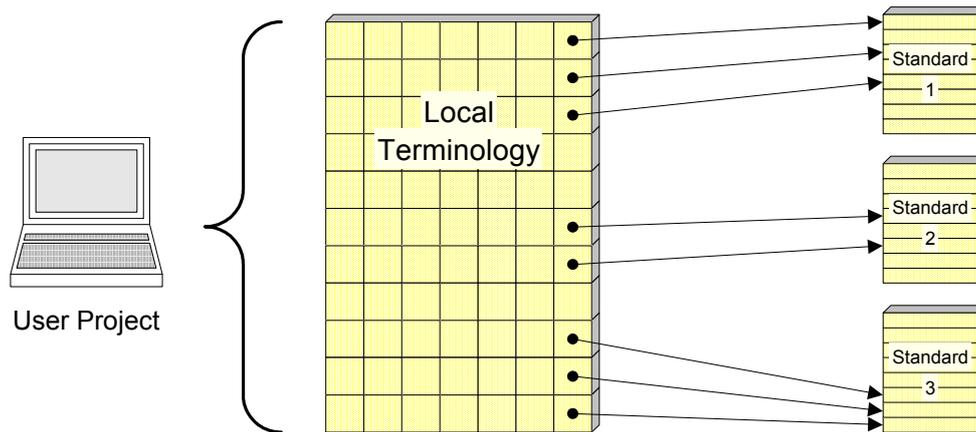


Figure 97. In the declarative model each term of the local terminology contains, among other data elements, an optional reference to a standard terminology. This reference is set by the designers of the terminology to declare that the local concept is identical with the concept in the standard. The standard may be available directly in digital format, or may be published in a printed publication. If the declarative model is combined with a template model, the reference will already be set for those parts copied from a standard template.

189. Terms in the terminology modules should be identified by GUIDs.

190. It is desirable that the relation between locally defined terms and external standard terminology modules can be expressed through GUIDs. The relation may be of several kinds: e. g., “copied from template”, believed to be “similar” or “essentially identical”.

Conclusions

A desirable solution that is both flexible and efficient seems to be a combination of the declarative and the template model. This may allow the evolutionary term-by-term mapping of existing, locally defined terminologies towards standardized and shared terminologies, while at the same time profit for new terms from the work invested into standard terminologies. New terms could be used as templates and would – in addition to including many presentational or assumptional elements – already contain the declarative information required to map to a standard, making the laborious and error-prone ontology-mapping process no longer necessary.

By referring to published and standardized core terminologies it will be possible to create federated descriptive data collections, where multiple independent sites store descriptions that can be compared or integrated. The use of Globally Unique Identifiers (GUIDs) even allows one to directly join terminologies based on ID identity, requiring no online access to the standard terminologies to which they ultimately refer (Fig. 98).

In the future it is to be hoped that a large library of reusable and tested terminology modules for a wide variety of biological groups and methods will become available. Not all such modules need to be declared a “standard”; becoming a standard could be an evolutionary process of demand and acceptance. Researchers starting to develop descriptive data sets for groups of organisms where no “standards” exist could yet use previous work as a template and revise existing definitions rather than start from scratch.

To the author's knowledge no library of terminology definitions exists so far. Even for the DELTA standard only a handful of reusable character definitions can be found on the web, since most “DELTA” data are actually the binary, encoded Intkey data usable only for identification but not as a template for further character development.

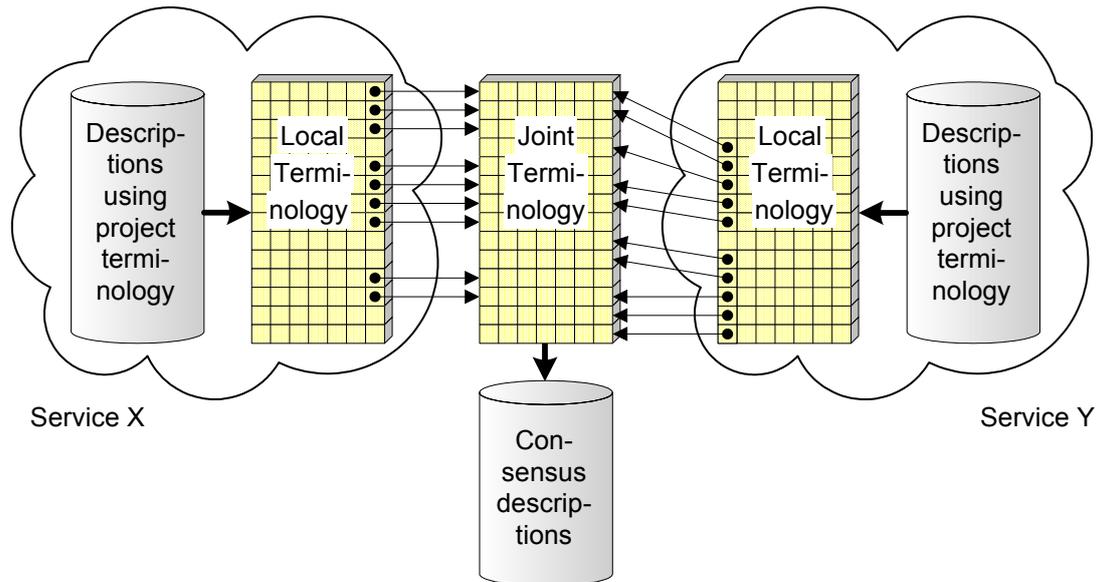


Figure 98. Consensus terminology created by a join of multiple terminologies from multiple sites on the internet. The descriptions can then be used and queried across database borders. The join shown is an outer join, so that no descriptors are dropped. Only the matching descriptors can be used together. Alternatively, the terminology could be reduced to matching descriptors (inner join).

4.14. Modifiers

Introduction

Composition (part-of) and generalization (kind-of) ontologies for object parts (structures), properties, and measurement methods have been discussed in the previous sections (p. 131 ff). When studying an actual example, however, it appears that several aspects of descriptive data are inadequately covered by these ontologies:

“Subgenus *Myrmeurynota* FOREL. Pronotum *very* broad, with a *lateral*, lamelli-form margin, *often* vaulted. Thorax *rapidly* narrowing *behind*. Epinotum *very* narrow *at its sloping face*, which *often* has a peculiar appendage. Gaster broad, short, and small, *sometimes more or less* spherical. *Probably* arboreal.” (Ant descriptions by Wheeler, example provided by R. Morris, potential modifiers emphasized.)

These terms may be called *modifiers* of frequency, certainty, degree, and location. As discussed above (see, e. g., Fig. 89, p. 179), descriptive information often may variously be placed in terminology or descriptions. The options available for handling modifiers are:

- Modifiers are embedded in the character or state terminology.
- Modifiers are freely added to the descriptions as unconstrained text annotations (“comments”).
- Modifiers are added to descriptions in a structured form, constrained by and referring to a separate mechanism in the terminology.

In classical DELTA (p. 19), only the first two options are available (Fig. 99 top). The second option is impractical for frequency or certainty modifiers, if the data are intended for identification or analytical purposes (is the stem of a specimen “*frequently hairy*”, “*rarely hairy*”, “*probably hairy*”, or “*perhaps hairy*”?). In contrast, modifiers of degree may be embedded in states (for a character: “*Stem (hairiness)*” the states could be: “*not hairy*”, “*hairy*”, “*slightly hairy*”, “*strongly hairy*”, and “*hairy at the tip*”. However, this is often unsatisfactory, causing an inflation of

character states that have complex relationships among each other. Consequently, in DELTA data sets the preferred method of expressing modifier information is the use of free-form text comments. The problems with doing so are:

- Important information is not accessible to machine reasoning (except perhaps by sophisticated natural language processing). For example, for identification processes, frequency, certainty, and misinterpretation information is relevant but difficult to obtain from unconstrained text.
- Interpreting the use of similar, perhaps synonymous modifier phrases is difficult. For example, “often” and “usually” may express the same or different frequency concepts.
- When creating multilingual data sets, allowing people from different cultures to collaborate, comments must be translated in each description rather than a single time in the terminology. Treating modifier information as free-form text comments drastically increases the number of comments, causing a heavy translation burden (60-90% of free-form comments are typically modifier-related, unpubl. analyses of DELTA data sets).

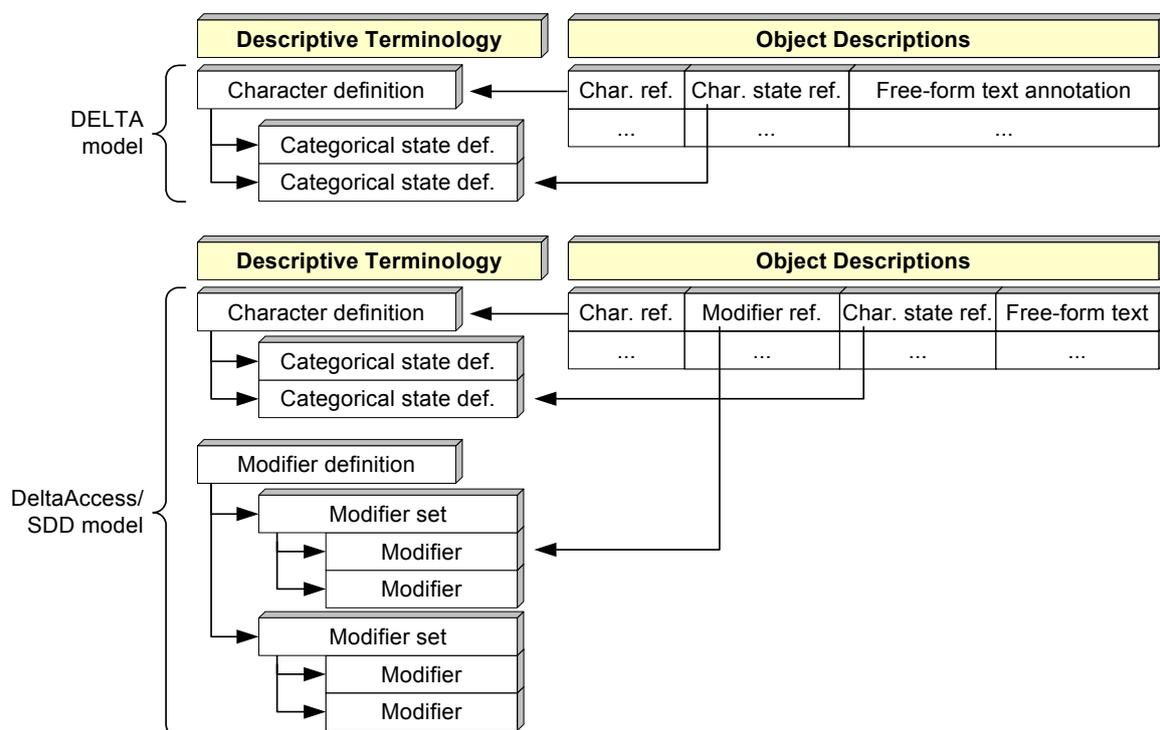


Figure 99. Simplified comparison of DELTA-like and DiversityDescriptions/SDD models in regard to modifiers and free-form text annotations.

The option to add structured modifier information (constrained by a separate modifier terminology) is not available in classical DELTA but has since been proposed in various descriptive information models (see below). Such an additional, independent dimension of terminology can strongly simplify a scoring scheme (Fig. 100). Structured modifiers (as provided in Diversity-Descriptions and SDD, Fig. 99 bottom) have many advantages:

- Modifiers provide flexibility in the level of detail that is recorded. They decouple the level of detail imposed by the character definition from levels of detail provided in data or chosen for various analysis purposes.
 - By initially ignoring modifier information, a coarse view of descriptive data is often desirable during identification to concentrate on major issues.
 - In data analysis one may choose between a coarse analytical treatment (ignoring frequency, uncertainty, and minor modifications of degree or location), and a detailed analysis where

values with different modifiers are considered to be different. In conventional DELTA this decision must be made when the character definition is elaborated.

- Machine-readable frequency and certainty information can be evaluated by identification algorithms.
- When aggregating information (e. g., from specimens to species, or species to genus descriptions), modifiers can generally be handled better than free-form text comments. Without NLP methods, comments can only be concatenated, whereas modifier identity or differences can be analyzed and appropriately aggregated.
- By reducing the number of free-form text comments, they simplify the translations of a data set. Similarly to characters and states, modifiers must be translated only once for all descriptions, and not per-description like comments in the description.
- If the designer of the character and modifier terminology has options to impose (constraining the validity of data) or recommend (accepting or ignoring recommendations has no impact on the validity of data) associations between characters and modifiers, additional benefits arise. For example, it is possible to provide concise modifier pick lists in the data entry user interface, containing only modifiers applicable to the current character.

<p>[No separate modifiers available] (e. g., as in DELTA)</p> <p>Spore appendages</p> <p>Appendage presence/frequency</p> <ul style="list-style-type: none"> <input type="checkbox"/> 1. without appendages <input type="checkbox"/> 2. rarely with appendages <input checked="" type="checkbox"/> 3. usually with appendages <input type="checkbox"/> 4. always with appendages <p>Diameter [] (μm)</p> <p>Diameter at base [1.4] (μm)</p> <p>Diameter at middle [] (μm)</p> <p>Appendage tip</p> <ul style="list-style-type: none"> <input type="checkbox"/> 1. blunt or rounded <input type="checkbox"/> 2. pointed <input checked="" type="checkbox"/> 3. strongly pointed 	<p>[Partly delegated to modifiers] (e. g., as in DiversityDescriptions)</p> <p>Spore appendages</p> <p>Appendage presence</p> <ul style="list-style-type: none"> <input checked="" type="checkbox"/> [usually] 1. present <input type="checkbox"/> [] 2. absent <p>Diameter [at base] [1.4] (μm)</p> <p>Appendage tip</p> <ul style="list-style-type: none"> <input type="checkbox"/> [] 1. blunt or rounded <input checked="" type="checkbox"/> [strongly] 2. pointed
--	---

Figure 100. Excerpt from a scoring scheme for fungal spores. The left side illustrates several cases that can be simplified with the introduction of modifiers (right side).

Definition

The term “modifier” is used in natural language and has a specific meaning in grammar (Table 45). Both senses are in agreement with the usage proposed here in descriptive data. “Qualifier” is approximately synonymous with modifier and could be used instead of “modifier”. The different use of qualifier in UML is clearly very specific and would cause no confusion (Table 45). However, no advantage of “qualifier” over “modifier” can be seen either. According to CED (1992), “modifier” is the preferred term for the grammatical concept, and it has been used in descriptive information models for some time now (Hagedorn 1997).

As shown in the definitions, a modifier may either be a noun in a composite noun or an adjectival or adverbial word or phrase. In natural language, many character states are expressed as adjectives of the objects being described (exceptions are “kind-of” states, such as: *fruit = capsule, berry, nutlet, etc.*). As a consequence, in English many modifiers take the form of adverbial modifications of these adjectives. Many types of adverbs commonly distinguished in English grammar correspond to potential modifier categories:

- **Adverbs of place and time:** “stem hairy *at the tip*”, “leaves glossy *towards the tip*”, “stem hairy *in upper half*”, “stem diameter *at the base* 8-15 mm”, “flower violet (*when mature*)”, “stems glaucous (*in spring*)”;
- **Adverbs of frequency:** “usually”, “rarely”;

- **Adverbs of probability:** “*probably*”, “*perhaps*”;
- **Adverbs of manner:** “twigs breaking *easily*”, “leaves rotting *slowly*”;
- **Adverbs of degree:** “*slightly* rough leaves”, “*strongly* pointed tip” (other examples: “*almost*”, “*completely*”, “*extremely*”, “*hardly*”, “*nearly*”, or “*particularly*”).

Table 45. Examples of dictionary and UML definitions for “modifier” and “qualifier”.

Term	Collins English Dictionary (CED 1992)	Merriam-Webster’s Collegiate / New Oxford Dictionary (EB 2001)	UML 1.5 definitions (OMG 2003)
Modifier (noun)	1. Also called: qualifier. <i>Grammar:</i> a word or phrase that qualifies the sense of another word; for example, the noun <i>alarm</i> is a modifier of clock in <i>alarm clock</i> and the phrase <i>every day</i> is an adverbial modifier of walks in <i>he walks every day</i> . 2. a person or thing that modifies.	1. a person or thing that makes partial or minor changes to something. – <i>Grammar:</i> a word, especially an adjective or noun used attributively, that restricts or adds to the sense of a head noun (e. g., <i>good</i> and <i>family</i> in a <i>good family house</i>). – <i>Genetics:</i> a gene which modifies the phenotypic expression of a gene at another locus.	(not used in UML)
Qualifier (noun)	1. a person or thing that qualifies, esp. a contestant in a competition who wins a preliminary heat or contest and so earns the right to take part in the next round. 2. a preliminary heat or contest. 3. <i>Grammar:</i> another word for modifier (sense 1).	1. a person or team that qualifies for a competition or its final rounds. – A match or contest to decide which individuals or teams qualify for a competition or its final rounds. 2. <i>Grammar:</i> a word or phrase, especially an adjective, used to attribute a quality to another word, especially a noun. – (In systemic grammar) a word or phrase added after a noun to qualify its meaning.	An association attribute or tuple of attributes whose values partition the set of objects related to an object across an association.

Modifiers in natural language may occur before the character wording (e. g., “upper stem leaf hairy”), between character and state wording (e. g., “leaf strongly hairy”, “leaf margin hairy”) or after the state wording (e. g., “leaf hairy on veins”). In “basal leaves in spring”, “basal” and “in spring” may be called a *pre*-modifier and a *post*-modifier, respectively. In the case of coded descriptions this becomes merely language-specific metadata for natural language output.

An exact definition of the concept of modifiers in the context of descriptive information models is not easy. Both an intuitive and a more formal attempt are given:

Informal: *A modifier is a unit of information that adds detail (or constraints) to the statement to which it is applied. When the modifier information is ignored, the original statement must retain a substantial, albeit more general meaning. A modifier may be applied to statements already modified. Modifiers themselves are constrained by a terminology.*

More formal: *A modifier is a unary function applied to a proposition, resulting in a modified proposition. Modifiers may be applied to modified and unmodified propositions.*

Especially modifiers of probability/certainty, but perhaps modifiers in general may be considered as an application of modal logic in the broad sense (i. e., including temporal logic, conditional logic, etc.; see Stanford Encyclopedia of Philosophy 2003). Modifiers may be thought of as bringing simple predicate-subject descriptions into a modal form.

This topic needs further study.

Current usage of modifier-related concepts

DELTA and New DELTA: Although the need for modifications of character state \times entity instances was recognized (Dallwitz 1980), the provision of free-text comments (Fig. 99 top) was considered sufficient at the time. Proposals for a revised “New DELTA” (p. 20) did include several additional mechanisms, called “*coded comments*”. Some of these are related to modifiers:

- Two forms to express probability or frequency values (“<@probability x>”, “<@x%>”). These can be combined with a free-form text like “frequently” or “usually”, but do not replace them. An exception is “rarely” which is available as a system-defined coded comment (“<@rarely>”). No other frequency categories are defined in “New DELTA”, nor can they be defined by the content authors. It is not possible to distinguish between probability based on frequency, and probability based of uncertainty estimates.
- The coded comment “<@about>” for quantitative numeric data as a system-defined approximation modifier.
- The coded comment “<@?>” to mark guessed values, essentially a system-defined “probably”. No other certainty modifiers (such as “perhaps”) are available.
- The coded comment “<@reliability x>” to modify the reliability defined for the character in the terminology in a specific description.

Further coded comment mechanisms in “New DELTA” support information about values that are inherited or calculated along the taxonomic hierarchy (“<@up>”, “<@down>”), supply hidden notes not visible in natural language output (“<@note: text>”, replacing the DELTA ‘inner comment’ mechanism), and specify alternative character values for particular applications (“<@use n: s>”). These are all unrelated to the concept of modifiers.

The coded comment system of New DELTA is not extensible, only modifiers defined in the standard may be used.

NEXUS (p. 18): For categorical data, NEXUS supports counts or frequencies if the subcommands “StateFormat=Frequency” or “StateFormat=Count” are given). All entries in the “Data, Matrix” block then must be enclosed in parentheses. For example, in “taxon_1 (0:0.25 1:0.75) (0:1.0 1:0.0 2:0.0)” the first character is polymorphic with states ‘0’ and ‘1’, the second monomorphic with state ‘0’ and alternative states ‘1’ and ‘2’. No other forms of modifiers (or free-form text comments) are available in NEXUS.

DiversityDescriptions (Fig. 99 top): At about the same time when the New DELTA was initially proposed, a generalized modifier concept was developed in DiversityDescriptions. Already the first version (Hagedorn 1997) included a flexible concept of user-definable, reusable modifiers. Modifiers were defined in a single list for an entire project (Table 46), with a separate list defining the applicability of modifiers to characters. In descriptions, only applicable modifiers are selectable for a given character. For categorical characters, the list of states and the list of applicable modifiers can be freely combined in descriptions.

Modifiers were categorized into usage categories (which could be used when defining the applicability of modifiers to characters) and a number of properties could be defined for each modifier (influence on character value reliability, output in natural language before or after the value, and with or without a blank). A template definition of modifiers was provided as a convenience to content authors, but modifiers could be freely added or deleted.

The “usage categories” of DiversityDescriptions are unordered sets; it is not generally possible to define a ranking within such a set. In newer versions of DiversityDescriptions, a ranking can be indirectly defined for frequency modifiers through a quantitative frequency interval (attributes “LowerFreq”, “UpperFreq”, not shown in Table 46). As a result of the discussions in the SDD group, later versions of DiversityDescriptions further added the concept of misinterpretation modifiers (see below, p. 209) pioneered by CBIT Lucid/Lucid3.

Both the list of modifiers (Table 46) and the usage categories are fully extensible in Diversity-Descriptions, separately for each project.

Table 46. Excerpt from template modifier definitions in DeltaAccess/DiversityDescriptions 1.0 (Hagedorn 1997; later versions added additional concepts).

Usage class*	Modifier	Reliability*	Postfix*	Use*-Blank	Usage class	Modifier	Reliability	Postfix	Use-Blank
Frequency	almost never	1	no	yes	Location	at the base	5	no	yes
	rarely	1	no	yes		at the apex	5	no	yes
	sometimes	2	no	yes		near the base	5	no	yes
	often	4	no	yes		near the apex	5	no	yes
	mostly	4	no	yes		toward the base	5	no	yes
	frequently	4	no	yes		toward the apex	5	no	yes
	usually	4	no	yes		in lower part	5	no	yes
almost always	4	no	yes	in upper part		5	no	yes	
...				in lower half		5	no	yes	
General	just	5	no	yes		in upper half	5	no	yes
	ca.	2	no	yes		anterior ones	5	no	yes
	much	5	no	yes		posterior ones	5	no	yes
	nearly	3	no	yes	...				
	almost	4	no	yes	Morphology	finely	5	no	yes
	about	4	no	yes		distinctly	5	no	yes
	normally	4	no	yes		narrowly	5	no	yes
	?	1	yes	no		broadly	5	no	yes
	probably	1	no	yes		broad	5	no	yes
	slightly	5	no	yes		bluntly	5	no	yes
	somewhat	1	no	yes		faintly	5	no	yes
	scarcely	3	no	yes		sub	5	no	yes
very	10	no	yes	sub-		5	no	no	
...				minutely		5	no	yes	
Time	early ones	5	no	yes		...			
	soon	5	no	yes		Color	dark	4	no
	later	5	no	yes	light		4	no	yes
	late ones	5	no	yes	dull		4	no	yes
	earlier	5	no	yes	pale		4	no	yes
...				...					

* *Usage class* = broad categorization of modifiers. *Reliability* = expression allowing a modifier to influence the reliability score of the base statement or character. *Postfix* = in natural language, the modifier is rendered after the modified statement (else before). *UseBlank* = the modifier is connected with the statement using a blank. Compare p. 347.

CBIT Lucid (p. 21, Fig. 101 top): The absence of structured mechanisms to express frequency and misinterpretation information was a major reason for the developers of Lucid to develop a proprietary exchange format instead of using DELTA (K. Thiele, pers. comm.). Lucid contains a small set of value modifications, namely “rare”, “misapplied”, and “uncertain”. This list is not extensible because it is part of the character scoring mechanism inside the data matrix (i. e., modifications are not added to a score, but implied in the alternative score options: “absent, present, unknown, rare, commonly misinterpreted, and rarely misinterpreted”). Note that no “official” documentation of the LIF format was found; the information given here is inferred from data sets analyzed (Leary & Hagedorn 2004) and confirmed by K. Thiele (pers. comm.), one of the main designers of Lucid. In his comparison, Dallwitz (2006d) records Lucid as having “value is unknown”, but no “value probability” (i. e. “uncertain”) modifier. Lucid indeed calls the modifier in question “unknown”. If all states of a character are scored such, the result will indeed be equivalent to a coding status value “unknown” for the entire character (no data have or could be observed, see p. 74). However, if some states are scored normally and others as “unknown”, the result will be best interpreted as these states being “uncertain”. The developers of Lucid support this interpretation of “unknown” when they state that “Lucid can encode uncertainty for a state, while in DELTA uncertainty can only be encoded for a character” (http://www.lucidcentral.org/lucid3/lucid_translator.htm). Lucid supports no unconstrained text, so that all expressiveness is limited to the three modifiers supported, and to modifiers embedded in the character or state defi-

nitions. This design decision makes Lucid simple to use for identification purposes, but limiting when aiming for comprehensive natural language descriptions.

XPER and **XPER²**: These programs (tested March 2007, latest version XPER²: 1.70) do not support modifier concepts. XPER² supports unconstrained text on character data (“descriptors”), but not on individual state scores. This makes it difficult to express state-specific modifier information the way the original DELTA does.

Nemisys/Genisys: The mechanism of modifiers is related to the “*name-extension*” mechanism (Fig. 101 bottom) proposed in Diederich (1997) and Diederich & al. (1997). A name-extension modifies the semantics of either the part (i. e. “structure”) or the property name of a character. It is proposed as a mechanism to curb the explosions of the number of characters that may occur otherwise; e. g., Fig. 9 in Diederich (1997) lists eleven different ways of measuring body diameter of nematodes.

Although both constituents of the decomposed character potentially need modification in a single character, according to Diederich (1997) only a single data element exists for the “name-extension”. Thus either structure or property, but not both may be modified. The examples given in the tables in Diederich & al. (1997) seem to support this interpretation, making “name-extensions” and modifiers rather similar (compare examples given in Table 47). Some guidelines are given in Diederich & al. (1997) indicating that modifiers referencing structural terms (at mid-body, at anus) are to be constrained to existing, defined structures, and extensive guidelines are given on the use of spatial modifiers (anterior/posterior etc.). No similar guidelines are given for non-structural modifiers.

Table 47. Examples of use of name-extensions from Diederich & al. (1997).

Structure	Structure extension	Substructure	Property	Property extension	State
Excretory pore	–	–	position relative to	{median bulb, nerve ring}	anterior
Stylet	–	–	length	{along the axis}	(value)
Body	–	–	diameter	{at mid-body, at stylet, at Vulva, at Anus}	(value)
Body	{anterior part, posterior part}	Lateral fields	orientation	–	symmetrical

As defined above, “name-extensions” are meant to modify only structure or property terms, not quantitative values or categorical states. Consequently, no equivalents for frequency of probability modification (“rarely”, “perhaps”, etc.) are discussed in the Nemisys/Genisys model.

It remains unclear whether a separately defined, reusable modifier terminology is intended or not. Diederich (1997) states that “Instances of basic properties maintain a list of name extensions”, and “when a character is created, an instance of a basic property is created”. However, Diederich (1997) does not distinguish between character variable (terminology) and character data (description); the term *character* may refer either to “(biological structure, property, state/value)” or to “(structure name, property) tuples, ignoring the states”. Thus, the statement may refer either to constraining name-extensions in descriptions by a list of defined extensions in the terminology, or to a model where within each description, a list of multiple name extensions is allowed.

“Name extensions” are further used to represent the additional information necessary for Diederich and Fortuner's “relational properties” (e. g., “presence-at ...”, requiring a second structure, see Table 9, p. 63 and compare “Relational characters revisited”, p. 122). One may view “relational properties” as properties that are constrained to require a structural modifier. It is unclear, whether in this case the “name extension” is limited to defined structure terms, or whether it is in fact considered free-form text, allowing a combination of modifier and related-structure information. In the first case, it may be difficult to express a relation to a part of a structure that would otherwise require a modification. Furthermore, the main structures in the Nemisys/Genisys model

are divided into structure and substructure, whereas name extension is a single data element, again pointing to the interpretation that the name extension is originally designed as a free-form text element.

Prometheus description model (p. 21): This character decomposition model elaborates and refines the Nemisys/Genisys proposals. The name “modifier” is explicitly used for the Nemisys/Genisys “name extensions” and the missing concept of frequency modifiers (certainty seems to be not mentioned, but would be a simple extension of the model) is added. Similar to the Nemisys/Genisys name extensions, Prometheus modifiers have a dual nature:

- A basic type of modifier modifies an otherwise complete statement: How frequently was something observed, when was it observed in time, or where exactly was something observed within a structure. Pullan & al. (2005) further distinguish:
 - Simple categorical modifiers (they only name frequency as an example); these may be stored directly as an attribute of a description element.
 - Modifiers which modify a statement by reference to a spatial or temporal “landmark” (examples: modifier = “at” + spatial landmark = “breast height” or modifier = “during” + temporal landmark = “summer”). In this structure, the modifier part will only take very few values (e. g., “at, below, above” or “after, before, during, or while”). It seems implied that this information cannot be stored as part of description elements but requires a separate data structure.
- An entirely distinct type, the *relative modifier* combines two existing description elements (rather than referring to one), and either expresses knowledge about relative order or rank (supporting the fixed operators “>, ≥, <, ≤, =, ≠, ratio-of”), which may be combined with a value. Using this method information such as “leaf width > length”, or “petal length > 2 × sepal length” may be expressed.

The second concept is related to the use of name extensions together with the “relative properties” in Nemisys/Genisys. The solution is, however, different and more general. Whereas in Nemisys/Genisys the name-extension would be used to complete an otherwise incomplete single statement in a single record, Prometheus seems to introduce a concept of non-scored or non-valued “description elements” (i. e., a part-property-value tuple without a categorical or quantitative value). Two such description elements (valued or unvalued) are then combined with a relative modifier, to form a new statement. Table 48 is an attempt to illustrate the proposed solution.

It is debatable whether this is appropriately called a “modifier”. The Prometheus “relative modifiers” are structurally completely different from the basic modifiers, and it remains unclear which information they modify – they rather create an entirely different form of statement. It does not meet the definition of modifiers proposed here, and neither does any of the dictionary definitions of “modifiers” cited above support calling comparison operators “modifiers”. Pullan & al. (2005) themselves explicitly state the requirement that “when querying descriptions, it is possible to ignore these modifiers without detrimental effects to the results.” This requirement is clearly violated by “relative modifiers”.

Further problems with the concept of *relative modifiers* are:

- It is unclear how a statement combining a ratio and a relative operator “length-width-ratio > 2.0” (a frequent form in mycology) is handled. This problem seems to indicate that handling comparison operators and functions in a single data element is inappropriate.
- Pullan & al. (2005) in their Figure 2 explicitly mention a requirement for a “Defined Unit” (i. e., measurement unit, like cm) for relative modifiers. This requirement needs further study: ratio- or comparison statements should normally always be dimensionless and no example for a relative modifier requiring a measurement unit was given.

Returning to the basic modifiers, it is debatable whether the distinction between simple and landmark-modifiers is justified. The separation of spatial modifiers into two parts (modifier plus landmark) is clearly often advantageous since defined structural terms can be reused (“stem hairy” + “at” + “tip”, “base”, “middle”, “inflorescence”). However, the quoted example of “at” +

“breast height” already indicates spatial modifier landmarks will not always be a natural part of the part-of vocabulary used in character decomposition (other example: “width”+“at”+“widest point”). In the case of temporal modifiers the landmark vocabulary will even be less reusable (e. g., “nightfall”, “fruiting”, “first flowering”, or “spring after cutback”). However, if a separation into two data elements is already introduced for spatial modifiers, reusing it for temporal modifiers is logical. It remains unclear in the model, in which part of the terminology landmark terms (special spatial as well as temporal ones) will be defined, and whether separate data structures are envisioned for this.

None of frequency, spatial or temporal modifiers (e. g., frequency, distance along stem, or time of day) may carry a quantitative expression in Prometheus; quantitative values are reserved for relative modifiers.

Table 48. Interpretation of the structure of relative modifiers, based on information in Pullan & al. (2005). Note that description elements may or may not have values. Note the interpretation of “value” in the modifier entity differs strongly whether expressing is length/width = 2.1, or length ≥ 2 × width.

Description elements:				
ID	Structure	Property	Value	Meas. Unit
1	Leaf	Length	5.1	cm
2	Leaf	Width		
3	Sepal	Length		
4	Petal	Length	7	mm

Relative modifier-statements:					
Example	Source (ref. to ID)	Destination (ref. to ID)	Operator	Value	Meas. Unit
1	1	2	Ratio	2.1	
2	1	2	≥		
3	1	2	≥	2	
4	3	4	>	2	

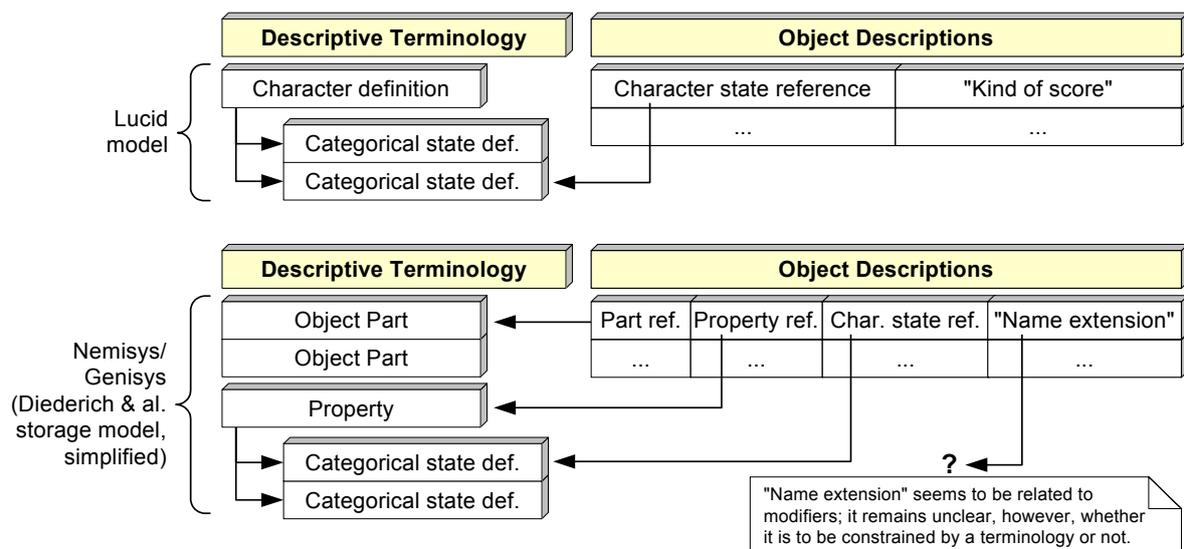


Figure 101. Simplified comparison of CBIT Lucid3 and Nemisys/Genisys models in regard to modifiers and free-form text annotations. Compare Fig. 89 (p. 179) for additional information about the Nemisys/Genisys model.

SDD (p. 20) defines modifiers as part of the descriptive terminology. These modifiers are grouped into sets of modifier concepts. In descriptions, character data may be modified by referring to these defined modifier concepts. SDD also supports the unconstrained free-form text for individual character state scores, replicating the functionality present in DELTA.

Modifiers were intensively discussed in the SDD group and repeatedly changed. In SDD version 1.1, the main concept ontology, which is also used to create character hierarchies, is also used to define modifiers. Thus a hierarchy of modifier concepts may be created, e. g., with several different sets for frequency modifiers. Modifier concepts may contain a special specification, whether the sequence of modifiers defined within a concept is considered to be significant (“ordered=true”, as in ‘weakly’–‘moderately’–‘strongly’) or not. With “ordered=false”, the sequence of modifiers is intended for display purposes only and carries no additional semantics.

Both the concepts (modifier sets) and individual modifiers are fully user-definable and not constrained by SDD. However, to support application interoperability and identification processes, this is complemented by two modifier specification attributes: A modifier *Class*, with an enumerated list of values (see Table 49), and a quantitative range, that, depending on the modifier class, is interpreted as a quantitative frequency, certainty, etc. estimate. Modifier classes have been defined where a quantitative range was considered desirable, or where modifiers are expected to influence data analysis and identification processes (especially, frequency, probability, and misinterpretation). The modifier class “Other” is left undefined, to support any future uses of modifiers.

Table 49. Enumerated modifier classes in SDD 1.1 (based on annotations in the SDD schema).

Class	Description	Interpretation of Proportion
Frequency	Frequency modifier. Examples: “rarely, occasionally, usually”.	Values specify a frequency range
Certainty	Certainty modifier. Examples: “perhaps, probably”.	Values specify a certainty range
Seasonal	Seasonal modifier. Example: “in spring”.	Values specify a season of the year. The value 0 is to be interpreted as day 1, the value 1 as day 365 of the year
Diurnal	Diurnal modifier, referring to parts of the day (24 h clock, i. e., including ‘nocturnal’ events). Examples: “in the morning, at night”.	Values specify a time of the day. The values 0 and 1 are both to be interpreted as midnight. Example: A modifier “at night” may be specified as ‘0.8-0.2’.
TreatAs-Misinterpretation	States to which modifiers of this class are added are known to be intentionally wrongly scored to anticipate known misunderstandings of the character under study. Example: if bracts look like petals, petals may be scored as ‘white (by misinterpretation)’.	None
Other	All other modifiers for which specifications are not yet defined. Examples are developmental, absolute, and relative spatial modifiers, or modifiers of degree.	None

A separate mechanism allows recommending modifiers for certain characters. This “enabling” of modifiers for characters has been deliberately *not* formulated as an identity constraint; it is perfectly legal to have a modifier in a description that is not currently recommended for this character (it must only be present in the terminology of the data set). A major reason for this design is that large institutional data sets may contain descriptions from various sources (e. g., NLP-processed natural language descriptions, imported DELTA, or CBIT “LIF” data). These older descriptions may use a richer set of modifiers than what has been agreed in a project to use in the future. SDD conforming applications may choose whether to use the information about recommended modifier/character association for data entry or not. They are encouraged to do so.

Modifier sets and sequences

For several reasons it is desirable to organize modifier definition nested inside higher concepts, called here “modifier sets”:

- For some modifiers (frequency, certainty) it is desirable to define a ranking (linear ordering). The order of modifiers within a set defines an order from lowest to highest frequency/certainty. New data gathered under a single or shared terminology will normally use modifiers from a single frequency and certainty set. However, data obtained from a variety of sources will profit from an ability to define multiple frequency or certainty sets, each with a separate ranking. Under the provision that estimates for quantitative probability ranges may be given for individual modifiers (as in SDD, see above), data involving modifiers from multiple sets may still remain comparable.
- It is desirable to be able to define the applicability of modifiers to characters (or other forms of variables in character decomposition models). This enables the designer of the terminology to restrict the modifier vocabulary and to achieve better consistency in data entry (e. g., using a modifier pick list, see p. 191) and analysis. Defining modifier applicability individually for each character and modifier would be very cumbersome, and changes in the terminology would be difficult to manage. Organizing modifiers into sets that can be enabled as a whole greatly simplifies this task.

Modifier sets need a label so that they are selectable when defining the applicability of a modifier set to a character. They further need a Boolean attribute defining whether the order in which the modifiers are defined is semantic (i. e., the modifiers are ranked) or irrelevant.

Although it is strictly required only for ordered (ranked) modifier sets, it seems reasonable to require that all modifier sets may only contain modifiers from a single modifier type. For example, it is not possible, to mix spatial and temporal modifiers in a single set. This can easily be achieved by making the modifier type an attribute of the modifier set, rather than individual modifiers.

Modifier combinations

If the information model enables multiple modifiers on a single character value in descriptions, a clarification about the semantics of the relationship between modifiers and character value is necessary. In principle, it may be desirable to express the following combinations of two modifiers and a value:

1. Modifier hierarchies:
 - “modifier1 on (modifier2 on value)”
 - “(modifier1 on modifier2) on value”
2. Boolean operators:
 - “(modifier1 or modifier2) on value”
 - “(modifier1 and modifier2) on value”
3. Intermediate modifiers:
 - “(modifier1 to modifier2) on value”

1. Modifier hierarchies: The first two examples interpret a sequence of modifiers as a hierarchy, distinguishing, e. g., between “usually (reddish green)” and “(usually reddish) green”. In the first case, a modified value is further modified, in the second case a modifier is modified before applying it to a value. The desire to distinguish between these cases creates several problems: a) in many languages the difference is difficult or impossible to express in natural language (i. e., without using some form of mathematical notation), b) as a result of this, when coding from existing legacy descriptions, it will often be time-consuming to infer the implied hierarchy, and perhaps a third option “hierarchy undecided” may be desirable, and c) the hierarchy (which may involve more than two modifiers) severely limits the options for the design of the user interface.

For many purposes, the distinction between the two cases is negligible. The first (nested) case is more general than the second (modified modifier) and, to keep the information model simple, is proposed as the preferred interpretation of modifier sequences for the purpose of machine-reasoning. Thus, where a sequence of multiple modifiers is applied to a character value, the modifiers are to be interpreted as nested modifier statements, with the last modifier being the innermost, e. g., “probably”+“frequently”+“at the base”+“stem = hairy” should be interpreted as “probably (frequently (at the base (stem = hairy)))”.

In the case of combinations involving frequency or certainty modifiers, the second case (modified modifier) can often be reformulated by making the implicit alternative explicit. For example, “(usually reddish) green” could also be expressed as “usually (reddish green), rarely (plain green)”. Where this is not possible and it is considered essential to express a “modified modifier”, it is possible to define a combined modifier, or even to simply use free-form text annotations.

2. Boolean operators: The latter two cases (“[modifier1 and/or modifier2] on value”) are short cut notations for “(modifier1 on value) and/or (modifier2 on value)”. For example, “stem hairy at the base and at the tip” may also be stated as “stem hairy at the base and stem hairy at the tip”. Since the problem of ‘and’/‘or’ Boolean operators already occurs between different states (compare p. 95), it seems desirable to simplify the information model by reusing existing mechanisms. Although Boolean operators between modifiers would not be supported in the information model, a statement like “stem hairy at the base and at the tip” can easily be generated during natural language report generation if different modifiers are detected on the same value.

3. Intermediate modifiers: Ordered or ranked modifiers may categorize an underlying continuously varying variable (similar to character states, compare p. 53). Users may desire to express that a modifier is intermediate between or spanning across modifiers by combining these. An example would be “rarely to occasionally hairy”. In the SDD model a decision was made not to support such usage in the interest of keeping the information model simple. Instead, additional intermediate modifiers may be defined for which the wording would be “rarely to occasionally”. However, it is possible to develop a natural language reporting rule that if multiple modifiers are present on a single character in a description, consecutive modifiers from the same ordered/ranked modifier set are rendered as “strongly to weakly”, rather than “strongly or weakly”.

The advice against interpreting a sequence of modifiers as “modified modifiers”, implied Boolean operators, or intermediate modifiers may not always be understood by users coding data. Also, in some languages, “and/or” between modifiers may be implied and “stem hairy (at the base, at the tip)” may sound more natural than it does in English. It would therefore be desirable to have some kind of validation for spotting cases where a modifier combination may be used with one of the semantics considered undesirable here.

Since modifiers are defined by users of the information model, this is not trivial. One option is to require that a combination of modifiers on a character in a description is limited to at most one modifier from each modifier set. The assumption would be that modifier sets are defined in a way that combinable modifiers are in different sets. For example, to express “basal, between the veins, towards the margin hairy” the three different spatial modifiers, each of which is based on a different kind of orientation or direction, would have to be defined in different sets. Enforcing this requirement would make combinations like “strongly or weakly” mentioned above not possible.

More experience with modifiers and modifier sets is required to decide whether such a validation should be enforced by the information model, or only used as an optional tool to find potential errors.

Modifiers as an alternative to character proliferation

Modifiers may be a method to avoid the creation of a large number of characters differing only in minor aspects. An interesting test case is observations that differ in respect to where a property is observed in space and time on a single object-part such as a leaf. In many cases, the various orientation/location mechanisms behave almost like independent dimensions, i. e., all or most combinations may occur. For example, when describing fungal infection symptoms in plants, leaf spot properties like density or color may be specific to each of the following:

1. Upper side or lower side of the leaf, or: “both sides are equal and the side is irrelevant”,
 2. Tip, base, margin, center, or entire leaf (many further locations such as “on veins” or “between veins” occur, but are ignored in this example),
 3. When young, when old, or throughout leaf development,
- leading to a total of $3 \times 5 \times 3 = 45$ different spatio-temporal locations where leaf spots may be recorded.

Table 50. Examples illustrating the diversity of spatio-temporal leaf spot locations. The letters correspond with those given in Fig. 102.

Example description	
A	leaf spots yellow
B	leaf spots yellow on the upper side, cream colored on the lower side of leaf
C	central leaf spots yellow, marginal spots brown, almost black at the leaf base
D	leaf spots initially yellow, on the upper side becoming reddish
E	leaf spots present only at the tip, yellow on the upper side, olive on the lower side
F	brown spots appear on old leaves (in autumn) at the margin of the leaves
G	brown spots appear on old leaves (in autumn) in the center of the lower side of the leaves

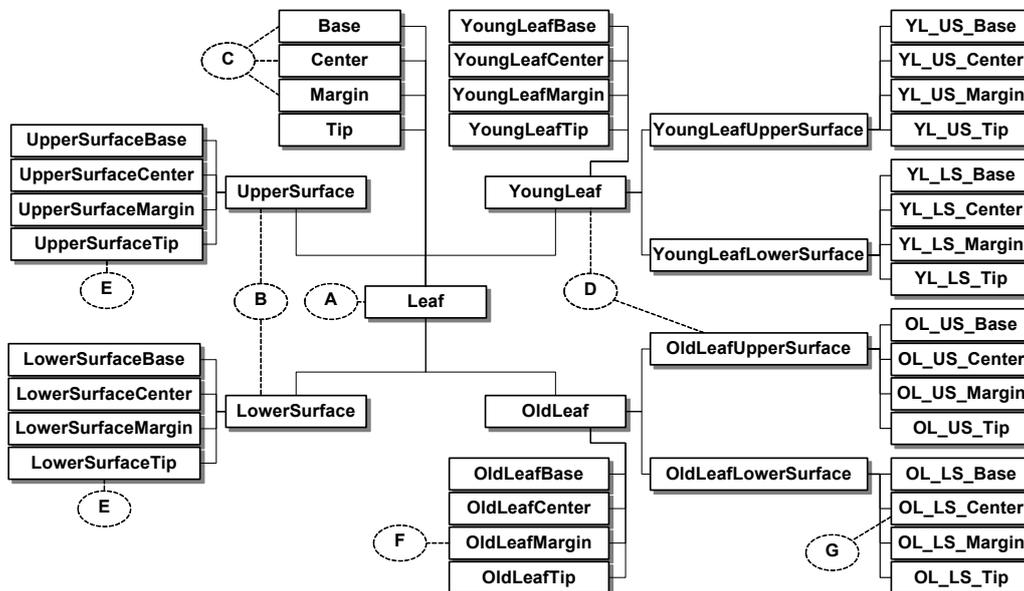


Figure 102. Three dimensions of spatial and temporal change resulting in $3 \times 5 \times 3 = 45$ different specialized classes. The class hierarchy may be composition or generalization hierarchy (not shown here). Letters in oval tags with dashed lines indicate where data from the leaf spot examples given in Table 50 would be recorded.

Even if only a subset of all combinations is used (Table 50), it is no surprise that expressing the combinations of object part, location, orientation, and temporal change in the character defi-

tion leads to very complicated models (Fig. 102). Furthermore, one may want to consider that some categories are artificial classifications of a continuum. As a result even more complex intermediate situations are possible (e. g., “late summer, 2 mm distant from the margin”).

The number of combinations actually occurring is a function of the number of taxa described. Single (temporal or spatial) object specifications (other than entire leaf, entire growth period) in this example occur with moderate frequency (upper and lower side being most frequent) and combinations of several spatio-temporal locations are used only in very few organisms. Introducing a complex model to handle rare cases, at the cost of drastically complicating general use seems to be not advisable. Notably, the example can be presented with considerably less complication in a table that provides one dimension for each of the three spatio-temporal orientations used in the example (Table 51). Here the model is easily extensible to additional combinations occurring as the number of taxa studied increases.

Table 51. Three dimensions with $3 \times 3 \times 5 = 45$ combinations expressed in a table (compare Fig. 102). These dimensions may either be part of character definitions or could be expressed through modifiers. The categories “either” and “any” may be further split into “all” and “unknown” (side, time, location), increasing data recording complexity to $4 \times 4 \times 6 = 96$ combinations.

Time	Side	Location				
		Any	Base	Margin	Center	Tip
Either	<i>either</i>	A	C	C	C	°
	<i>upper</i>	B	°	°	°	E
	<i>lower</i>	B	°	°	°	E
Young	<i>either</i>	D	°	°	°	°
	<i>upper</i>	°	°	°	°	°
	<i>lower</i>	°	°	°	°	°
Old	<i>either</i>	°	°	°	°	F
	<i>upper</i>	D	°	°	°	°
	<i>lower</i>	°	°	°	°	°

Note: Letters indicate where data from the examples given in Table 50 would be recorded.

In the example the multiplicity of characters depends only on object-parts and character decomposition models (p. 116) offer no advantage over DELTA-like character/state models (p. 104).

No current information model for descriptive data explicitly proposes such a complex object hierarchy as shown in Fig. 102. However, any object-oriented system that does not provide for alternative multidimensional methods to provide character specifications will encounter it. For this reason, the information models in DiversityDescriptions, Genisys/Nemisys, and Prometheus all propose some form of modifier-like solution.

A special problem in the practice of DELTA data sets is, however, that the content authors tend to avoid creating additional characters and prefer to add new character states. Over time, when these states become too numerous and cause confusion for users, some authors remove the specialized states and transfer the information to unstructured textual notes where they are mostly inaccessible for analysis and identification tools.

An interesting aspect of the example is the desire to either specify a location, or not (e. g., “leaf spots in general”). This is partly a question of a generalization hierarchy (compare Fig. 80, leaf upper/lower side, p. 168), but may also be a question of data quality. If expressed precisely, one may want to distinguish between “both sides of leaf” – i. e., studied and distribution is consistent – and “entire leaf” – i. e., the observer did not care or record the location. If truly both upper and lower sides have been studied, this is worth recording. Furthermore, in legacy descriptions a statement “leaf spots grayish” may imply knowledge that the fungal group defined in the scope of the entire treatment only occurs on the lower side of leaves, and only at certain times of the year.

Modifier classes

Several classes of modifiers are rather distinct and should be distinguished. Some modifier classes have special properties; others are useful to simplify terminology management and evolution. The following modifier classes will be discussed:

- Spatial modifiers (also called “location” or “topological” modifiers);
- Temporal modifiers (p. 204);
- Method modifiers (p. 205);
- Frequency modifiers (probability of observing a true statement, p. 206);
- Certainty modifiers (probability of a statement being true, p. 207);
- Approximation modifiers (p. 209);
- Misinterpretation hints through modifiers (p. 209);
- Negation through modifiers (p. 211);
- State modifiers (p. 212, modification of quality, degree, manner, etc.);
- Reliability modifiers (p. 213);
- Other modifiers (p. 214).

Spatial modifiers

Biological objects are often inhomogeneous: e. g., a plant may be hairy at the base of the stem, but glabrous towards the tip. It would be possible to create separate characters for the base and the tip of the stem. However, this a) requires to score two characters in the majority of plants where the stem is homogeneously hairy or glabrous, b) makes it difficult if the data source leaves it open whether the base has been studied at all, and c) if another plant is glabrous only in the lower middle of the stem, further characters have to be created. Such an inflation of characters makes it difficult to analyze the data, and causes confusion when the data are used for identification purposes. An actual example (Seethalakshmi & Muktesh Kumar 1998) may illustrate the relevance of location references in descriptions:

“Thamnocalamus falconeri [...] Leaves 10 cm long and 1.2 cm broad, oblong-lanceolate, thin, base alternate into a short petiole, apex acuminate, scabrous on the edges, smooth on both surfaces, leaf sheath long glabrous, striate, callus minute, ligule elongate, hairy. [...] Prickles costal and intercostal, frequent, more towards the margins of the leaf, base round with short pointed apex, on the costal zone arranged in rows. Microhairs intercostal, common, bi-celled, small, basal and distal cell equal in length, base filled with vitreous silica. Macrohairs infrequent to rare, costal, present towards the leaf margin, short to medium in length, base slightly raised.”

Clearly, some of the location references in the example will be part of character definitions or the object composition ontology. In other cases this would seem awkward causing an inflation of characters or object parts.

Unfortunately, the choice between these options is not clear cut and subject to personal preferences. It will often depend on the planned or expected usage of data sets, which may cause instability if these purposes change over time. However, a mixed use of character ontologies and spatial modifiers may be a necessity, as shown above (“Modifiers as an alternative to character proliferation” p. 201).

Applicability: Spatial modifiers (as well as temporal modifiers discussed below) modify the concept of what a character variable is expected to contain. These modifiers may be viewed as instantiating a modified character variable derived from a base character variable. As a result, spatial modifiers are applicable to all character types (categorical, quantitative, free-form text) and to sample data as well as aggregated summary data (including statistical measures, see “Standard aggregation methods”, p. 85).

Since spatial modifiers create “derived characters”, multiple information items are grouped by (or nested within) these modifiers. For example, if stem diameter and color is measured at various locations, each modifier would group the categorical states or statistical measures (like mean, minimum, maximum, standard deviation), respectively. In contrast to modified categorical data (where “stem brown at bottom, yellow in the middle” may be generalized to “stem brown or yellow”), similar generalizations of statistical measures are undesirable. For example, “stem diameter mean at bottom 3 cm, maximum at middle 2 cm” may not be generalized to “stem diameter mean 3 cm, maximum 2 cm” by ignoring modifier information.

In contrast to many other modifier types, the use of spatial modifiers (as well as temporal and method modifiers discussed below) on free-form text characters or states is desirable, because the concept of the character is modified. As a consequence, when mapping multiple character terminologies on each other, it may be necessary to include modifier usage in the mapping.

Status: Spatial modifiers are available in DiversityDescriptions, Nemisys/Genisys and Prometheus (region modifiers).

Aggregation issues: If class descriptions are aggregated to a higher level (e. g., multiple species to genus) modifiers may be handled in special ways. Aggregating “hairy in center” and “hairy at margin” may result simply in “hairy in center or margin”. Ordered modifiers that after aggregation have more than two consecutive modifier categories present may be reported as modifier ranges rather than lists of individual modifiers.

If many different spatial modifiers occur in such an aggregation, it may be desirable to generalize the statement. This can be achieved either by ignoring the spatial modifiers or by rendering it as “hairy, at least in parts”. The generalization may simply be motivated by the desire to create, concise, readable statements (which “hairy on the margin, or in the center, or on the veins, or between veins” is not...).

However, it may also be possible that the list of modifiers applied to a descriptive statement is “saturated”, i. e., it contains all possible modifications after aggregation (e. g., “flowers rarely, frequently, or always blue”). A saturated statement is uninformative. This can be detected only if the modifier set defined in the terminology informs whether such a saturation situation requires special handling. If in the example, the set of modifiers in the terminology contained only “sometimes” or “rarely”, the “saturated” statement would still be informative. Furthermore, aggregated modifiers may inform on variability of expression (e. g., “weakly or strongly rough”), which also may remain informative. A modifier set metadata option “*suppress-if-saturated*” may be desirable in the future. It is not included in the first versions of SDD.

In practice, aggregation issues are rare since in most cases at least one of the aggregated descriptions does not use any modifiers. Any unmodified statement has precedence over modified statements (e. g., “leaves hairy” and “leaves hairy at the margin” becomes “leaves hairy”).

Temporal modifiers (diurnal, seasonal, etc.)

As discussed above (p. 162), biological objects frequently change over time. Often knowledge of the correct time is considered implicit. For example, fruit characteristics are expected to be observed in mature fruits, not on immature or overwintered fruits. However, when creating an identification key to plants in winter it is desirable to differentiate observations on overwintering fruits from the “standard” values. Similarly, the fact that emerging leaves are pubescent only the first days will usually be entirely ignored (due to inappropriate time of observation). However, if young leaves are pubescent for extended periods, a differentiation between young and old leaves is desirable. An extreme example is presented in the case of dimorphic species, with markedly different generations in the year (e. g., *Araschnia levana*, Lepidoptera).

One possible method to express such knowledge is the use of temporal modifiers such as “when young”, “when old”, “in spring”, “in autumn”, “immediately after emergence”, “during high tide”.

During identification, absolute temporal modifiers (e. g., “summer”, “during high tide”) could provide interesting query options (e. g., “show only stages in winter”). However, this requires special data set where the majority of descriptions in at least some characters explicitly use these modifiers. Most temporal modifiers are considered implicit, and added by humans based on secondary knowledge. Query algorithms will generally lack this implicit knowledge and have to assume that unmodified “flower blue” is observable in summer and in winter. Extending the current information model with option to inform about implicit modifier value might be an interesting area of future research.

Relative temporal modifiers (oriented relative to the development phases of the organism) will currently be mostly informing humans and not used for machine reasoning. However, given that the problem of implicit modifiers is adequately solved, the identification process may prefer combinations of predicates that occur at the same time in the development process. For example, if leaf size is recorded separately for seedlings and adult plants, without a constraint based on temporal modifier equality, an adult plant with small leaves could be misidentified as a plant where only the seedlings have such leaves.

The future possibilities of temporal modifiers are not yet adequately explored. Several systems of temporal logic exist that explicitly deal with reasoning involving time. A prerequisite for studying the usefulness of adding temporal reasoning to identification or other analysis processes is, however, the existence of data sets using temporal modifiers at all.

Applicability: Almost everything that has been said about spatial (location) modifiers applies to temporal modifiers as well. Both spatial and temporal modifiers are defined on dimensions in a coordinate system, which may be absolute or relative. Absolute temporal modifiers bound to day, lunar month, or year correspond to absolute spatial modifiers (e. g., geotropical modifiers), and temporal modifiers relative to developmental stages of the organism itself correspond to relative spatial modifiers (proximal, abaxial, etc.). In all aspects studied, the applicability of temporal modifiers is identical to that of spatial modifiers; and it may even be desirable to propose only a single category of spatio-temporal modifiers.

Status: Temporal modifiers are available in DiversityDescriptions and Prometheus.

Method modifiers

Similar to location and time, it may occasionally be desirable to shift minor methodological details from the character definition (or the decomposed character ontologies) into the description by providing “method modifiers”. Methods are particularly important in organism groups that require laboratory procedures for their description. However, methods are relevant to classical morphological data as well. For example, the result of observing surface structures may differ depending on whether the unaided eye, a hand lens, or a stereo-microscope is used.

A special class of modifiers is referring to natural growth conditions (e. g., “in the shade”, “in shallow water”). In analogy to controlled growth conditions in microbiology (culture medium, temperature, etc.), these may be considered a kind of method modifier, but different solutions are possible. The modifier discussion is a special case of the method/instrumentation/property discussion above (e. g., Fig. 87, p. 177). Currently, it remains an open question whether a more detailed classification of “method modifiers” (e. g., “instrumentation modifiers”) is desirable.

Applicability: The applicability of such method modifiers is identical to those discussed for spatial and temporal modifiers.

Status: Although no method modifiers are contained in the template list (Table 46, p. 194), it is possible to create them in DiversityDescriptions. Because of the unresolved questions around method classification, no explicit category for method modifiers is proposed for the first version of SDD; instead method modifiers may be handled by the general/other modifier category in SDD 1.0.

Frequency modifiers

If a property of an object has multiple states (i. e., polymorphic), the frequency of these states is often recorded (Fig. 103). Polymorphisms may occur on the class level (population, species, genus, etc.), but also within individuals (see “Aggregation within individuals”, p. 93). Frequency information is either expressed directly through values or ranges (e. g., “20-50%”), or through categorical statements (e. g., “usually”, “occasionally”, “rarely”, etc.). Although frequency values or ranges are more informative for data analysis and identification, they are often more time-consuming to acquire and only categorical frequency information is recorded in most cases.

When categorical frequency modifiers like “often” or “usually” are not accompanied by semantic definitions, the only advantage gained over free-form comment text is improved consistency and multilingual translation. To also make them analytically accessible, a ranking of frequency terms (e. g., “usually” > “often”) or a frequency range (e. g., “usually” = 25 to 75%; “often” = 10 to 40%) may be defined. Providing only a ranking of frequency terms is often simpler, but only frequency ranges allow a joint analysis of descriptions with direct frequency values or ranges and categorical frequency statements.

Semantic information may be provided either by the original author (as annotation of intent) or estimated by a later reviewer (as an interpretation). Information about the difference between intent and interpretation may be relevant for analysis.

Occasionally, frequency statements do not refer to a simple polymorphism, but hide more complex information. For example, a characteristic may vary at different times of observation (e. g., spring, autumn) or between parts of the organism (e. g., hairiness on different parts). Thus “glabrous, rarely hairy” may really mean “glabrous, hairy when emerging in spring”, or “glabrous, hairy at the tip”.

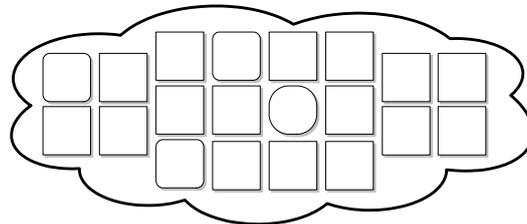


Figure 103. Objects that belong to the same class may have variability. The population of objects shown may be described as “square, sometimes with rounded corners (15%), rarely appearing almost circular (5%)”.

Applicability: Frequencies can only be determined for categorical and other forms of discrete data (especially counts). Continuously varying quantitative data are, however, often mapped to discrete categorical states, resulting in a special form of frequency distribution (histogram, p. 66). Both property values (e. g., “elliptic”, “round”) and object composition information (e. g., “presence”, or “count”) may be polymorphic. In the case of descriptive information models following the character model (p. 104), these are treated identically so that frequency is applicable to all categorical characters. Frequency modifiers are not applicable to statistical measures, including those that may be defined on categorical data (e. g., mode or median).

The designer of a terminology may desire to further limit the applicability of frequency modifiers to individual states (e. g., allowing a full set of “very rarely”, “rarely”, “occasionally”, etc. in some characters, but only “rarely”, “usually” in others). This is, however, not a question of data integrity, but rather of recording consistency.

In principle, the use of frequency modifiers implies the existence of more than one state for a character in an object or class description. The statement “usually hairy” alone is incomplete. Humans are able to provide the implied “... or rarely glabrous” from background knowledge, but this is usually not available to algorithmic analysis. For a data recording application it may be

desirable to issue a warning in this case. However, the information model should not require the scoring of multiple states, since incomplete and ambiguous information may be present in legacy data. Modern ontology languages such as OWL permit one to externally represent enough information to deduce implications such as “usually hairy” implies “rarely glabrous”, as long as only categorical expressions are involved (R. Morris, pers. comm.).

In contrast, it may be desirable to constrain descriptions to at most one frequency modifier per state. Statements like “usually or rarely present” make little sense both to humans and algorithms. They do exist occasionally in legacy data, usually as a result of careless data aggregation procedures (e. g., as a result of abbreviating “usually present in the North, rarely in the South”). However, to simplify the information model and to avoid case logic in all data consuming applications (analysis, identification), it may be desirable to require resolving this problem at the time of data entry.

It is probably not desirable to provide for both a quantitative frequency statement (value or range) and a categorical frequency modifier on a single state in a description. In principle this information may be congruent (the quantitative information being more exact than the categorical). The case is therefore different from the application of multiple categorical frequency modifiers on a single state. However, if a frequency value is known and if the categorical frequency modifiers in the terminology are defined with a frequency range, an algorithm would be able to choose an appropriate modifier term where such output is desired instead of quantitative values.

Status: The SDD workgroup discussed frequency modifiers at the SDD meetings in Australia, (March 2002), Brazil (2002), and Paris (2003). SDD provides in descriptions for categorical as well as quantitative frequencies statements (Value, or LowerEstimate and UpperEstimate, introduced at the SDD meeting in Brazil, see minutes: Hagedorn 2003a). In SDD terminology, a modifier definition includes mandatory labels and optional natural language wordings (both optionally in multiple languages). Further, it is possible to rank frequency modifiers and provide a frequency range definition. The problem of ranking was raised by R. Vignes Lebbe at the Paris meeting of SDD in 2003, and included in the SDD proposal presented in Christchurch 2004.

Aggregation issues: Two descriptions “a or very rarely c” and “b or rarely c” can easily be aggregated into “a, b, or rarely c”. However, it may be desirable to suppress states present only with very low frequencies in the generalization/aggregation process, e. g., when aggregating species descriptions into family descriptions.

Whether and when such suppression becomes desirable depends on the abstraction level. Inclusion of a rare state may be desirable in a genus description, but ignored in the description for the order. “PropagateFrequencyOnAggregation” is thus a property of the process, rather than a metadata attribute on frequency modifiers in descriptions.

Relation to formal logic: Normally the structure of modifiers is: “modifier applied to proposition” (“usually [flower is blue]”), which in predicate logic would correspond to second-order predicate logic/higher predicate calculus. For the special case of the principal distinction between “always” (frequency = 100%) and “sometimes” (frequency < 100%), frequency information is related to the concept of universal (“for-all”, \forall) and existential (“exists”, \exists) quantifiers in Boolean predicate calculus. Reducing frequency modifiers to “always”, “sometimes”, and “never” reduces the problem to first-order predicate logic (i. e., lower predicate calculus) as in (\exists flower: flower is blue) *versus* (\forall flower: flower is blue).

If reasoning based on fuzzy logic is being employed, frequency information may be used as a “degree of truth”.

Certainty modifiers

The categorical or quantitative value of a character in an object description may be known, unknown, or uncertainly known. The case that character values are entirely unknown has already been discussed under “Coding status” (p. 74). This is different from an uncertainty that a specific statement is true (e. g., “probably elliptic”). Examples of certainty modifiers are: “probably not”,

“perhaps”, “probably”, “likely”, “very likely”, and “almost certainly”. Uncertainty in descriptions may express that the scientist creating the description:

- (when recording from original material:)
 - is confused about the interpretation of (perhaps poorly worded) terminological definitions,
 - has methodical observation problems (for example, values are out of the range of normal observation method),
 - suspects or knows about preparation or conservation artifacts in the observed material (for example, color bleaching or shape of deep sea fishes).
- (when recording from a published data source:)
 - doubts the factual correctness of information in the data source,
 - is uncertain about the relationship between the descriptive terminology used for data recording and the terminology used in the data source.

Certainty modifiers may provide an extensible way to define a terminology of both generic and specific certainty modifiers. An example for a specific certainty modifier is “probably (material poorly conserved)”. Applying these modifiers to descriptive statements provides a method to express doubt in an analytically traceable way. The certainty of knowledge is especially valuable to preserve doubt when scoring specimen data. Indications of doubt may later be removed after an adequate sample of specimens has been studied, but provide a means to avoid the propagation of error during data aggregation and generalization.

Certainty and frequency modifiers both express a probability that a given object will express a given value. Certainty is the probability that the value is expressed at all, frequency is the probability that a statement applies to a given object where the class is polymorphic. In statistics, certainty modifiers may be used in Bayesian statistics (“Bayesianism” supports – among other kinds of *a-priori* probabilities – degrees of belief as a basis for statistical calculation), frequency modifiers to the “standard” statistical hypothesis testing (parametric tests such as analysis of variance or regression analysis as well as non-parametric tests such as Kruskal-Wallis).

Similarly to frequency modifiers, it is desirable to add semantic information (ranking by increasing probability, or providing quantitative probability ranges) to the definitions of certainty modifiers.

Whereas for frequency modifiers the desire to directly record frequency values or ranges rather than only the modifier categories in descriptions is very strong, this is less so for certainty modifiers. Exact certainty values for statements in the description occur extremely rarely in practice (some “identification” processes, like automatic chromatographical detection of substances do provide probabilities of correctness). However, if the information model is simplified by treating frequency and certainty modifiers analogously, supporting certainty values and ranges would not be problematic.

Applicability: Certainty modifiers are applicable to categorical and quantitative values. Statistical measures (such as mean or confidence intervals) already provide a form of measuring (rather than guessing) uncertainty. The applicability of certainty modifiers to these is doubtful. However, doubt may arise either as to the quality of the underlying data or the applicability of the statistical method itself. Thus, while the expression of such doubt with a certainty modifier may have no formal mathematical relation to the rigorous calculation of a statistical measure like a mean, it is occasionally useful to signal such a lack of confidence. In contrast to frequency modifiers, certainty modifiers may therefore be judiciously applied to statistical measures.

Certainty modifiers may also be applicable to free-form text characters or states. Because of the unstructured nature of free-form text, however, there is only a marginal benefit of doing so. If it is desirable to simplify the information model by not supporting modifiers on these data types, little is lost.

Note that saying an entire character is “uncertain” without giving any character value information (as in “leaf shape uncertain”) is indistinguishable from saying that it is “unknown”. It is a coding status (p. 74), clearly distinct from certainty modifiers expressing metadata about a value statement. Certainty modifiers are therefore not applicable to entire characters.

Relation to formal logic: Certainty information is the standard test case where modal logic (which may be applicable beyond these) is discussed.

Status: Uncertain knowledge is called “guessed values” in Dallwitz (2006d).

Related to certainty modifiers are the issues of approximation (e. g., “ca. 3 mm”, “about elliptical”), negative statements (e. g., “certainly not”), and perhaps even misinterpretation hints (e. g., “certainly not, but commonly misinterpreted as such”). These are discussed in the following.

Approximation modifiers

Approximation modifiers, such as “ca.”, “approximately”, “about”, “roughly”, “nearly”, “more or less”, or “almost”, express doubt in the accuracy (nearness to the true value) or precision (reproducibility) of a statement. Approximation could be interpreted as a kind of certainty expression: “perhaps exactly so, but certainly close to it”. More appropriately perhaps, approximation modifiers are interpreted as a short cut notation for fuzzy value ranges or confidence intervals. They are used if the range exceeds the default conventions for scientific data that are expressed in the number of decimals given (based on scientific rounding rules, “2.0” indicates “1.95-2.05”).

Different modifiers are categories that may express different degrees of inaccuracy of a reported value. Potentially, quantitative information about this degree of inaccuracy could be defined or estimated together with the modifier definition, improving machine reasoning and analysis routines. Both absolute (e. g., “ ± 2 units”) and relative (e. g., “ $\pm 5\%$ ”) accuracy values could occur. In principle, accuracy may be expressed as a range for which a formal confidence interval is given (e. g., “accurate to $\pm 5\%$ of value, with a 95% probability”). The occurrence of such a probability value could be interpreted as indicating a relationship between accuracy and certainty modifiers. However, most accuracy modifiers are broad categories and no practical examples could be found for such use.

In fact, even rough accuracy values are probably extremely rarely defined for approximation modifiers. No publication known to the author defines the semantics of “ca.”, “about”, etc. Also, no general implied definition is known to the author. Implied knowledge may exist, however, for a given methodology. For example, the measurement of small objects in a light microscope using oil-immersion optics has an implied accuracy between “ $\pm 0.3 \mu\text{m}$ ” and “ $\pm 0.5 \mu\text{m}$ ”. Mean spore size may then be reported as “ $2.2 \mu\text{m}$ ”, “ca. $2.2 \mu\text{m}$ ”, the latter emphasizing the fact that accuracy is less than the expectation based on rounding rules would suggest. Where accuracy depends on methodology, it may be more appropriately expressed in the terminology rather than in individual descriptions. For this reason, in the present version of SDD, approximation modifiers are not yet accepted as a separate modifier type with explicit semantics and methods to express accuracy quantitatively, but subsumed under the “general modifier” category. This topic should be discussed further.

Applicability: The issue of approximation exists both for categorical (e. g., “about elliptical”) and quantitative (e. g., “ca. 2 mm”) data types. It is not applicable to statistical measures and free-form text characters or states.

Aggregation issues: “ca. 2 cm” and “2 cm” should be aggregated to “ca. 2 cm”. “ca. 2 cm” and “5 cm” could be aggregated to “ca. 2-5 cm”. However, as long as no quantitative measure of approximation is known, aggregating an approximation with a range is not easy. Would “ca. 2 cm” and “1-5 cm” become “1-5 cm” or “ca. 1-5 cm”? Would “ca. 2 cm” and “1.9-5 cm” become “1.9-5 cm” or “ca. 1.9-5 cm”? It may be possible to develop some heuristics based on the implied precision of a statement like “ca. 2 cm”, but an exact method is not possible.

Misinterpretation hints through modifiers

In an ideal world, the application of well-defined descriptive terminology to the objects to be described or identified would be unambiguous. In reality, however, it is very difficult to produce

exact and concise definitions. Not only will long definitions often be ignored (and are impossible to memorize), also most users will not even attempt to consult a definition if they believe they already understand a term (and the introduction of ever new terms to avoid this is not very desirable, either). Furthermore, even when the definition is fully understood, the exact process of determining whether an object part or property fits the definition may often be impractically complicated (involving, e. g., developmental studies), necessitating the use of “shortcut” definitions that result in the correct application in most, but not all cases.

In the practice of biological descriptions and authored identification keys, the problem is handled by addressing a certain level of expertise (usually that of general university training) and adding explicit misinterpretation hints for common or all known cases of misinterpretations. These hints take the form of annotations in natural language descriptions, but often take the form of “false” leads in authored branching identification keys (i. e., a taxon is keyed out in a branch of the key that does not fit its true description).

In the case of a descriptive data matrix intended both for the generation of natural language descriptions and identification keys, these issues are difficult to separate, and a special mechanism is required. If the designer of a data set adds false statements to preempt misinterpretations of users without distinguishing them from true statements, the data set degenerates and becomes difficult to manage and revise. Furthermore, from then on it can be used only for identification purposes and not, for example, for phylogenetic analysis.

Several types of misinterpretation can be distinguished:

1. The organism part (or “structure”) is generally likely to be misinterpreted
Examples: a phylloclade (cladode) is interpreted as a leaf, or a rhizome as a root.
2. The organism part is likely to be misinterpreted within a given taxonomic group
Example: the inflorescence is often interpreted as a flower in *Euphorbia*; the bracts of *Cornus florida* (dogwood) are interpreted as petals.
3. The property state is generally likely to be misinterpreted.
Example: a spore surface that is visibly rough in a good microscope may be interpreted as smooth because of insufficient optics or inappropriate handling of the microscope.
4. The property state is likely to be misinterpreted within a given taxonomic group.
Example: Leaves of *Lotus corniculatus* are palmate with 3 leaflets plus 2 leaflet-like stipules, but often misinterpreted as pinnate with 5 leaflets.

The cases differ in how they ideally would be handled:

1. In the first case, a generic misinterpretation tolerance mechanism (e. g., a mapping) of the organism part would be desirable. For example, if a root character is specified during identification, the identification application could search both under “root” and “rhizome” characters. This general error/misinterpretation tolerance mechanism would then automatically apply to any character in the descriptions.
2. In the second case, a taxon-specific mapping of organism parts would be desirable.
3. The third case could be solved by a state mapping that adds error tolerance to the identification for all taxa.
4. In the last case, a special attribute could be added to the state for each object or class description giving rise to a misinterpretation.

In principle, all cases can be managed with description \times state-specific misinterpretation attributes (case 4). This would, however, often require extensive work by the content authors to add numerous misinterpretation statements to all characters based on a misinterpreted object part in all taxa affected. A special mechanism to “map” concepts for parts may be desirable here. This mechanism is probably already implicit in the generalization hierarchy for object parts discussed above (p. 153). Both the root and the rhizome could, for example, be generalized to a “root-like underground structure”.

Regardless of other mechanisms, description \times state-specific misinterpretation attributes (case 4) are desirable. Using a kind of modifier for this is not a necessary conclusion, but if modifiers can also handle misinterpretation hints, the general information model could be kept simpler. Furthermore, some structural similarities exist between such misinterpretation modifiers and certainty modifiers:

- The certainty/probability range discussed for certainty modifiers makes sense insofar as a probability of ‘0’ correctly expresses the likelihood that the state is indeed present. Scoring “leaf shape elliptic (by misinterpretation)” in a species having phylloclades instead of leaves can be interpreted as being scored for the purpose of identification, but for other purposes (e. g., phylogenetic analyses) the probability that this is indeed so is ‘0’.
- Certainty modifiers cannot logically occur together with a misinterpretation modifier: “flowers probably white (by misinterpretation)” does not make sense, because here the author of the information is certain that this is not so, but the author suspects that consumers of the information believe otherwise (perhaps because they misinterpret a structure as a flower).
- Only probability and misinterpretation modifiers (but not frequency or general modifiers) are applicable to statistical measures.

On the other hand:

- In the statement “probably white (by misinterpretation)” as rejected above, certainty modifiers express the likelihood that a statement is factually correct. It would, however, also be possible to also express a “misinterpretation probability”, where a value “0.9” would mean: “a random scorer will score this so with 90% probability by misinterpretation”. A need to express such information would suggest handling misinterpretation modifiers separately from certainty modifiers.

Status: In DELTA applications like Pankey or CSIRO DELTA, misinterpretation information can only be stored as a comment. CBIT Lucid pioneered the use of misinterpretation scoring and fully supports it in a structured way. DiversityDescriptions supports misinterpretation markers through structured modifiers. Although this has a consistency advantage over DELTA comments, the initial versions shared with DELTA the problem that algorithms could not automatically recognize misinterpretation modifiers. Starting with DiversityDescriptions 1.8, a new Boolean attribute was added to modifier definitions, allowing the recognition of misinterpretation modifiers.

SDD (since version 0.9) attempts to handle state-specific misinterpretation hints with a special form of certainty modifiers: “certainly not (but true by misinterpretation)”. A special Boolean attribute “IsTrueByMisinterpretation” in the modifier definition of all certainty modifiers is set to true for misinterpretation modifiers. In addition, the ProbabilityRange of these modifiers should be set to “0..0”. The advantage of this is that software designed to handle the certainty probability ranges will automatically produce correct analysis results, whereas simple identification software will most likely ignore the modifier altogether, which results in the desired error tolerance.

Negation through modifiers

Natural language descriptions and especially branching identification keys (i. e., dichotomous or polytomous keys) often contain a mixture of positive and negative statements like the following:

- “flowers orange, but usually not red or yellow”
- “flowers red or yellow, but never orange”
- “flowers dark orange, but never red, brown, or yellow”

Such statements usually attempt to clarify the interpretation of a character state by adding a negative expression that can be expressed with greater certainty. In principle, if orange is appropriately defined in the terminology, the addition of “but usually not red or yellow” is redundant. In contrast, “flowers red or yellow” may well be misinterpreted as a range from red to yellow, including orange. The purpose of adding “never orange” is to preempt a possible expectation that

the author may not have distinguished between “red or yellow (but no intermediates)” and “between red and yellow”.

A related situation is the case of single negative statements, like:

- “flowers not green”,

which usually result from abbreviating a long list of alternative states. This situation occurs especially on a higher taxonomic level. It is unusual in descriptions, but does occur in identification keys. Such a “negative state” may already be defined in the list of character states. This is obvious in a case like “symmetrical/not symmetrical (or asymmetrical)”, but is implicitly present also in a state list like “round, elliptic, lanceolate, ..., irregular”.

Whether negative expressions should be handled through a modifier mechanism or not is an open issue. On the one hand the modification is extreme and requires substantially different handling in analysis or identification than other modifiers. “Negating modifiers” must be recognized, i. e., the modified state treated as non-scored. In addition, the complement of non-negated states could be treated as “scored”, modified by “perhaps”.

On the other hand, if certainty modifiers are defined with probability ranges like:

- “probably”: > 50 and $\leq 99\%$ probability,
- “perhaps”: > 5 and $\leq 50\%$ probability,
- “probably not”: $\leq 5\%$ probability,

then a negative expression may be viewed as:

- “certainly not”: 0% probability.

Wordings for such “negating modifier” could be: “, but not”, “, but never”, or “, and never”. This would require an additional mechanism to suppress the normal delimiter (‘and’, ‘or’, or comma rules) for enumerated states in the natural language output.

Furthermore, additional difficulties arise if several states are explicitly negated as in “A, but not (B, C, or D)”. The central problem is that the brackets shown are not used in natural language representations, and difficult to represent in the user interface. When humans read: “flowers orange, but not red, brown, or yellow” they will normally interpret implied brackets, parsing the semantics as “flowers orange, but not red, and not brown, and not yellow” (ultimately using Morgan's rule, changing or to and!). Making this transparent to the user in a user interface requires a substantial effort by the application designer.

For the first case (negative statements in addition to positive statements), an alternative handling is to use free-form text notes, for example: “flowers red or yellow (‘but never orange’)”. This does not work in the second usage case, where negative statements are used as an abbreviation. It is possible to express the latter by scoring the complement of states with the certainty modifier “perhaps” added, but this results in unwieldy statements that are difficult to consume for humans, and are further not stable when the terminology evolves and new states are being added.

Applicability: Negative statements as discussed are largely applicable to categorical data. A related situation exists where quantitative range statements (e. g., “2-10 cm”) require contradiction (e. g., “but not 2-4 cm”). In a single data set, single-user environment, the statement would simply be revised (e. g., “4-10 cm”). However, in a collaboration scenario it may not be possible to “overwrite” the erroneous statement and contradiction may be necessary to prevent aggregation (without contradiction, aggregating “2-10 cm” and “4-10 cm” would result in “2-10 cm”).

Status: No application or information model for descriptive data in biology that handles negative descriptive statements (statements intended to contradict other statements) is known to the author. Whether a modifier mechanism is appropriate to solve this situation is an area for future research. Consequently, negative modifiers are not included in the current versions of SDD.

State modifiers

The modifiers discussed so far either affect the character definition (location, time, method, condition) or add an entirely independent dimension to the statement (frequency, certainty, approximation). In contrast the following “state modifiers” create derived states from a fundamental base

state. In natural language these modifiers would be called modifiers of quality, degree, emphasis, or manner. Examples are “very”, “weakly”, “slightly”.

The state modifier mechanism is a convenience rather than a necessity. Instead of creating narrowly defined states by combining a modifier with a state, the state list itself could already contain all narrowly defined states. In classical DELTA-based applications, long state lists may be confusing during data entry or identification. However, if the relations between wide and narrow state concepts are defined (compare “Mappings within categorical data”, especially Fig. 23, p. 69) and presented as a hierarchical tree, this may be a stable and practical solution.

On the other side, the combination of basic states with state modifiers offers a practical solution in cases where most states in a set may be modified in a regular manner (e. g., emphasized with “strongly” or “weakly”). An example where a full state list may be exceedingly large and complex is color. Using modifiers to combine major color categories with shade and intensity modifications (e. g., “greenish”, “bright”, “dull”) may sometimes be desirable during data entry and identification. Although the concept of such a color-modifier is much more difficult to illustrate than individual colors, the sheer amount of colors that may have to be illustrated if each color variant is given its own state (perhaps several hundred color states to choose from) may make the “illustrated color” less desirable than the “conceptualized”.

The availability of modifiers leaves the decision when to use a modifier approach and when to use widely plus narrowly defined states together with a mapping to the designers of the descriptive terminology.

State modifiers may be abused to give a character state an entirely different meaning (e. g., creating a modifier “not recognizably”; the combination “not recognizably sharp” would probably mean “blunt”). Such modifiers may work in natural language descriptions, but will lead to undesirable results during data analysis and identification. The possibility of abuse is, however, not specific to state modifiers: any form of modifier or free-form text notes can be abused. Occasionally, it may depend on the intended use whether something is considered abuse or not: ignoring a modifier “when infected by fungi” may be desirable in identification, but undesirable in phylogenetic analysis. Controlling this is the responsibility of designers of terminology and of users performing analyses, not of the information model itself.

Applicability: By definition, state modifiers are applicable only to categorical states. In general, no examples of quality, degree, or emphasis applicable to other character types have been found so far.

Status: State modifiers are implemented in DiversityDescriptions.

Reliability modifiers

The proposals for “New DELTA” (p. 20) contain the concept of “coded comments” to modify the general reliability of a character in a specific description. In general, reliability as well as other character-ranking metadata used for identification purposes (convenience, availability, etc.) are important ranking schemes to influence recommendations which characters are most suitable for identification purposes (compare “Identification methods: Authored character guidance”, p. 267).

A generic character rating (i. e. a rating defined in the terminology, as in DELTA) is not necessarily appropriate throughout the entire data set. A character may be convenient and reliable in one taxonomic group, but not so in another (Diederich & al. 1989, Diederich & Milton 1991) and reliability metadata must be organized appropriately. However, having to define character-ranking metadata separately in each description seems undesirable.

The proposal of a reliability modification in the “New DELTA” model may be interpreted as follows: global reliability is preserved as terminological metadata (as in the original DELTA model) and may be interpreted as “average” or perhaps “default” reliability. It may then be modified in individual descriptions to correct for taxon-specific character reliability variation. Attribute reliability values in New DELTA are intended to “modify” the attribute reliability of

the character (not the character data, as other modifiers would). This model is certainly pragmatic and is likely to work well for relatively small, taxonomically focused revisions or monographs.

However, this model inherits only a single character reliability value to all descriptions using this character, which then has to be modified in each individual description. In large projects spanning substantial taxonomic diversity, a better inheritance model is desirable, inheriting character reliability metadata automatically down the taxonomic hierarchy, until a new rating occurs. Such inheritance would not easily be achieved by a reliability *modifier*, i. e., a structure that is modifying a value and inheritable only in combination with a value. A separate inheritance of ratings is desirable. Although this may be considered similar to a modifier structure, it is not truly the same.

Character-ranking metadata may also have an interaction with character values rather than with the taxonomic tree. For example, a character representing length measurements may be convenient and reliable for large taxa, and inconvenient for small ones. Although this will often be broadly correlated with taxonomic groups, this is not necessarily so. Whether such a character \times value interaction is common enough to warrant the introduction of reliability modifiers (in addition to inheritable character reliability ratings) requires further study.

Other modifiers

During the development of the modifier concept, it was initially expected to discover a finite set of modifier classes that are relevant to descriptive data. However, as more and more potential uses have been discovered, it is believed that the information model should be extensible, supporting new user defined modifier classes.

In principle, any secondary information that may be placed in notes or comments attached to character data may also be handled in a more structured form through modifiers. This is generally appropriate if certain kinds of comments are reoccurring, and if some analysis desires to distinguish between them. The two major usage categories are:

- Constraints or conditions under which the data are valid, for example:
 - environmental conditions like “on wind-exposed slopes”,
 - life cycle stages or generations (e. g., “seedling”, “caterpillar”, “summer generation”), or
 - character correlation issues like sex (see “Secondary classification resulting in description scopes”, pp. 217 and 218).
- Information that identifies or qualifies the source of information. In DiversityDescriptions data sets, modifiers have been used to:
 - identify the source publication from which recorded data are derived, or
 - identify data that are based on type material itself.

Handling cases like these through modifiers is certainly not ideal – e. g., SDD provides explicit mechanisms to record the source of information – but should nevertheless be expected and tolerated where such flexibility is needed for users. Given the limitations of specific descriptive data applications and the time resources available, the use of modifiers may offer new options for managing descriptive data projects.

Character- versus value-modifiers

The modifier classes discussed so far (location, time, method, frequency, certainty, and manner/degree) can potentially be classified into two super-classes:

- “Character modifiers” address the problem that the number of characters (i. e., the combination of object part + region or location within that part + temporal definition + property + observation method in character decomposition models) often become undesirably large due to minor variations (compare “Modifiers as an alternative to character proliferation”, p. 201).

- “Value modifiers address” the problem that minor variations of categorical values and the aspects of frequency and certainty may result in an undesirable increase of the number of categorical states.

In the SDD 1.0 beta 2 proposal (for overview of available documentation see Hagedorn 2004c), character and state modifiers were distinguished. The structural complications for the SDD exchange format were, however, considered too large and the proposal to separate two kinds of modifiers was rejected at the TDWG meeting in Christchurch 2004 (see minutes of SDD session, Hagedorn 2004d). The fundamental difference between a modification of the definition of the variable, and a modification of the value was not disputed. However, the SDD proposal at that time attempted to define character modifiers as any modification that was in principle applicable to all kinds of character types. This leads to a different division, e. g., in the case of certainty modifiers (e. g., “probably”). These are applicable to all kinds of character types (categorical, quantitative, but potentially also molecular sequences), but modify the value (or result) rather than the variable concept.

The choice of a single extension concept for both variable and value modification is based partly on the reasoning that modifiers can be understood as a modification of the statement. Thus “petals *usually* red” can be regarded as: “the assertion ‘petals are red’ is *usually* true”, “*probably* blue eyes” as: “the assertion ‘eyes are blue’ is *probably* true”, “stem hairy *at the tip*” as: “the assertion ‘stem hairy’ is true *at the tip* of the stem”. In logic this method is called a reification of a statement (reify = make a thing of something), which is, e. g., supported in OWL (McGuinness & van Harmelen 2004). Following reification, new statements can be made about the reified statement.

It must be noted, however, that when using the modifier concept to create subclasses of character states, the reification perspective becomes less convincing. Treating “*slightly* rough leaves” as: “the assertion ‘leaves are rough’ is only slightly true” is a logical error which only superficially looks correct. This can be seen in the case of “*pale* blue eyes” which may be interpreted as: “the assertion ‘eyes are blue’ is only true in the wider sense of blue, the exact narrower sense is *pale blue*”. It is uncertain whether this can be interpreted as reification at all.

Despite these problems, combining variable and value extensions into a single extension concept seems to be a good compromise. In fact, no validation mechanism could be found in the SDD discussions to constrain a modifier mechanism to purposes that clearly are a reification while preventing its use for character state modifications. Similarly, the attempt to introduce a fully logical reification mechanism at various levels seemed to overload the design with undue complexity and make implementations of the information model expensive. A single level of modifier types seems a good compromise between expressiveness and complexity of the model.

4.15. Secondary classification resulting in description scopes

Introduction

Many differences between objects are not captured by the primary taxonomic class hierarchy, even if low-level ranks such as subspecies or variety are being used. Some differences are due to random effects in the history of individual objects and do not lead to additional classification systems. Other differences are, however, systematically repeatable and – in biology – often genetically coded (sex, life cycle stages). These may give rise to additional classification systems (Table 52).

Table 52. Examples of alternative classification systems and sources of intra-class variation in biology and the study of musical instruments.

Taxonomic classification	Biological organisms	Musical instruments
Phylogenetic / Inherited (→ multiple characteristics are linked)	Evolutionary history /taxonomic classification (e. g., order/family/genus)	Craftsmanship, technological, or industrial traditions of instrument creation
Operational (arbitrarily based on a single characteristic)	Tree/shrub/herb, water vs. land plants	Sachs-Hornbostel system (idiophones, membranophones, chordophones, aerophones, electrophones)
Source of further variation		
Individual history		
a) chance effects	Scarring of skin, mutilations	Scratching, discoloration
b) systematic responses to the environment	Phenotypic responses like flowering time, leafing in deciduous plants, variable shape to maximize resource utilization,	Response to humidity or submerging in water
c) essential and repeatable history	Developmental stages: e. g., egg/embryo, larva, adult; Life cycle stages: e. g., gametophyte, sporophyte	Phases in the construction of an instrument; tuning of instruments
Genetic polymorphism	Sexes or blood types (multiple alleles for a gene present within populations)	Decorative styles spanning multiple instrument types and traditions

A characteristic that is polymorphic within a class (i. e., taxon in biology) is often still informative for diagnostic purposes. If a plant species has red or white flowers (e. g., *Ranunculus asiaticus* L.) and other species have yellow, blue, red, or white colors, specifying flower color removes some classes from the set of potential identification results. A description “flowers red or white” is a meaningful part of a diagnostic class description.

Certain kinds of polymorphisms change highly systematically. A description “sex male” is meaningful for an individual object, but “sex female or male” is meaningless in a class description. By definition, the two sexes occur together. Similarly, recording the presence of life stages may or may not be meaningful, depending on the taxonomic scope and whether all classes have a larval and an adult stage. This problem of character “saturation” (i. e., all potential character states present) can be automatically detected if a character has been recorded either for all classes or for a sufficient sample of objects. It normally does not require the recording of additional information. Some of these characteristics, however, form an operational classification system. In biology these “secondary classification systems” are independent of the primary system of taxonomic names. The most frequently encountered examples are designations of sex (male/female), generation (e. g., spring/summer), and life cycle or development stage (e. g., larva, adult).

Importantly, the values of these classifiers are not directly observable characters, but typify sets of correlated character expressions. Objects with different classifier values will have moderately or strongly different descriptions. If for a secondary classifier like “sex” the object descriptions differ only in expected characteristics (the sex organs), the values of the classifiers would normally be suppressed in the class descriptions (e. g., a species description). Other weakly correlated characteristics (e. g., males being slightly smaller than females) will be presented as a generalized description (e. g., as a size range including both sexes). However, separate descriptions will be prepared if several diagnostically relevant characteristics (e. g., wing pattern of butterflies or bird plumage), or unexpected characteristics differ between sexes. This is then called a “sexual dimorphism”. The sex values will not become part of the description, but will be used to group or structure the descriptions. Depending on the amount of differences, the grouping may precede the primary class name (e. g., taxon name), be a subheading within descriptions, or only an annotation at individual descriptive statements (Table 53). Furthermore, if different sexes or

life cycle stages are keyed out separately in identification keys, the classifier values are usually added to the name that is keyed out.

Table 53. Examples of different presentations of sex and life cycle stage classifiers.

Stage grouping preceding class name	Stage as subheading within description	Sex as note to individual statement within description
Larval descriptions <i>Colias alfacariensis</i> Ribbe 1905 <i>Colias crocea</i> (Geoffroy 1785)	<i>Colias alfacariensis</i> Ribbe 1905 Distribution: ... Common characteristics: ... Larval characteristics: ... Adult (imago) characteristics: ...	<i>Colias alfacariensis</i> Ribbe 1905 ... Larva: Size ... body green, ... Adult (imago): ... Size ..., wings white (females) or clouded yellow (males)
Adult butterfly <i>Colias alfacariensis</i> Ribbe 1905 <i>Colias crocea</i> (Geoffroy 1785)		

Storing the information about classifiers as character data is satisfactory for object descriptions, but not for class descriptions. Although sets of correlated characters can be detected algorithmically, it is difficult or impossible to detect which of the correlated “characters” are truly observable characters, and which “characters” summarize and generalize sets of character correlations.

Before proposing an information model for secondary classifiers like sex, generation, or stages, it must first be decided whether it is appropriate to generalize these to a single concept. As a first step, the most important classifier concepts in biology will be discussed.

Note: an early version of this discussion, with comments from R. Morris and B. Heidorn can be found on the SDD Wiki under the topics of “SecondaryClassifiersWithinClasses” and “SecondaryClassifiersProposal”.

Mating type and sex

Many organisms have a breeding system involving multiple **mating types** to increase the evolutionary advantageous outcrossing. Mating types may be classified as **sex** and morphological or physiological **self-incompatibility systems**. Note that instead of using “mating type” as a generalized term (i. e., including sex), many authors use it when referring to reproductive compatibility types (studied using internet search mechanisms). This may be due to these authors working on taxonomic groups that do not show morphological differentiation into sexes (e. g., yeasts).

In biological usage, **sex** is defined as the sum of morphological and behavioral features that distinguish organisms on the basis of their reproductive function (EB 2001, CED 1992). The concept of sex is fundamentally limited to two different sexes (“male”, “female”). The combination “hermaphrodite” (a single organism being both male and female), the absence of sex, combinations of sexes in parts of development stages of a single organism (e. g., in plants: monoecious, dioecious, trioecious) may lead to additional states, but overall the number of states is general and limited. In contrast, the number of compatibility types differs strongly among organism groups, as do the names used for individual types (e. g., “+”/“–”, “A”/“a”/“alpha”, “b1”/“b2”/“b3”). Mating types are usually genetically determined (an exception is, e. g., the marine worm *Bonellia* with environmental sex-determination, EB 2001).

In many animals, either sex is the only mating type, or sex and **self-incompatibility system** are always correlated. The difference between the two concepts can be seen in plants like *Nicotiana* that are sexual hermaphrodites in having both anthers and gynoecium in each individual, but have a **physiological self-incompatibility system** to prevent inbreeding. Similarly, fungi may produce differentiated male and female organs on the same thallus but remain self-incompatible (heterothallic) due to a separate physiological self-incompatibility system. Many fungi or algae

have no morphologically identifiable sex system and are classified only according to their self-incompatibility system (which is often only called “mating type”).

An example of a **morphological self-incompatibility system** (i. e. heteromorphy) is the heterostyly in plants (e. g., in *Primula* species: distyly or in *Lythrum salicaria* and *Eichhornia*: tristily). This mechanism is independent of the sex system, but closely linked with a physiological incompatibility system where present (Richards 1986).

Generations, life cycle, and developmental stages

The term **generation** in biology denotes the steps in the cycle of reproduction. Consecutive generations may be morphologically similar or dissimilar and they may be genetically different (especially after sexual reproduction) or not. The latter case may be due to vegetative reproduction (e. g., parts of a plant break off, are dispersed, and root again forming the next generation) or to apomixis (using an embryonic reproduction system without genetic exchange, e. g., producing seeds asexually). In single-celled organisms generation and cell division are synonymous (but regular or irregular morphological change occurs). Loosely the term “generation” is often applied to designate morphologically distinct “generation types”, but a more precise term is desirable.

Life cycle may be defined as “the series of changes in the life of an organism, including reproduction” (EB 2001, dictionary). Two kinds of life cycles exist (EB 2001, encyclopedia: “life cycle”):

- All stages occur within the life of an individual organism (single-generational life cycle). The life cycle may be:
 - truly having only a single generation, as in bacteria (haplontic life cycle), or
 - an alternation of haploid and diploid generations, one of which is so highly reduced that it is no longer considered a separate generation (haplontic or diplontic life cycle).
- The stages include several generations to complete a full life cycle (multigenerational life cycle).

The **life cycle stages** within a single generation are also called **ontogenetic stages**, **developmental stages** (or phases), or **growth stages** (for the latter see Pujar & al. 2006). These may either partition a continuous variation (e. g., embryo, infant, youth, and adult) or may relate to distinct structural changes (e. g., in holometabolic insects: egg, larval instars, pupa, and imago).

The term “life cycle stage” is often used as a general synonym of developmental stage (which conforms to the dictionary definition cited above). This causes no problem in organisms that complete their life cycle in a single generation, but appears unfortunate in organisms having both distinct generational stages and developmental stages within a generation.

In the case of multigenerational life cycles, both “generation” and “life cycle stage” may refer to distinct generations. The use seems to be not consistent. For example, in the red algae *Polysiphonia* the haploid generation (gametophyte) is followed by two distinct diploid generations (carposporophyte, tetrasporophyte). In a small informal survey (Google, 2007-05-01) the following terms were used: life cycle stages, life-history phases, generations, somatic phases, somatic stages, life stages, with a preference for the first term. In another example, the seasonal dimorphism (“polyphenism”) of “*Araschnia levana* gen. vern.” versus “*A. levana* gen. aest.” was mostly termed spring and summer generations, and only rarely “life cycle stages”. Thus, while *development/growth stage/phase* unambiguously refer to development within a single generation, no generally accepted term for morphologically or genetically distinct generations in a life cycle exists.

A special problem is the dikaryotization of many basidiomycetes. After the sexual partners have fused, the new nucleus divides and propagates itself through an existing cellular structure (the previously monokaryotic hyphae). It is unclear whether this should be considered a generation because of the genetic change, a life cycle stage because of the change in ploidy, or a developmental stage.

Temporal changes in descriptive data may occur cyclically within the life of individuals. These changes may be influenced by seasons (summer, winter) or synchronized biological clocks (circadian = changes over the day, circalunar = changes over the lunar month period, the latter is especially relevant to aquatic organisms depending on the tidal cycle). These changes are not normally considered developmental stages, but no term summarizing correlated descriptive changes seems to exist. Examples are seasonal habits of plants during the year or different bird songs in spring and winter (like in the Eurasian robin, *Erithacus rubecula*).

The topic of temporal change has also been discussed under “Change of object concepts through temporal development” (p. 162).

Other classifier concepts

Other concepts that exhibit similar classification or grouping properties in descriptive data are:

- Social insects such as ants, bees, termites, and wasps have morphologically differentiated individuals belonging to different castes (queen, workers, soldiers, etc.). The castes are a polymorphism between generations which cannot be treated as life cycle stages, because most individuals are sterile and die without progeny. Instead, they may be viewed as polymorphic generations. Caste differences may be caused by genetic (e. g., drones) or by non-genetic factors such as responses to nutrition during early development (e. g., worker versus queen). In contrast to seasonal dimorphism, however, these factors themselves are controlled by the behavior of the population.
- The result of an identification key may be a morphological variant (albinism, gigantism, etc.). In botany some variants that occur regularly and with considerable frequency may be recognized as infraspecific taxonomic ranks; but in zoology “variant” or “forma” ranks are no longer in use.
- A special case of variants are “photosymbiodemes” or “morphotypes” in lichens. Here a fungus forms a lichen with different algal partners (photobionts), resulting in lichen forms that are anatomically identical (isomorphic) or different (heteromorphic).
- Descriptions may be specific to geographic regions or different environmental strata (e. g., highland versus lowland). The underlying mechanisms are usually a combination of systematic responses to the environment and changes in allele frequencies of polymorphic genes.
- Descriptions may be based on living or dead material. Many characteristics can only be observed when living (e. g., in *Orbilina*, see Baral 1992).
- Descriptions may vary when based on material preserved by different methods (e. g., drying or ethanol conservation).
- A custom concept may be needed to summarize a complex mixture of other potential classifiers. A marked example for this is the complex life cycle of rust fungi (Uredinales, see also p. 157 in “Problems with specialized, context-dependent names for object parts”). Many different life cycle variants exist in rust fungi. For example, wheat rust (*Puccinia graminis*) has five different spore types (spermatia, aeciospores, urediniospores, teliospores, basidiospores), each of which must be measured and described separately. The relation between spore types and generations of the organism is complex: generations may produce multiple spore types, functioning as gametes (spermatia) or leading to the same (“repeating” or “epidemic” cycle of urediniospores) or an alternative kind of generation. Furthermore, the nuclear cycle is not directly coupled to spore types or mycelial generations. Instead of creating a new generation after “mating” (spermatia fusing with receptive hyphae), the existing mycelium will be dikaryotized (i. e., only the new nucleus divides and propagates itself through existing cells), creating a new genetic generation in the same somatic generation. As in most basidiomycetes, the creation of a single nucleus (karyogamy) occurs only much later (in the teliospores). Thus, the traditional classifier concept of “spore states” summarizes aspects of generations, nuclear cycle, function (e. g., overwintering of teliospore), and morphology. It forms a pragmatic con-

cept that cannot easily be reduced to its basic constituents without becoming impractical. Modeling rust fungus descriptions may be a useful “data challenge” to test information models.

Generalized term for sex, generation, life cycle stages, etc.

The various classifier concepts discussed above all describe why multiple classes of descriptions may exist within the most specific class defined in the primary (i. e. taxonomic) classification system. It seems advisable for a descriptive data information model to provide a generalized mechanism rather than selecting specific classification systems (sex, life cycle, etc.) because:

- The number of secondary classification systems discussed so far is relatively large and there is no reason to assume that the list is finite,
- the model would become specific to biological descriptions,
- the individual classification systems may be interrelated in complex ways as has been shown in the example of the castes of social insects or the spore stages of rust fungi.

No existing generalized term for such classifier concepts could be found. An internet search for a generalized name for at least sex, generation, and life cycle stages was unsuccessful. The following definition is therefore proposed:

Secondary classifiers = *variables used for classification systems that are independent of the primary classification system (which in biology are taxon names or non-taxonomic names like disease names). Secondary classifiers provide a naming system summarizing information about systematically repeatable descriptive variation that is independent of the primary classification. Multiple secondary classifier concepts (each with multiple state-values) may exist. Together with the primary classifier they define the scope of a description.*

Neither the proposed term “secondary classifiers” nor the definition given above are fully satisfactory. One problematic point is that the description scope (the “operational description unit”) is usually defined by practical considerations. It may involve sets of classifier values (e. g., description 1: first instar; description 2: second, third, etc, instars; description 3: penultimate instar; description 4: imago) and it may depend in very complex ways on fundamental secondary classifiers (compare the rust fungus example on p. 219). The current discussion should be seen as a first step in tackling the problem systematically. Therefore, the following list documents the **candidate terms** that were considered before choosing “secondary classifiers”:

- “Classifiers” seems to be too general without further qualification.
- “Non-taxonomic classifiers” is inappropriate; the primary class names may already be non-taxonomic, as in disease names. Also, the problem of secondary classification exists in description models outside of biology as well, and it would be advantageous to find a general term not restricted to biology. “Taxonomy” could be understood in a general way, but then the secondary classifiers would themselves form a kind of taxonomy.
- “Context” or “class context”. Usage examples: “This character in the context of a male individual...” (pers. comm. D. Hobern).
- “Variant” or “class variant”. Usage examples: “This class description is for variant of type male...” (pers. comm. T. Paterson).
- “Description classifiers” – this might perhaps be more intuitive than “secondary”, but it would include the primary taxonomic classification as well.
- “Determinants/classification determinants” – very similar to “description classifiers”.
- “Orthogonal classification system” stresses the independence of the primary and secondary classifications, but provides little other intuition.

- “Phenotypical classifiers” would be confusing, since phenotypic is usually considered an antonym to genotypic. Classifier concepts may be phenotypic (environmental sex determination), genotypic (genotypic sex determination), or ontogenetic (development stages).
- “Scope”: If the description has been split into separate parts, each part has a geographic, sex, life cycle stage, etc. scope. However, the term scope is only applicable to viewing generalized class descriptions. It is not applicable to recording object descriptions or to refer to the determinants of scoping (geography, sex status, etc.) itself. Definition of scope (CED 1992): “2. range of view, perception, or grasp; outlook.” Usage examples: “The scope of this class description is male objects”, or “... is objects from Morocco”. “This character is a scoping character”.
- “Applicability”: Usage examples: “The applicability of this class description is male objects”.
- “Sub-concepts” (Chalubert & Vignes Lebbe 2006).

Context of secondary classifier data

In descriptive information models, secondary classifier information may occur in several places:

- Defining the scope of coded or natural language descriptions;
- defining the scope of descriptive data sets (including digitized natural language descriptions or identification keys; e. g., a key may cover only larval stages of an insect group);
- specifying the scope of a result (e. g., “keyed out taxon”) in authored identification keys (e. g., a digitized identification key);
- in observation or collection databases as part of the specimen identification information (e. g., “male lobster”).

In the case of entire data sets a human-readable free-form text element (title, label) will often be sufficient. Similar to individual descriptions, however, machine-readable scope definitions will become more and more important with the increasing number of data sets to manage.

The occurrence of secondary classifiers in the identification results in identification keys and the desire to record similarly constrained information in specimen or observation databases seems interesting. Specimen or observation information models (DarwinCore: <http://wiki.tdwg.org/twiki/bin/view/DarwinCore>; CDEFD: Berendsohn & al. 1996a; Berendsohn & al. 1999; ABCD: Berendsohn 2005) often have special fields for sex, or life stage. Given the diversity of other classifier concepts listed above, a generalization seems to be desirable in these information models as well.

Another interesting point is that in specimen collections, multiple individuals having different classifier values (stage, sex, etc.) may be in or on a single collection unit (e. g., herbarium sheet). In models like CDEFD (Berendsohn & al. 1996a; Berendsohn & al. 1999) or ABCD (Berendsohn 2005) the data are, however, not necessarily bound to a management unit like the herbarium sheet. Individuals or even parts of them may be referred to as well. It is thus possible to have no classifier values, specific classifier values (e. g., “male”), or a set of values (e. g., “male, female”, indicating that it is explicitly known that the data apply to both scopes). This is very similar to the situation in class descriptions, where the scope may be unknown, restricted, or explicitly known to be general.

Classifier-related characters

A confusing aspect of classifiers is that – although the values do not contribute to the class descriptions – the existence of values or their frequency is part of the descriptive knowledge expressed in descriptive databases. For example, the frequency of males and females is a property of classes (e. g., species in social hymenoptera), and different classes may have different development or life cycle stages (e. g., reduced forms of the full heteroecious rust life cycle, or neoteny in animals). Such information may be considered separate characters distinct from the “secondary

classifier” values itself, but this seems artificial and contrary to the handling of frequency in other characters. In theory, the frequency of sexes could be calculated from descriptions that have a male/female sex classification. In practice this will not be possible, since the sampling of descriptions in a database (and of specimens in a collection) is highly non-random. Although presence/absence suffers less from sampling bias, complete and systematic bias (e. g., the database contains only adults) is not infrequent. Thus, classifier-related characters will normally have to be recorded independently from the data recorded in some kind of classifier mechanism.

Some classifier-related characters are often omitted from descriptions optimized for identification, because they are inconvenient to study (e. g., requiring observation over prolonged periods or population sampling). This is, however, no unique property of classifier-related characters. The convenience of a character for identification purposes may be separately recorded (“ratings”, compare “Authored character guidance”, p. 267, and “Reliability modifiers”, p. 213). Furthermore, some classifier-related characters are convenient, e. g., “sex status” with the states “monoclinous (having male and female organs in the same flower)” and “diclinous (in different flowers)”.

On the other hand, classifier-related characters have an influence on classifiers. If a “life cycle type” character of plants has the states “annual, biennial, perennial”, a possible life cycle stage “plant in the second year” is inapplicable for “annual”. Similarly, if “heterostyly” has the states “monostylous, bistylous, tristylous”, and a related heterostyly classifier the values “short, medium, long style”, the entire classifier would not be applicable for heterostyly = monostylous, and only the values “short” and “long style” would be applicable for heterostyly = bistylous.

Existing models of handling secondary classifiers

The special properties of sex, generation, life cycle stages are not explicitly discussed in current descriptive software or information models (e. g., DELTA, DiversityDescriptions, Lucid, NEXUS, Prometheus, Nemisys/Genisys). The special properties of secondary classifiers are, however, responsible for certain design features (especially the loose “item name” concept in DELTA) and problems in using existing models and software. The following sections discuss various ways in which secondary classifiers are handled currently.

Secondary classifiers nested within class names

Secondary classifiers like sex and life cycle stages may be considered part of the class name, i. e., nested within the taxonomic hierarchy. In applications based on the DELTA information model, the “item names” for larvae and adults of the monarch butterfly may be “*Danaus plexippus* (larvae)” and “*Danaus plexippus* (imago)”. Since “item names” are unstructured text, DELTA applications will not be able to distinguish the added classifier information from an infraspecific taxon. Chalubert & Vignes Lebbe (2006) propose to formalize this, nesting secondary classifiers like sex and stage as “sub-concepts” inside the taxonomic hierarchy. One may even desire to treat them as “pseudo-ranks” (discussed in Morris & al. 2004).

Nesting classifiers in class names allows using data inheritance mechanisms. For example, in DELTA the “variant item” mechanism may be used for sex or other classifiers. A “variant item” is a specially marked description immediately following a primary description, inheriting the information of the primary one (see p. 101 in “Inheriting data”). Normally used for infraspecific taxa or specimens, this may also help managing the information differing according to classifier values. A severe limitation of this is that the variant item mechanism is limited to a single hierarchical level, i. e., only one of specimens, infraspecific taxa, or classifier descriptions can be treated this way.

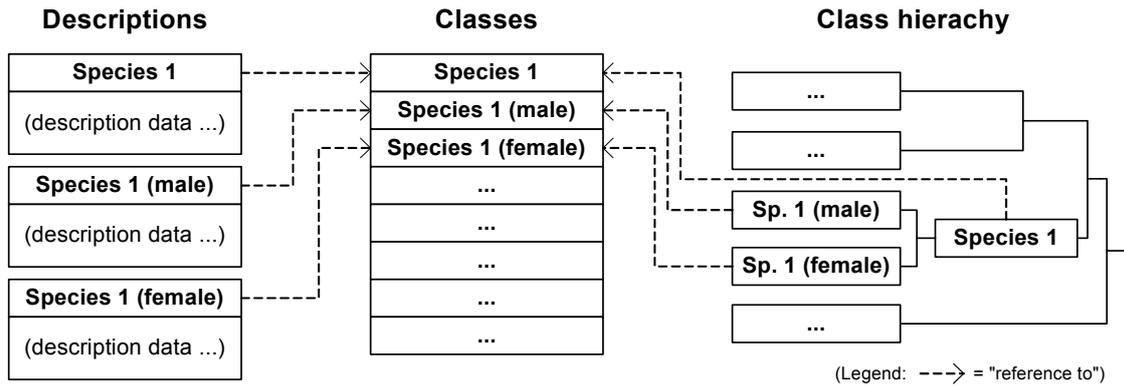


Figure 104. Treating sex similar to infraspecific taxon ranks works well on the side of descriptions, but requires to add two new “pseudo-taxa” to each taxon, both in the list of class names (which is referenced by descriptions) and in the class hierarchy.

In systems implementing description names as an unconstrained string (like DELTA), adding sex or stage information to the name is a feasible solution. However, if the class names of descriptions are handled through references to a formal list of class names (internal or external nomenclatural standard databases), this approach soon becomes highly undesirable (Fig. 104). The following major problems can be identified:

- Specifically dependent classes for sex, life cycle stages, etc. must be introduced for each identifiable class. Since objects identified to a supraspecific taxon may still be classified (e. g., for stage, like “butterfly”), this applies to supraspecific taxa, species, and infraspecific taxa.
- These additional “pseudo-classes” would also have to be added to the class hierarchy definition. Most biologists will automatically assume that “*Danaus plexippus* (larvae)” can be generalized to “*Danaus plexippus*”, but this involves a parsing of the string and semantic knowledge, allowing us to distinguish between a classifier “(larvae)” and a taxonomic author name which may be in the same position. Relationships thus have to be formally expressed.
- The taxonomic hierarchy is naturally nested. Classifiers act as separate dimensions independent of this hierarchy (Figs. 105-106). Although in general any single dimension that is independent of a hierarchy may also be viewed as nested within the hierarchy, in the presence of more than one classifier, arbitrary nesting decisions will have to be made (Fig. 107).
- In biology, the classifier extension needs multilingual support, whereas the primary scientific taxon name system is language-independent (based on Latin). However, if the primary classifier system is intended for non-scientific names (e. g., disease names), the multilingual support would already be present.

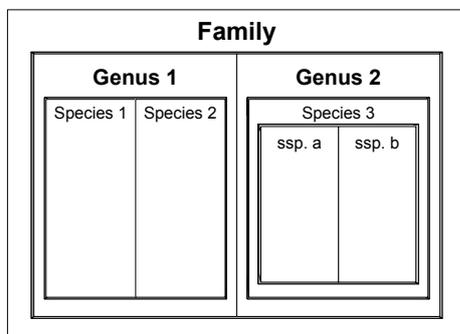


Figure 105. Visualization of the nested nature of the taxonomic hierarchy, with 2 subspecies, 3 species, and 2 genera in a family.

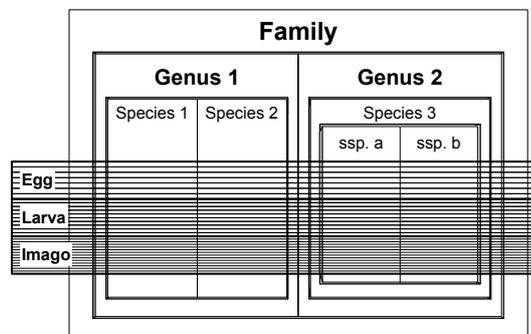


Figure 106. Developmental life cycle stages run across the taxonomic hierarchy; the stage concept does not depend on taxa (although the presence of a stage may depend on the taxon).

Furthermore, the classifier dimensions may or may not be dependent (Fig. 108):

- In humans or butterflies, sex and development stage are entirely independent.
- In the red algae *Polysiphonia* (see above), only one of the three life cycle stages (generations) is sexually differentiated. The classifier related character “Sex presence” and the dependent sex classifier are nested within stage.

Another problem is that – for report-generation purposes – classifiers may have a higher grouping priority than the primary class hierarchy. For example, caterpillar and butterfly stages may be presented in separate descriptions and identification keys (compare Table 53, p. 217). Although it is possible that software may support this, it is an operation unnatural for hierarchical arrangements and is not required for the naturally nested taxonomic hierarchy.

One possible solution would be to handle classifiers in an unconstrained string introduced in addition to the formal class name reference. This would avoid many problems noted above, but would not allow classifier-specific processing, for example, producing generalized descriptions for sex, but separate ones for stages.

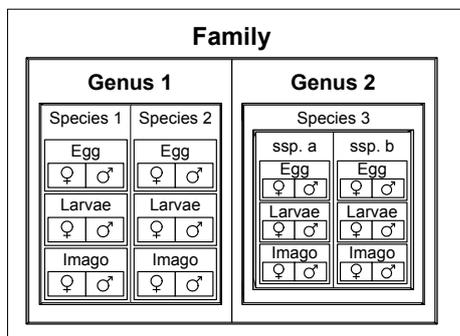


Figure 107. Sex and stage arbitrarily nested inside the taxonomic hierarchy. Males and females of different taxa or stages are assumed to have no relation or similarity.

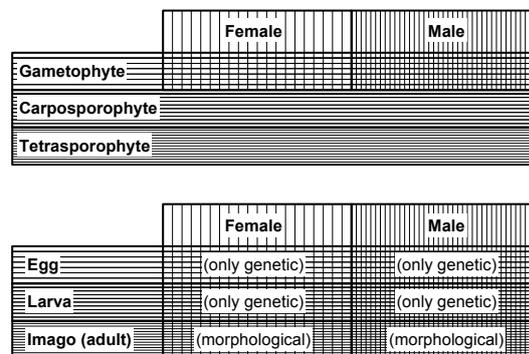


Figure 108. The conceptual dimensions of sex and life cycle stages may be dependent and nested (top; e. g., red algae) or more or less recognizable throughout all states (bottom; e. g., butterflies).

Secondary classifiers as normal characters

Secondary classifiers may be considered normal characters (like “shape” or “color”; compare the section “Classifier-related characters”, above). This approach is relatively rarely found in DELTA data sets. Prometheus (Paterson 2004) explicitly considers sex and life cycle as “qualitative description elements”, i. e., as properties of structures, equivalent to DELTA characters of type ‘UM’ or ‘OM’.

Using normal characters to express classifier information has the advantage that applications have no additional implementation requirements because existing mechanisms are used. However, the approach is problematic in that:

- Secondary classifiers are important factors when aggregating specimen data, or generalizing multiple taxon descriptions to higher taxon descriptions. If the aggregation/generalization algorithm can test which observations belong to secondary classifiers like sex or life cycle stage, it could make rule-based decisions whether to ignore sex or stage differences, or whether to create separate descriptions for them.
- The solution does not work for authored identification keys (e. g., larvae and adults of the monarch butterfly are keyed out in separate places in a single key, or in separate keys).

The first problem could be solved by defining an additional Boolean attribute on certain characters/state sets, indicating which ones are defining secondary classifiers. To solve the problem

with identification keys, one could provide a “micro-description” facility at each point where a key ends with a class reference.

Secondary classifiers modeled through character sets

Probably the most customary model in the case of life cycle stages is to provide a separate set of characters for each stage (Fig. 109). In general, this solution is preferred if the descriptions depend strongly on the classifier value, at least some characters are specific to a classifier value, and perhaps even structural differences in the object composition occur (as in caterpillar and butterfly). Usually only a limited amount of characters (overall size, DNA) are truly duplicated. However, for these duplicated characters the information to perform a generalized analysis (ignoring the classifier dependency) is lacking in currently implemented information models.

	(Characters for insect larva)										(Characters for adult insect)										
Object descr. 1	xx	xxxx	xxx	xxx	xx																
Object descr. 2											xxx	x	xx	xxx	x	x			xxx	x	x
Object descr. 3											xxx	x	x	xxx					xx		xxx
Object descr. 4	x				xx	xxx	xxxx														
Class descr. 1	xx	xxxx	xxxxxx	xxxxxxx	xxxxxxx	xxxxxxx	xxxxxxx	x	xxxxxxx	x	x	xxxxxxx	x	x	xxxxxxx	x	x	xxxxxxx	x	xxx	
...																					

Figure 109. Character × description matrix where development stages are expressed through separate sets of characters.

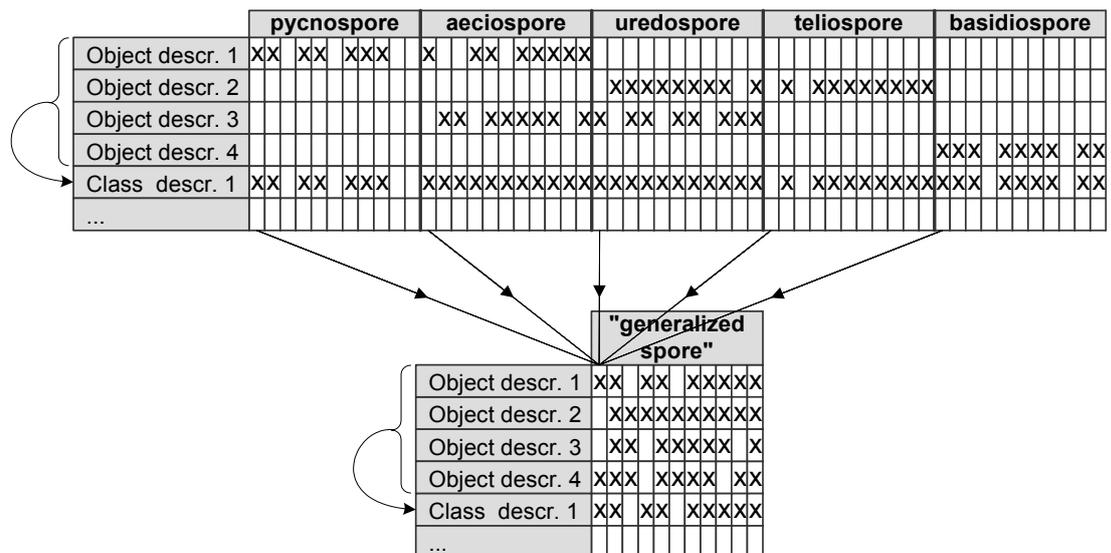


Figure 110. Character × description matrix where spore stages of rust fungi are expressed through separate sets of characters. Each set is assumed to contain the similar characters (length, width, shape, septation, wall thickness, surface ornamentation, etc.) that are specialized only through the spore stage they describe. One generalization dimension abstracts from objects to a class description. However, another desirable generalization dimension shown below the main matrix would generalize to a “generalized spore”. The arrows show the generalization only for the first character in each set. The class description in the lower matrix combines both generalizations.

In descriptions of rust fungi, almost all characters may be duplicated for the spore stages of the life cycle (Fig. 110; see also p. 219 and p. 157 for life cycle of rust fungi). A generalization as shown in Fig. 110 is highly desirable for identification purposes. Without knowledge of ontogeny and nuclear cycle, spore stages may be difficult to distinguish based on morphology alone.

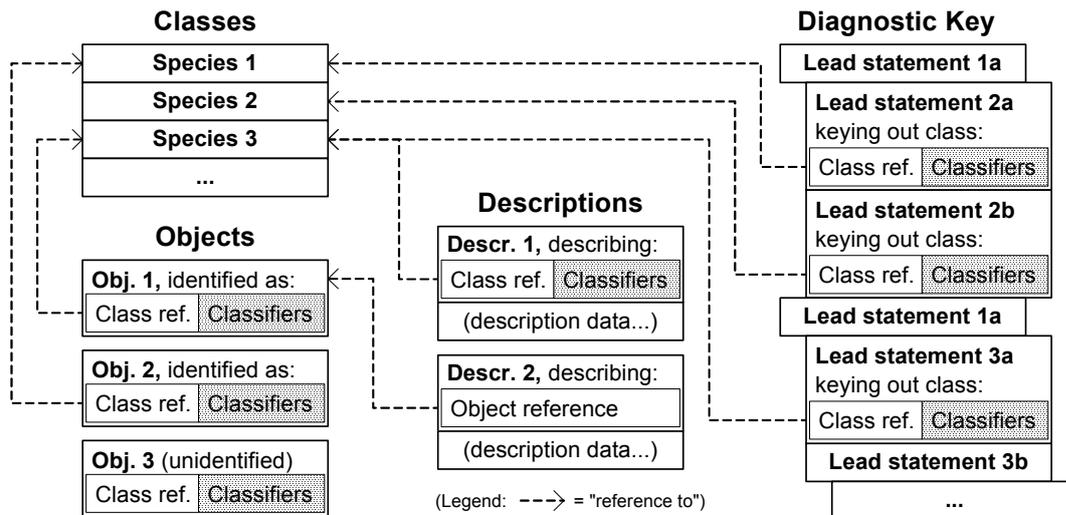


Figure 112. Visualization of objects with class (i. e. taxon) references that require an additional secondary classifier mechanism (sex, life cycle, or developmental stage).

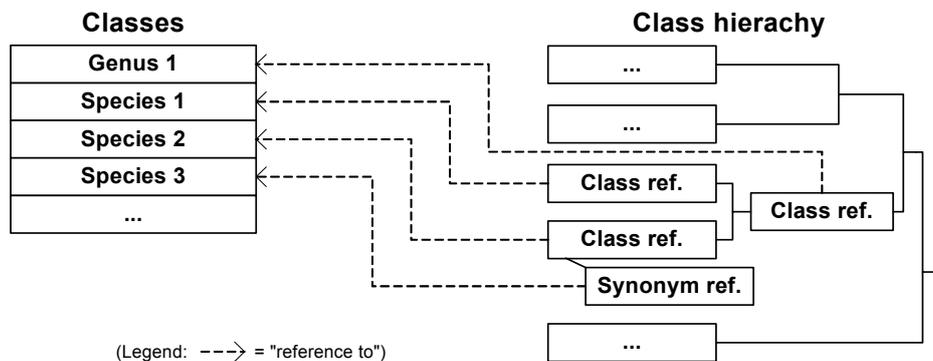


Figure 113. The class references from class hierarchy (in biology = taxonomic hierarchy) to class definitions, and from synonyms to class names do *not* require a secondary classifier mechanism (compare Fig. 112).

Summary and conclusions

This section is a first attempt to analyze the problems involved in sex, life cycle and other additional secondary classifiers used to define the scope of organism descriptions. It primarily tries to explore the problems encountered when trying to replace the free-form-text description heading or label traditionally used in descriptive information models (“item name” in DELTA) with more structured data. The term “secondary classifiers” is chosen here primarily because no better generalization has emerged. The topic would clearly benefit from work on it.

So far, the following conclusions and requirements for an information model can be summarized:

191. For biological objects, the primary classification of organisms by taxonomic name needs to be supplemented by additional concepts to describe the scope of a descriptive data set. Lacking a better concept, these may be called “secondary classifiers”. Secondary classifiers summarize a special form of correlated variation that is independent of the taxonomic classification.

192. Handling secondary classifiers as an extension of the primary taxonomic hierarchy (i. e., below infraspecific ranks) is theoretically possible, but leads to severe artifacts and is not recommended.
193. Secondary classifier concepts are not limited to sex and life cycle, seasonal, or developmental stages. Many further general concepts (castes of social insects) or highly desirable “custom” classifiers (like spore states of rusts) exists. A generalized concept is required.
194. Secondary classifiers are required in the following contexts of an information model:
 - When defining the scope of a coded or natural language description.
 - When defining the scope of an identification key.
 - When defining the scope of an identification key result (“keyed-out taxon”).
 - As part of a specimen identification information in observation or collection databases.
195. Secondary classifiers are distinct from characters. Classifier-related characters exist, but:
 - classifier-related characters can usually not be calculated based on classifier values;
 - classifier-related characters require no special handling in identification.
196. Dependency relations between secondary classifiers and characters exist:
 - Classifier-related characters may control the valid values for classifiers (see heterostyly example);
 - Classifiers may control characters (e. g., only part of the life cycle stages may have sexual differentiation).
197. The presence of some secondary classifier concepts (especially developmental stages) depends on the taxonomic group (i. e., the “primary classifier”).
198. Secondary classifier values of a given description may be unknown. A description may either be general (e. g., apply to both sexes) or the scope may be unknown (e. g., it is unknown whether the description applies to one or both sexes). Support for coding status values or a similar concept is desirable.
199. Secondary classifiers require representations for multiple audiences/languages.
200. Although the analysis was limited to biological objects, some secondary classifiers (e. g., geographical scope) may occur outside of biology as well. Thus, although the priority may be lower for non-biological applications, secondary classifiers are required for all descriptive information models.

These requirements express much structural similarity between secondary classifiers and normal characters. A future information model may attempt to develop a classifier mechanism based on the normal character system in the form of “micro-descriptions” that define the scope of a description, a key, or an identification key result. Further studies in this area are needed.

5. Identification methods

5.1. Introduction

The terms “to identify” and “identification” are used in various senses (Table 54). With respect to classes and individuals, they may be used when establishing:

1. the equivalence or identity of two class concepts,
2. a relation between an individual object and an individual name (or other identifier), and
3. a relation between an individual object and a class concept (and its class name).

Although all senses of “identification” may have a definitional rather than analytical perspective (i. e., “from now on this should be understood such...”), the term “identification” is not normally used in a definitional sense. In biology, the perspective used is always one of detecting or determining a concept that has been defined earlier (“determination” is therefore considered a synonym of “identification”). Assigning selected individuals to a class for the purpose of defining or establishing a concept is called “typification” or “vouchering”; the common term for establishing class equivalency is “synonymization”. Here, biology focuses on (re-)defining names for classes, rather than on detecting (or identifying) the equivalence of taxonomic classes. Thus the first sense of “identification” (establishing equivalence or identity of concepts) is never used in biology.

The second sense of identifying named individuals does occur in certain biological studies. However, only rarely does the individual variation of descriptive features (in the broad sense defined on p. 27) itself enable identification. Individual identification of animals is more often based on color or incision marks, coded tags, rings, or radiofrequency-ID devices (RFID) purposely added by humans. Human identification may similarly be based on descriptive data (face recognition, fingerprints, DNA profile) or purposely added identification vouchers.

Whereas in common language the second sense is dominant, in biology the third sense is the most common. The object to be identified is usually an entire organism or a representative sam-

Table 54. Examples of dictionary definitions for “identification” and “identify”.

Term	Collins English Dictionary (CED 1992)	Merriam-Webster's Collegiate / New Oxford Dictionary (EB 2001)
Identification	<ol style="list-style-type: none"> 1. the act of identifying or the state of being identified. 2. a) something that identifies a person or thing. b) (as modifier): <i>an identification card</i>. 3. <i>Psychol.</i> a) the process of recognizing specific objects as the result of remembering. b) the process by which one incorporates aspects of another person's personality. c) the transferring of a response from one situation to another because the two bear similar features. 	<ol style="list-style-type: none"> 1. the action or process of identifying someone or something or the fact of being identified: <i>each child was tagged with a number for identification, it may be impossible for relatives to make positive identifications.</i> 2. a means of proving a person's identity, especially in the form of official papers: <i>I asked to see his identification.</i> 3. a person's sense of identity with someone or something: <i>children's identification with story characters.</i> 4. the association or linking of one thing with another: <i>the growing identification of anti-slavery with political liberalism.</i>
Identify	<ol style="list-style-type: none"> 1. to prove or recognize as being a certain person or thing; determine the identity of. 2. to consider as the same or equivalent. 3. (often followed by with) to consider (oneself) as similar to another. 4. to determine the taxonomic classification of (a plant or animal). 5. (usually followed by with) <i>Psychol.</i> to engage in identification. 	<ol style="list-style-type: none"> I. (<i>with object</i>): 1. a) establish or indicate who or what (someone or something) is: <i>the judge ordered that the girl should not be identified; the contact would identify himself simply as Cobra.</i> b) recognize or distinguish (especially something considered worthy of attention): [...] <i>that the pupil's real needs are identified.</i> 2. a) (identify someone/thing with) associate someone or something closely with; regard as having strong links with: [...] <i>being identified too closely with the peace movement.</i> b) equate (someone or something) with: <i>because of my country accent, people identified me with a homely farmer's wife.</i> II. (<i>no object</i>): 1. (identify with) regard oneself as sharing the same characteristics or thinking as someone else: <i>I liked Fromm and identified with him.</i> — Origin mid 17th cent. (in the sense “treat as being identical with”): from medieval Latin <i>identificare</i>, from late Latin <i>identitas</i> + Latin <i>-ficare</i> (from <i>facere</i> ‘make’).

ple of it (e. g., a plant without the root system). Certain specialized identification processes may require only fragments of organisms (especially DNA-based techniques, but also morphological identification techniques for airborne spores or pollen, tea drugs, animal gut contents, etc.).

“Identification” may refer either to the identification process (as in “the identification has failed”) or to the resulting state of being identified (as in “the specimen has three identifications”). This duality is usually unproblematic in practice; in the following, the term identification is used primarily in the sense of “identification process”.

A special problem in the clarification of the term is the case whether testing for a simple presence/absence situation may be considered an identification or not. For example, a shipment of wheat may be tested for the quarantine pathogen *Tilletia indica* or pork may be tested for trichina cysts (*Trichinella spiralis*). Although these cases are often much easier to solve, it seems artificial to exclude them from the domain of “identifications”. Other cases with only two or a few potentially occurring species often offer similarly simplified options for identification. Furthermore, most identification use cases in ecology use keys restricted to species known from a given geographical area – which may occasionally be a single lizard, snake, etc. species occurring on an island.

5.2. Classification of identification methods

A large number of terms have been introduced to classify identification methods, sometimes with variable definitions. The lack of agreed terms and the occasionally counter-intuitive terms often impede communication. Helpful general discussions of aspects and terminology of identification methods are Pankhurst (1991) and Stevenson & al. (2003). The following section is an attempt to clarify the terminology to initiate a discussion and perhaps to agree on common terms.

Kind of data used for identification

One relatively unproblematic classification of identification methods is based on the kind of descriptive data used, leading to terms like “*molecular identification*”, “*field identification*”, etc. Probably any kind of descriptive data has been used in identification processes. Morpho-anatomical data are often most convenient, but other common examples are: Nucleic acid length polymorphism patterns (RFLP, AFLP, RAPD, etc.), nucleic acid sequences (DNA, RNA), cell wall antigen patterns using ELISA or similar immunological tests, presence and kind of enzymes (e. g., “spot tests”) in fungi, presence of secondary metabolites in fungi and lichens (Elix & al. 1988), carbohydrate cell wall composition in fungi (Prillinger & al. 1993, Schweigkofler & al. 2002), total fatty acid composition in bacteria (MIS Microbial Identification System, MIDI 2007), sound (e. g., Orthoptera identified by sound, Ingrisch & al. 2001, Palm & Dietrich 2001).

On an abstract level, this classification is relevant only insofar as different data require different data types (see p. 49) with corresponding similarity metrics and identity functions.

Levels of interaction

Another classification is based on the different degrees of interaction between human users and identification methods. Humans may be involved in the identification process as:

- initiator, requesting an identification,
- operator, performing standardized routine operations,
- expert, adding expert knowledge or reasoning not embedded in the key itself,
- teacher, explaining identification concepts.

In principle, any form of involvement may be considered an “interaction”, but it seems desirable to restrict this term to the cases where the interaction involves human knowledge and reasoning.

In this sense one can define:

- **Automatic identification** as a process that can be performed by a machine (with humans only initiating processes and performing standardized operations). Examples are DNA sequencing, bar-coding (e. g., Cowan & al. 2006) or DNA microarray methods (e. g., Loy & al. 2002, Leinberger & al. 2005), many image processing methods like human iris or face recognition, leaf outlines (Agarwal & al. 2006), recognition of spores (e. g., Chesmore & al. 2003), shell fish larvae (Tiwari & Gallager 2003), hymenopteral wings (ABIS method, Steinhage & al. 2001), and spiders (Do & al. 1999), or the fatty-acid-based MIS Microbial Identification System (MIDI 2007). Further examples and a discussion of the opportunities and obstacles to automated identification may be found in Gaston & O'Neill (2004). Dreams for the future include handheld devices usable by the general public, performing automated molecular identifications (Janzen 2004).
- **Interactive identification** as a process where human knowledge and reasoning is supported by a knowledge base, whether or not a computer is involved in processes like sorting, lookup, and elimination of options or not. Note that, in contrast to this definition (which includes the majority of humans using a printed key), the term has been widely used to exclusively refer to computer-aided identification using multi-access keys; details of this are discussed in the appendix (p. 394).

This distinction between automatic and interactive identification is not always unambiguous. Automatic identification methods often require manual preparation and processing of objects which, although in principle standardized, may involve some complex choices to be made. Conversely, computer-aided interactive identification may occasionally require only fairly standardized operations by humans, e. g., comparing an object with colors or shapes displayed on a computer screen. One may say that a truly interactive identification occurs only if it exploits a substantial amount of human knowledge and experience (e. g., of terminology, classification, or frequency of object occurrence). A typical case of such interactive identification is the informed choice of characters in a multi-access key.

Identification may be semi-automatic in the sense that a computer-aided system may present a choice of potential results based on fully automated methods (e. g., image processing), optionally together with an estimate for the likelihood of correct identification, but then requires a human confirmation of the result.

Phases of interactive identification

Most biological interactive identification processes, whether supported by a computer or not, may be divided into the following phases:

- **Orientation phase.** Initially, the appropriate resources (e. g., identification keys, descriptions, images) and observation methods (hand-lens, microscope, chemical reagents, etc.) for the object at hand are selected. The choice of the right key may depend on many factors beyond the recognition of a broad taxonomic group: e. g., expertise of identifier, geographic location, season (summer, winter). In publications addressing experts, the orientation phase may be considered implicit, but publications for the general public or students will often explicitly discuss the choice of methods and often provide entry keys (e. g., to families or orders). A fundamental part of the identification process will almost always be based on common knowledge. However, the extent of what is considered “fundamental” (recognition of something as an insect versus as a member of the moth family Geometridae) is audience-dependent.
- **Identification phase,** consisting of one or both of:
 - **Key phase.** Using some form of identification key (e. g., a dichotomous or multi-access key), the object is identified until only one, or a few taxa remain. Typically, only a subset of the potential descriptive characters will be used during this phase. Well-designed keys may come close to a binary search algorithm, which is very effective for a large number of species. In authored keys, e. g., dichotomous keys, the experience of generations of scien-

tists is provided, leading to a selection of effective, convenient, and reliable characters. The phase may be skipped if previous experience already results in a sufficiently small set of taxa to warrant starting with a browsing phase.

- **Browsing phase** (“*scanning*” in Stevenson & al. 2003). Short descriptions or illustrations of organisms are compared with the object to find the correct class name. The phase is skipped if the key phase ends with a single identification. The number of taxa after which a browsing phase is considered more profitable than continuing the key phase will depend on the difficulty of the key, and may differ between professionals and amateurs.
- **Confirmation (or verification) phase.** The object is thoroughly compared with the information available for the identified class (a free-form natural language description, a tabular character synopsis, or images, audio tracks, videos, etc.). The use of a wide range of characters (in contrast to selected ones during the key phase) helps to avoid errors due to oversight, bad material, misunderstanding of a character, or bad key design. Although probably most biologists have learned to distrust their own initial identification results, this phase is often overlooked in formal treatments of identification.

Although much of the discussion about identification centers on the *key phase*, keeping all phases in perspective is important when developing use case scenarios.

The relevance of the phases depends on the experience of the person performing the identification. Experienced identifiers for a given taxonomic group may not recognize an orientation phase at all. The key phase may be skipped if a sufficiently small taxonomic group is suspected, warranting to start with a browsing phase. The suspected group may be as small as a single species (the object is “recognized” or “known”, Pankhurst 1993a). Finally, depending on the estimated probability of recognition, even the confirmation phase may be skipped.

This classification of the entire identification process offers important insight into how well-designed identification tools (be it printed or computer-aided) should be constructed. Material relating to the individual phases should be clearly and visibly marked, supporting the user to tailor the identification process to her or his needs. The material should be arranged such that subsequent phases are arranged close to each other for fast access.

The assumption that identification in popular plant or bird identification books (“field guides”) immediately starts with a browsing phase is somewhat erroneous. On closer inspection the user is usually guided by grouping the material into intuitive categories, often two or three levels deep (Stevenson & al. 2003). The result is equivalent to a short key phase. Even where no explicit “group keys” exists, some form of coarse classification (taxonomic or artificial, e. g., “flower color”) is usually embedded in the design of the book and occasionally a “key” is incorporated into the table of contents. Identifying a “bird” truly by browsing is not very practical, identifying a bird of prey, a dabbling duck, or a garden song bird by browsing is.

A combination of key and browsing phase occurs if the key phase ended with multiple potential results. Reasons for the latter are a) the key itself may be unable to distinguish taxa (perhaps a genus key is present, but no key was created for a genus with three species), b) the variability of the objects may be higher than expected in the key, leading to contradictions and uncertainty, c) some key questions may be not answerable because features are missing in the object at hand, or d) the user may have problems in interpreting terminology used in the key (but not defined).

If a browsing phase is present and printed descriptions and illustrations are used, the distinction between browsing and confirmation becomes very weak, the confirmation phase and the end of the browsing phase are usually identical. However, when computer-aided identification tools are used, the confirmation phase after a browsing phase may use sophisticated methods, such as:

- determining similar taxa algorithmically based on available coded descriptions,
- presenting a list of similar taxa based either on manually authored lists of “easily confused taxa” (used, e. g., by AditKey, p. 21, and EFG, R. Morris, pers. comm.), followed by a
- secondary identification phase, which in turn may be
 - a simple browsing phase,

- a key phase using a branching (dichotomous/polytomous) or multi-access key that is being generated dynamically based on diagnostic characters that have not yet been answered (“check key”, Payne & Preece 1977).

Further aspects of the confirmation phase are discussed in the form of use cases; see “Confirmation of identification”, p. 309.

Structural classification of identification keys

In general, an identification key is a device to accelerate the process of comparing a given object with all available object or class descriptions. Keys are closely related to indices in databases. This section tries to highlight essential differences of keys, especially where the difference is relevant to information models.

The most common form of keys in biology uses a process in which the user of the key compares the material to be identified with a group of logical propositions that evaluate to either true or false. The true proposition is accepted and leads to the next step in the identification process. Essentially, all propositions in a group must be mutually exclusive and exhaustive (i. e., no situation that is not covered by one of the propositions should occur at this point in the key).

Propositions may involve categorical as well as quantitative data. In printed keys, but also in many computer-aided keys, users compare their measured values against fixed limits (e. g., “ ≥ 12 ”) defined in a proposition instead of entering values (leg number, leaf length) directly. Computer-aided keys may also support alternative methods, where quantitative values are directly entered and then algorithmically compared with values in databases. The matching process used to do so may involve fixed error margins or multivariate statistical methods.

Propositions are found in two styles:

- In the **question/answer style** each proposition is the combination of a question and one of multiple answers. Although this is the most common style in general questionnaires, it is relatively rare in printed or computer-aided identification keys in biology. Example:

How many pairs of legs are present?

- 9-10
- ≥ 12

- In the **lead style**, paired propositions are presented, one of which should evaluate to true. The lead style implicitly starts each step with the question “which of the following statements is true?”. This style is the most common in conventional, printed identification keys. Example:

- 9-10 leg pairs, antennae short
- ≥ 12 leg pairs, antennae long

In the question/answer style the elements of an identification step are kept together by the question. In the lead style, commonly other means (formatting, numbering, etc.) are used. The set of propositions that must be evaluated in a single step is commonly called a “couplet”. Although primarily used for lead style keys, in the following the term is used for both question/answer style and lead style.

Question/answer style and lead style differ in their support of combinations of multiple characters in a single step (which is often desirable to create a set of mutually exclusive and exhaustive propositions). The question/answer style is usually limited to a single character (possibly with multiple states in a single proposition), whereas lead style often include multiple characters combined with Boolean operators such as ‘and’, ‘or’, or ‘not’ (‘and’ often being implied and represented by a comma). In principle, question/answer style may also include combinations of characters, but – as can be seen in the following example – the result is more difficult to interpret:

How many leaflets are on each leaf, and what is the shape of the leaflets?

- 1-5 and round
- 5-15 and not round

In addition, combinations which involve ‘or’ operators between characters or which leave information unspecified in some of the propositions (compare Fig. 114, couplet 5: the length of the tail is not mentioned in the first lead, implying that it is variable or irrelevant) are often either impossible to create or will be highly confusing in question/answer style.

Identification keys like the one described may be understood as a special form of structured diagnostic description (p. 39), omitting information that is redundant because of an earlier couplet. As a result, it is in principle possible to extract diagnostic descriptions from a key – but the result may not be very useful if many Boolean combinations of characters are present in the lead statements.

The classical key in biology is a **dichotomous key** (e.g., Figs. 114, 115), where the sequence of couplets is fixed and each couplet contains exactly two propositions (“leads”). A relatively frequent variant of the dichotomous key supports more than two leads per couplet. The recommended term for such a key is **polytomous key**. The term “*polychotomous*” is also found, albeit erroneously formed (*dichotomy* based on “*dicho-*”, Greek “*dikho-*” “in two”, “apart”, and “*-tomy*”, Greek *tomos*, a “slice”, “cutting”, “section”).

Whether a key is dichotomous or polytomous is often irrelevant. In many treatments both types are simply called “key” or “diagnostic key”, which, however, also include keys where the sequence of questions is freely selectable by the user (multi-access keys, discussed further below). A specific and generally agreed term for the generalization of dichotomous and polytomous keys is lacking; the term “**branching key**” is chosen here (discussed in detail in the appendix on p. 396).

It is often useful to provide alternative paths to a taxon in a branching key. Some characters in the identification path may be only seasonally applicable, others may be very useful when diagnosing some species, but difficult to assess or highly variable in other species. Furthermore, the tolerance of a key against common errors may be improved by such a design. Alternative paths may either be realized by inserting result leads (typically taxon names) multiple times, or by providing a redirection to an earlier couplet, essentially cross-linking the nodes in the couplet-tree (Fig. 116). The resulting graph still maintains a direction and no circularity and is called by mathematicians a **directed acyclic graph** (DAG, Stevenson & al. 2003). Knowledge whether branching keys are trees or DAGs is especially important when analyzing branching keys and when designing an information model to store such keys. Osborne (1963) calls this a “reticulating” key, because it creates a “net” instead of a tree. Osborne develops a mathematical framework to analyze the influence of reticulation on dichotomous keys for equal probability of answering couplets correctly, admitting that the analysis is misleading insofar as reticulation will in practice be focused on cases where a high probability of wrong answers is experienced. A continuation of this work, i.e., a framework for studying key optimality (with and without reticulation) based on given probability estimates for answering individual couplets correctly is not known to the present author.

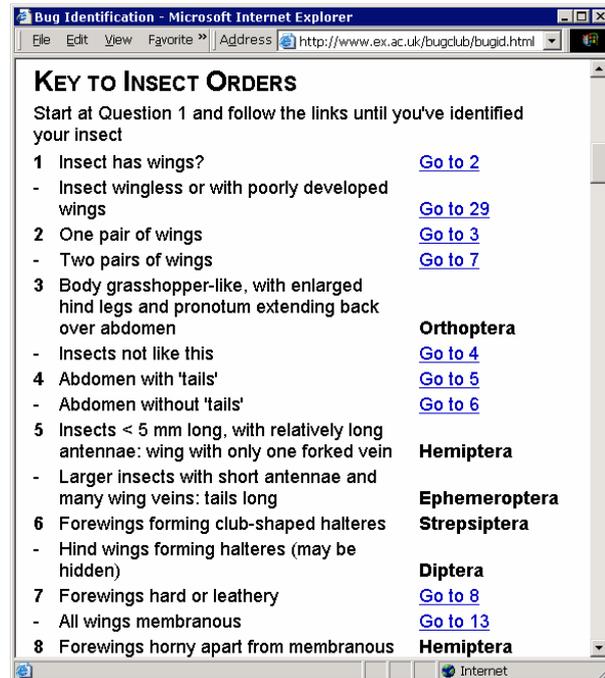


Figure 114. An example for a printable, hyperlinked dichotomous key (<http://www.ex.ac.uk/bugclub/bugid.html>).

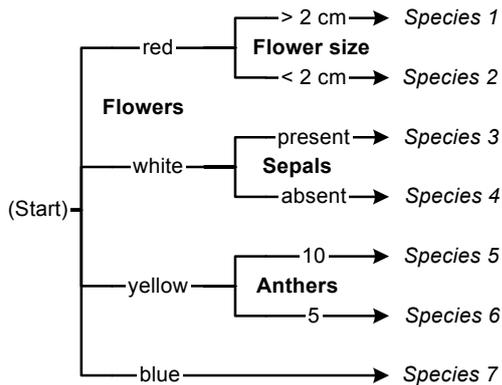


Figure 115. Principle of a branching key. The sequence in which questions must be answered is fixed. In the example, the first question is polytomous (having more than two alternatives), the following questions are dichotomous.

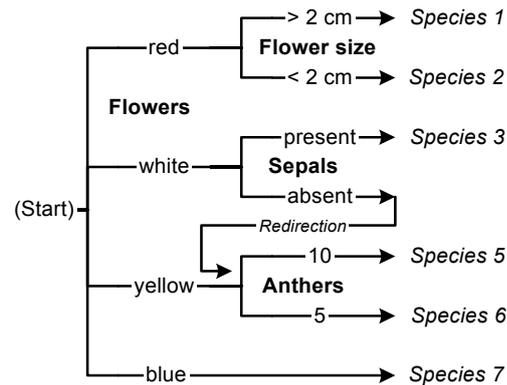


Figure 116. A variant class of branching keys includes redirections (or “reticulations”), resulting in a directed acyclic graph. Species 5 is assumed to have light yellow flowers that occasionally are almost white.

A special feature in many branching keys is the desire to label an inner node of the key as a result, but then continue with the key structure. This situation is termed a “**result-and-continue pattern**” here. It occurs whenever a taxonomic rank has been clearly identified, but the author of the key desires not to create a separate subkey (which would be common when reaching family or genus level), but to continue with the same key (common when reaching a subfamily, a tribe, or species with infraspecific taxa). Adding the taxon result as free-form text at the end of the lead text (perhaps formatted in bold) minimally fulfills this requirement. However, it will often be desirable to format the taxon name identical with other results (especially for species), which creates a requirement for a special data element. An alternative way to support this pattern would be to consider each such point creating a new subkey, and provide metadata at the subkey result pointer to request an “embedding” of the subkey. However, it is not always desirable to create separate subkeys for each species with infraspecific taxa.

The main alternatives to branching keys are keys where the user may freely choose the questions to be answered (Fig. 117), essentially telling his own story rather than being examined. Similarly to branching keys, no generally accepted and established term for these keys exists. Many terms have been proposed and at least one is confused and should be avoided (*synoptic key*, see p. 398). For the purpose of this thesis, following the review by Edwards & Morse (1995), the term “**multi-access key**” is preferred. Again, the alternative terms are discussed in the appendix (p. 397). A multi-access key has several advantages:

- It allows the user to ignore any question considered undesirable. In a branching key the failure to answer the next question (e. g., because of interpretation problems, because the object part is missing, or because the feature is not expressed at the time when the object was collected) compels the user to follow all leads of the couplet. Determining which leads are dead ends and which lead to a successful identification is, in practice, very time-consuming and error-prone.
- It allows the user to select the characters that are most conveniently observed in a given object. Although a branching key usually prefers convenient characters as well, while appropriate for the majority of taxa in the key, the selection may not be convenient for the current taxon. In a multi-access key the user is able to react to this.
- In a computer-aided multi-access key, quantitative characters can be employed directly (entered as measurements), rather than requiring a previous fixed categorization (e. g., into “ ≥ 2 cm”, “ > 2 cm and < 5 cm”, “ ≥ 5 cm”).
- Experienced users can apply previous knowledge to accelerate the identification progress. Even if they do not know the taxon name, they may often know that a particular character

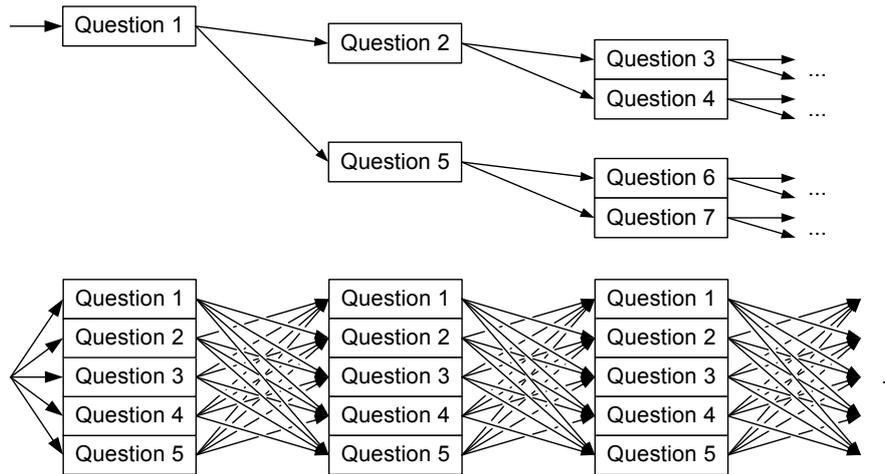


Figure 117. Visualization of possible user interaction steps in a branching key (top, the steps follow the data structure) and a multi-access key (bottom, the sequence is determined by the user). In multi-access keys, the interaction is essentially recursive (the user interface may or may not prevent directly revising previously answered questions).

state is rare or even unique in a group. By starting with this, an identification that would require dozens of steps in a branching key may be finished after two or three steps.

The choice between multiple questions at each step is the defining feature of multi-access keys. Without this being essential, most multi-access keys limit each question (i. e. couplet) to a single character, avoiding characters combined with ‘and’, ‘or’, or ‘not’. Most multi-access keys further make liberal use of more than two alternative answers; mostly one lead for every state of a character. Where too many states would result in a confusingly high number of leads, a lead may combine multiple states with ‘and’, ‘or’, or ‘not’ (either during key construction, or already embedded in the terminology).

The preference for multi-character couplets and dichotomy in branching keys and the preference for single character couplets and multiple leads in multi-access keys are interrelated. In branching keys both the desire to create mutually exclusive and exhaustive alternatives, and the need to make the key practical by offering alternatives to characters that are not always observable, often leads to multi-character couplets. However, evaluating three or more complex, multi-character statements quickly becomes a challenge in Boolean logic, leading to a preference for dichotomous keys. In contrast, in multi-access keys, the tolerance of these keys to non-exclusive leads and to missing information on specific characters removes the need for combining characters. On the contrary, combining characters usually makes it more difficult for users to select the next couplet. In the context of single-character couplets, limiting the choices to two leads becomes artificial.

The use of single or multiple characters in a key couplet is occasionally called a *monothetic* or *polythetic* key. In the Aristotelian sense *monothetic* or *polythetic* expresses whether class membership can be identified through an unambiguous combination of characteristics, i. e., whether a single set of necessary and sufficient conditions exists (monothetic) or not (polythetic). In a polythetic classification, members of a class have variable characteristics, and no single set of differential characteristics exists (Radford & al. 1974). A classification based on multiple characters may be either monothetic or polythetic. Polythetic taxon delimitations are indeed one reason why a key may have to use multiple characters, but – as discussed above – multiple other reasons exist. And conversely, polythetic taxa may alternatively be identified using single-character couplets, where taxa are keyed out in multiple places. *Monothetic* or *polythetic* should therefore not be applied when describing the structure of keys.

The most frequent style of a multi-access key is that each couplet is labeled by a noun-clause expressing the organism parts and property (e. g., “leaf shape”, “fore-wing spot number”). A more verbose style involving question syntax (e. g., “what is the leaf shape?”, “how many spots occur on each fore-wing?”) is occasionally found and may be termed “question-answer style”. This requires additional character metadata.

Identification keys may contain propositions that relate to circumstances or conditions of the observation process rather than to descriptive data of the organism that is to be identified. For example, a question in a printed key might be “material with/without flowers”. Where it is unambiguous this may even be abbreviated (somewhat confusingly) to “flowers present/flowers absent”, referring not to the fact that a species never has flowers, but that it is not in a flowering state. Similarly, the identification of microorganisms often depends on whether they are collected under natural conditions, or whether they have been cultivated under laboratory conditions. Such conditions often greatly influence the description.

Interestingly, whereas (authored) branching keys often include propositions depending on the circumstances of identification, multi-access keys, especially computer-aided ones, currently usually do not. The reason for this is that no current information model for descriptive data includes the necessary information; see “Dependencies on circumstances of identification” (p. 175).

Identification tools aiming at identification by browsing (especially field guides) may be considered another structural form of a key. However, as argued above (p. 231), these keys are usually organized into a sequence of categories, followed by descriptions or illustrations intended for browsing (and arranged taxonomically or alphabetically). The structure is thus identical to a branching key, where the number of leads in the browsing phase is usually much higher than in a “normal” polytomous key.

To summarize, the most relevant criteria for a structural classification of keys are:

- whether the couplets must be answered in a fixed sequence defined by the key authors, or whether the sequence is freely selectable by the user (*branching* versus *multi-access key*);
 - in the case of a branching key: whether each taxon is keyed out only once, whether a taxon may be keyed out in multiple places, or whether it supports redirections back into different branches of the key (“reticulated key”);
- whether the propositions (i. e. leads) in a couplet are limited to two alternatives or not (*dichotomous* versus *polytomous key*);
- whether each couplet is limited to a single character, or whether it may be a combination of multiple characters
 - in the case of multi-character propositions: whether Boolean operators such as ‘*and*’, ‘*or*’, or ‘*not*’ may be used; and
- whether couplets are a list of complete statements, or split into a question and answer parts (this may occur both in branching and multi-access keys);
- whether the key, in addition to descriptive data, supports the selection of available observation conditions, instrumentation or methods.

Both branching and multi-access keys can be presented in various formats or styles, that depend to a large degree on the presentation medium (printed or computer). These “Presentation styles of identification keys” are discussed in a later section (p. 242).

In a small comparative study, Morse & al. (1996) found a small advantage of multi-access keys over (printed and hyperlinked) branching keys with respect to the accuracy of identifications (not statistically confirmed). At the same time, the use of multi-access keys required substantially more time. The latter result may have been due to the fact that in the study the branching keys included illustrations, whereas these had to be looked up separately in print when using the computer-aided multi-access key. This topic requires further studies in the future. It is likely that the relative strength of branching and multi-access keys strongly depends on the number of taxa in a key, the difficulty in finding a consistent set of reliable and easily observable characters, the vari-

ability with which necessary characters can be observed, and the experience of the user. Due to the interactive properties of multi-access keys, these have a higher potential to become faster with increased experience of the user than branching keys. Given that both branching and multi-access keys seem to have advantages, the information model should support both types.

Propositional versus object matching metaphors

Two dominant metaphors exist for interactive identification processes:

- **Propositional or predicate logic:** The process is based on testing propositions (or “assertions”) that an object (or object part) has a property value or composition (e. g., “plant flower color is blue”). Multiple propositions may be combined with ‘*and*’, ‘*or*’, or ‘*not*’, resulting in a set of conditions that the object must fulfill. Although predicate and propositional logic are complex philosophical and mathematical topics, in their simple form they are widely understood and intuitively employed. They are used in high school mathematics and many interactions with computers (most programming and database query languages are based on predicate logic). Propositional logic is widely used in biology in branching keys. Many interactive multi-access keys also explicitly use propositional logic, e. g., the Identify function in DiversityDescriptions uses it to record the history of identification steps.
- **Object matching:** The object to be identified is described and the resulting description is compared with descriptions in a knowledge base. The knowledge base may simply be printed text or illustrations, or it may be queried using a software application. For example, the CSIRO Intkey program (Dallwitz & al. 2000b) explains identification as a matching process. In the Pankey package (p. 19) the program “Match” is based on this view.

These metaphors primarily influence the communication with the human user, but not the actual algorithms for identification. This can be seen, e. g., in current commercial database management systems (DBMS). Commonly, propositional query languages like SQL are used, in which conditions in the “where” and “from”-clauses must be true and are explicitly combined using Boolean operators. However, in the human user interface, questionnaire-like query-by-example interfaces (“qbe”, a form is filled out and matching objects are returned) may be used that correspond to an object matching metaphor. Most current database systems supporting qbe simply convert the data in the qbe form into the native, propositional query language of the DBMS. One may note that in some aspects the language used to explain the process to the user differs. For example, the use of ‘*and*’ and ‘*or*’ is reversed: “leaves (have to be) ovate *or* lanceolate” in predicate logic may become “some leaves (are currently) ovate *and* some leaves (are currently) lanceolate” in qbe logic.

Intuitively, one may assume that *matching* methods are more error-tolerant than identification methods using *propositional* language. The convertibility shows, however, that both methods may be designed to be more or less error-tolerant (see section “Equality criteria and error tolerance”, p. 264).

“Promorph” and “looks like” metaphors

Instead of relying on analytical characters, a special form of “assisted object matching” relies on the human intuition for “similar” patterns or forms. This approach has been termed the “promorph method” by Fortuner (1989, 1993) and the “looks like method” in the Electronic Field Guide project (EFG, p. 20). Fortuner defines a promorph as “a form that can be recognized before detailed study of its morphology” (Fortuner 1989). Promorphs, although typically supported by images, may be given names to be able to refer to the concept in written text (see “Tiger”, “Clearwing” in Fig. 118).

In these methods images (photographs or generalized drawings) are used already at a high level in the key, representing a group of similar species rather than individual species. The user of a key is confronted with a set of images (e. g., of butterfly wing patterns, Fig. 118) and chooses

the one considered to be most similar to the object. By creating several such “test panels” and by studying human similarity estimates in known test cases, it is possible to restrict the scope of objects returned by a query. This identification model utilizes the unconscious human pattern similarity recognition and is thus a completely different kind of “matching method”. It can be fast, intuitive, and requires minimal or no knowledge of terminology. On the other hand, it is not strictly analytical. The typical users may classify a species under multiple promorphs, and some species may not fit into any larger group of promorph similarity (requiring either to add a specialized, ineffective promorph, or some method to communicate that “other” species exist as well).

Promorph or “looks like” images typically guide the user to a set of species, narrowing the choices, so that the following steps in the identification (using diagnostic characters or images, including images highlighting diagnostically significant features, compare Fig. 119, p. 241) become more efficient.

The reliance on subconscious similarity estimates requires extensive testing of human similarity estimates for the objects to be identified. Similarity estimates may be culture-dependent and strongly depend on the expertise of the observer. An expert will include the known diagnostic features subconsciously into the similarity estimate, essentially weighting the general similarity estimates by recognizing parts that an inexperienced observer would probably ignore. This can be compensated by an appropriate choice of features displayed in the promorph image, but it makes the promorph images somewhat dependent on the set of promorphs that is displayed to the user.

Looks-like and promorph-guided identification is probably a highly efficient way of identification by humans. However, the required extensive testing of all potential user groups of a key is time consuming and expensive, especially if the number of potential identification results (species richness in the scope) is large. For taxonomic groups with low commercial impact, these costs will often be prohibitive.

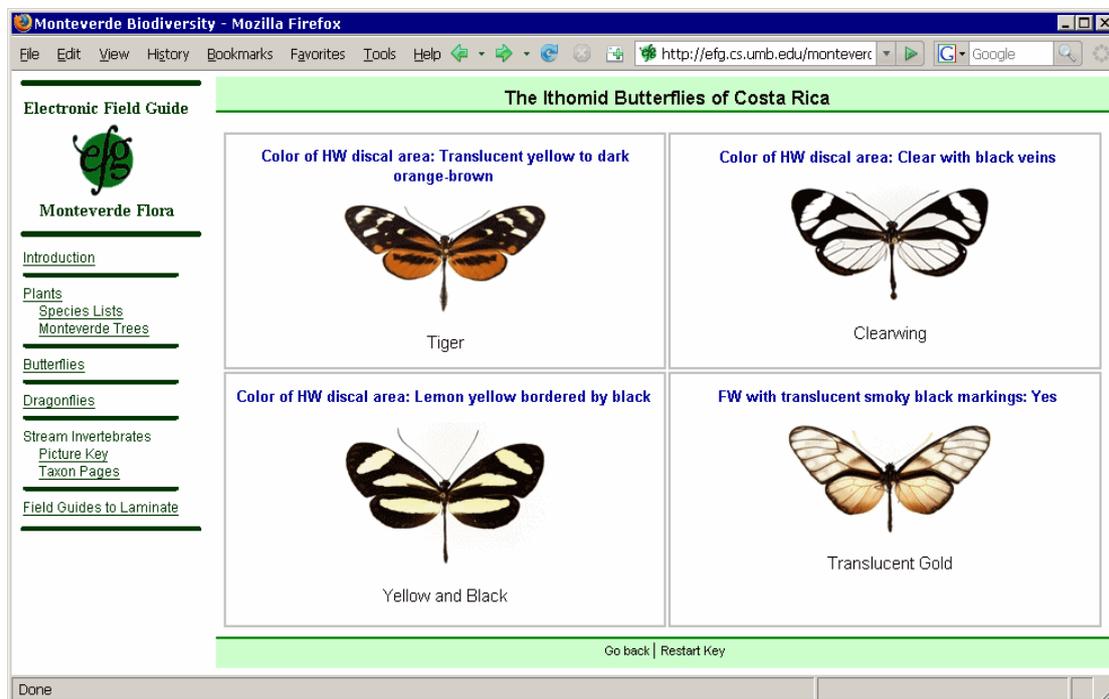


Figure 118. An example of a computer-aided polytomous key, split into one question per page (<http://efg.cs.umb.edu/monteverde>). Key statements and image captions are independent; the latter are used to provide mnemonics to the images.

Radford's classification

Radford & al. (1974) classify identification methods into:

- expert determination,
- recognition,
- comparison, and
- use of keys.

Expert determination seems to be on a different level, focusing on the quality of the identification result rather on the process. An expert identification may be based on recognition (i. e., comparison with memory, often subconsciously), detailed comparison with known objects (directly, or using illustrations or descriptions), and use of keys (i. e., comparison with optimized diagnostic descriptions).

Other classification criteria for identification keys

In addition to the criteria of content, structural classification, and interactivity discussed so far, some secondary classifications criteria are commonly used.

A major distinction occurs often between *field guides* and *expert keys*. Stevenson & al. (2003) show that the perception of what a “field guide” is, is strongly determined by market and publishing constraints, leading to browsing guides showing mostly the entire organism, tailored for a specific area, focusing on large and often colorful organisms that are abundant and easy to study. In practice, the balance between the market interest and the number of organisms in a taxonomic or ecological group and the geographic area corresponding to the market will determine, whether a field guide is reasonably complete (common for birds, mammals, amphibians, dragonflies, butterflies, trees, etc.) or whether the identification quality is compromised by ignoring a large proportion of the less frequently occurring or less showy taxa (common, e. g., for most plant, fungi, or insect groups).

With increased availability of digital software and hardware, the publishing constraints can be lessened, allowing for unlimited support of color photographs even for taxa with a smaller market impact. Computer-aided identification tools allow for better integration of elimination approaches (e. g., key-based) with browsing approaches, and for improving the relation between the browsing and the confirmation phase (by reusing the same material, but tailoring the amount of detail shown to the specific phase). Thus the term “field guide” should best be used in its original sense, as a guide optimized for quick identification of organisms immediately during observation or collection. It is always desirable to reduce the amount of technical language used to minimum, but whether the lowest level of expertise currently achievable for a taxonomic group makes a field guide easily usable by general public or not, should not determine its status. Similarly, the relative amount of analytical keys (allowing a process of elimination) and browsable illustrations should not be an *a-priori* condition of “field guides”, but determined by the number of taxa required to be distinguished. Picture browsing works best with perhaps up to 50 taxa in a group determined by means of elimination.

Other classification questions that may be relevant for both printed and computer-aided keys are:

- Is the key the result of a design process (*authored key*, containing information possibly not available elsewhere) or is it algorithmically created based on available data? Both branching and multi-access keys may be *authored* or *algorithmically created* (see also the use case diagram Fig. 202, p. 312). The sequence and selection of characters in a branching key may contain distilled experience of generations of researchers.
- Which combination of text and media resources (especially photos and drawings, called “*multimedia taxonomic keys*” in Morris & al. 2007) is used in the key? A key may consist entirely of images, entirely of text, of images with caption text, or of text with illustrations.

- If text and media resources are combined, the latter may be directly integrated into the key structure (in-place), or linked through reference numbers or hyperlinks. In the latter case, the resources may be available on the same page or screen (in-view) or elsewhere (look-up, in computer-aided keys especially in the form of pop-up windows).
- Do media resources represent the entire organism or are they analytical and specific to identification details relevant to decisions in the key? The latter criterion may be applied to photos and drawings (which may be appropriately cropped, or display area-of-interest boxes or arrows, e.g., Fig. 119), but also to sound, video, or 3-dimensional voxel pictures.

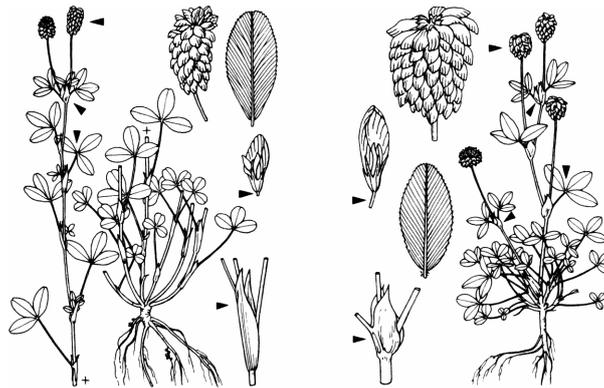


Figure 119. Example of analytical illustrations, detailing diagnostically relevant parts of the organism (left *Trifolium spadicum*, right *T. badium*); regions of interest are highlighted by arrows (after Rothmaler & al. 1985).

Some further criteria are only relevant for computer-aided keys:

- Handling of quantitative data:
 - Is it possible to directly enter quantitative values and compare these algorithmically with values in databases?
 - Are error-tolerant comparison methods supported or simple value comparisons?
 - Are multivariate statistical methods supported?
- Error tolerance (compare p. 264):
 - Is the identification error-tolerant, i. e.: is the key able to suggest taxa that are close but inexact matches?
 - Are contradictions silently accepted or is the user informed which data are in contradiction with the result?
- Guidance in character selection (multi-access keys only; see p. 267 for further information):
 - Is character guidance authored or algorithmically calculated based on coded descriptive data?
 - Can the list of characters be sorted such that recommended characters appear first?
 - Does the character recommendation algorithm work for quantitative and categorical, or only for categorical characters?
 - Are redundant characters (those that no longer contribute to the identification progress) marked in some way? Are they removed from the list of available characters and thus no longer available to confirm or contradict other data in the identification process?
 - Does the guidance adapt to identification progress, i. e., is it based on the remaining taxa, or is it always based on the set of all taxa in the key?
 - Are character applicability rules (“character dependency”, see p. 76) observed? Are inapplicable characters marked or completely removed? Are controlling characters implicitly scored if a dependent character is scored?
- History:
 - Is a history of “identification steps” or “information entered so far” available?

- How is this arranged (sequence of scoring, alphabetical, by concept, by part, etc.)? Can the user choose between different arrangements?
 - Is it possible to revert (delete) or update (change) a previous identification step?
 - Is the key *adaptive*, i. e., does it change its structure based on previous information? Digitized branching keys might be simply hyperlinked, or they might hide/fold parts of the key as they become irrelevant. However, a branching key that is split into one web page per couplet would be indistinguishable from a system that is adaptive by other means.
 - “Granularity of interaction”:
 - Is it possible to receive intermediate results (list of taxa remaining, number of taxa remaining)? Is this feed-back occurring automatically after each user action, or does it have to be explicitly requested?
 - Is it possible to enter multiple observations or answer multiple questions before the next time-consuming interaction occurs? In a local application, such a time-consuming step may be the evaluation of best recommended characters, or the calculation of the list of remaining taxa. In a web-based application it may be relevant whether it is possible to perform several such steps (e. g., answer multiple couplets, or enter several quantitative values) before sending answers back to the server.
 - Are methods for a final “browse-identification” provided (i. e., if the identification is incomplete and terminates with a set of taxa rather than a single taxon)? What kind of information is provided during this phase?
 - Is it possible to switch between different identification methods? Are identification progress data transferred – at least in part – from one method to another?
 - Are higher-order identification tools integrated? Examples might be a color picker to input the color of an object part, an algorithmic shape picker, image or sound analysis of imported media files, or interfaces to automatic data collection routines (e. g., chromatographic data).
- Many more criteria to differentiate computer-aided keys may be defined. A rich source for additional criteria is the comparison of computer-aided multi-access keys by Dallwitz (2005a).

5.3. Presentation styles of identification keys

Printable branching keys

The dominant presentation styles of branching keys are two styles which may be called “linked, parallel, or bracketed” and “nested, yoked, or indented” (Fig. 120).

- In the *linked, parallel, juxtaposition, or bracketed* style each couplet is numbered and all leads of a couplet immediately follow each other. A pointer linking to the resulting couplet, taxon, or subkey is given at the end of each lead.
- In the *nested, yoked, or indented* style, the leads within a couplet are split and all couplets that logically follow a given lead also follow immediately in display sequence. These couplets are usually indented. Taxon or subkey results follow at the end of leads like in the previous style, but pointers to the next couplet are redundant except for redirections (occasionally redundant pointers may be provided nevertheless, e. g., in Schuster 1958).
- Metcalf (1954) further mentions a *grouped style*, which in the present author’s estimate is very rare and, judging from the example in Metcalf, rather difficult to use. It is not further discussed here.

The terms *bracketed* and *indented key* seem to be the most established ones, but are somewhat questionable:

- The term “*bracketed*” continues to escape the intuition of the author. It probably relates to a historical typographical layout where brackets or braces were spanning multiple lines as a means of visually keeping couplets together (see, e. g., Fig. 6.2 in Pankhurst 1991). Among the alternatives, the term “*parallel key*” is preferred in Lawrence (1951) and Pankhurst (1991).

Unfortunately, all senses found for *parallel* in the Collins English Dictionary (CED 1992) would be equally applicable to the nested/yoked/indented style. Only a single current usage online reference for parallel key could be traced (Little 2002). Similarly, the usage of *juxtaposition* (“key with couplets in juxtaposition”) was only found in Metcalf (1954). As a more intuitive term, “*linked key*” is proposed here, referring to the explicit linking of couplets required only in this style.

- The term “*indented*” refers to an accidental (albeit frequent) rather than essential quality of these keys (Pankhurst 1991). The indentation is occasionally omitted in “indented” keys to preserve space, relying solely on the correspondence of the lead symbols (Fig. 122 right; used, e. g., in Oberdorfer 1983). Conversely, in linked keys alternating couplets may be indented to enhance the visualization of couplets (Fig. 122 left and Fig. 125 below). “Indented key(s)” is currently much more frequently used than the more neutral term “yoked key” (340 versus 3 Google results, 2005-04-13), preferred by Lawrence (1951) and Pankhurst (1991)). As a more intuitive term, “*nested key*” is proposed here.

Various **numbering or coding systems** to designate couplets and leads are in use. The majority of linked keys denotes couplets by simple consecutive numbers (e. g., “1, 2, 3”). In linked style, the leads may be denoted by a bullet symbol (e. g., “–“, “*“, “o“, Fig. 120) instead of the couplet number. Alternatively, in both styles, symbols, primes, or letters may be added to the second lead (1/1*, 1/1', 1/1b), or letters may be added to each lead in a couplet (1a./1b., Fig. 121, left). Occasionally numerals are added at each lead, resulting in a hierarchical numbering system of leads rather than couplets (e. g., “1.1/1.2, 1.1.1/1.1.2, 1.1.1.1/1.1.1.2”). Today this is rarely used because the lead codes quickly become lengthy and difficult to compare.

As an alternative to couplet numbers, letters may be used (e. g., A/A*, α/α). This style is relatively common in nested keys. Greek letters may be used once the Latin alphabet is exhausted. A combination of upper and lower case Latin and Greek letters provides for 100 couplets; this range may be further extended by using symbols (such as †‡◊◊) or letters with accent, diaeresis, tilde, cedilla, etc.

A special form primarily used in nested keys is the repetition of the couplet letter for the second lead (A/AA, B/BB, etc.). A disadvantage of this style is that for humans the corresponding second lead (which in nested keys may easily be on a different page) is more difficult to find, because it has a different graphical appearance than the first lead. Obviously, this style cannot be combined with numerals.

Numbering may occasionally be omitted in the nested style. If indentation is the only remaining clue to which leads belong to a couplet, this style is suitable only for very short keys. However, an attractive style for nested keys is to formulate the start of the leads to be identical within a couplet, and unique among the following (nested) couplets. Bold-printing these start phrases results in good orientation within a nested key (Fig. 121, right).

In the case of polytomous keys, more than two leads may be denoted as 1a/1b/1c/1d, 1/1'/1''/1''' or A/AA/AAA. In the linked style, it is also possible to simply use the same bullet symbol for the second and all further leads.

As a result of the numbering or coding systems listed above, the linked style adapts more easily to polytomous keys (i. e., more than two leads). Even in moderately sized, single-page nested keys it is considerably cumbersome and error-prone to find all corresponding leads of a polytomous couplet. Whereas the second lead must be present in all keys, further leads may or may not be present in polytomous keys, requiring the user to always scan the key to the end. The problem can be greatly reduced by various methods. One solution is to provide an indication of the number of leads, e. g., using a count-down numbering system. For example, for a couplet with three leads, one may use 1'' (or 1c or 1(3)) for the first lead, 1' (or 1b or 1(2)) for the second, and 1 (or 1a or 1(1)) for the last (Fig. 123, left). However, the author has not encountered such a key yet. Other methods are not explicit about the number of leads, and only generally alert the user to the existence of further leads. Pankhurst (1991) suggests that nested keys could carry indications in the print margin, where the next lead could be found. A special indication could be used for the

last lead. The example with three leads, might then look like *l* (*p. 3*) for the first lead, *l* (*bottom*) for the second, and *l* (*end*) for the last. Finally, one could use arrows, e. g., a downward arrow (“↓”) after the first, an upward arrow (“↑”) after the last, and a bidirectional arrow (“↕”) for all further leads (Fig. 123, right). The advantage of the arrows is that they require very little space and do not depend on formatting features available only in print (page margin, pagination).

Backtracking: A variant of the linked presentation style uses back pointers to simplify backtracking (to the previous question that led to the current one, also called back-tracing) in a key. Backtracking is important if users detect a situation where all leads of a couplet are not applicable to the object and conclude that they must have made an error in an earlier couplet. In the most frequently used backtracking style the number of the lead from which a particular couplet is derived is given in brackets after the couplet number. For example, a couplet starting with “25 (6)” indicates that one of the leads of couplet 6 pointed to couplet 25. In the case of a redirection, the couplet may be derived from two couplet numbers and the notation may look like “25 (6 or 102)”. Except for the case of redirection, separate back pointers are not necessary in the nested style of branching keys.

Linked Key Style (= parallel / bracketed style):

1. Ovule solitary, basal 2
 - Ovules numerous, axile or free-central 3
2. Perianth green, membranous or absent; stamens with free filaments **Chenopodiaceae**
 - Perianth translucent and papery; stamens with the filaments often united below **Amaranthaceae**
3. Placentation axile; leaves alternate **Saxifragaceae**
 - Placentation basal or free-central; leaves usually opposite 4
4. Sepals free; stamens on the same radii as or more numerous than perianth-segments .. **Caryophyllaceae**
 - Sepals united; stamens as many as and on radii alternating with the perianth-segments ... **Primulaceae**

Nested Key Style (= yoked / indented style):

1. Ovule solitary, basal
 2. Perianth green, membranous or absent; stamens with free filaments **Chenopodiaceae**
 2. Perianth translucent and papery; stamens with the filaments often united below **Amaranthaceae**
1. Ovules numerous, axile or free-central
 3. Placentation axile; leaves alternate . **Saxifragaceae**
 3. Placentation basal or free-central; leaves usually opposite
 4. Sepals free; stamens on the same radii as or more numerous than perianth-segments **Caryophyllaceae**
 4. Sepals united; stamens as many as and on radii alternating with the perianth-segments **Primulaceae**

Figure 120. Examples of the linked and nested styles of branching keys in lead style (content after Davis & Cullen 1989).

Linked Key Style (= parallel / bracketed style):

- 1.a Ovule solitary, basal 2
 - b Ovules numerous, axile or free-central 3
- 2.a (1) Perianth green, membranous or absent; stamens with free filaments **Chenopodiaceae**
 - b Perianth translucent and papery; stamens with the filaments often united below **Amaranthaceae**
- 3.a (1) Placentation axile; leaves alternate **Saxifragaceae**
 - b Placentation basal or free-central; leaves usually opposite 4
- 4.a (3) Sepals free; stamens on the same radii as or more numerous than perianth-segments .. **Caryophyllaceae**
 - b Sepals united; stamens as many as and on radii alternating with the perianth-segments ... **Primulaceae**

Nested Key Style (= yoked / indented style):

- Ovule** solitary, basal
 - Perianth** green, membranous or absent; stamens with free filaments **Chenopodiaceae**
 - Perianth** translucent and papery; stamens with the filaments often united below **Amaranthaceae**
- Ovules** numerous, axile or free-central
 - Placentation** axile; leaves alternate **Saxifragaceae**
 - Placentation** basal or free-central; leaves usually opposite
 - Sepals** free; stamens on the same radii as or more numerous than perianth-segments **Caryophyllaceae**
 - Sepals** united; stamens as many as and on radii alternating with the perianth-segments **Primulaceae**

Figure 121. Variant styles of branching keys. On the left a linked key is shown denoting leads by adding a/b to the couplet number and adding backtracking information in brackets. On the right a nested key is shown, displaying the start of the lead text in bold as couplet marker (compare Fig. 120).

Linked Key Style (= parallel / bracketed style):

1. Ovule solitary, basal 2
1. Ovules numerous, axile or free-central 3
 2. Perianth green, membranous or absent; stamens with free filaments **Chenopodiaceae**
 2. Perianth translucent and papery; stamens with the filaments often united below **Amaranthaceae**
3. Placentation axile; leaves alternate **Saxifragaceae**
3. Placentation basal or free-central; leaves usually opposite 4
 4. Sepals free; stamens on the same radii as or more numerous than perianth-segments **Caryophyllaceae**
 4. Sepals united; stamens as many as and on radii alternating with the perianth-segments **Primulaceae**

Nested Key Style (= yoked / indented style):

- 1↓ Ovule solitary, basal
- 2↓ Perianth green, membranous or absent; stamens with free filaments **Chenopodiaceae**
- 2↑ Perianth translucent and papery; stamens with the filaments often united below **Amaranthaceae**
- 1↓ Ovules numerous, axile or free-central
- 3↓ Placentation axile; leaves alternate **Saxifragaceae**
- 3↑ Placentation basal or free-central; leaves usually opposite
- 4↓ Sepals free; stamens on the same radii as or more numerous than perianth-segments . **Caryophyllaceae**
- 4↑ Sepals united; stamens as many as and on radii alternating with the perianth-segments ... **Primulaceae**

Figure 122. Further variants of branching keys, illustrating that (alternating) indentation may be present in the “non-indented” linked style and omitted in the “indented” nested style (compare Fig. 120).

- 1" Ovule solitary, basal
- 2" Perianth green, membranous or absent **Chenopodiaceae**
- 2' Perianth translucent and papery.... **Amaranthaceae**
- 1" Ovules numerous, placentation axile; leaves alternate **Saxifragaceae**
- 1' Ovules numerous, placentation basal or free-central; leaves usually opposite
- 3" Sepals free **Caryophyllaceae**
- 3' Sepals united..... **Primulaceae**

- 1↓ Ovule solitary, basal
- 2↓ Perianth green, membranous or absent **Chenopodiaceae**
- 2↑ Perianth translucent and papery ... **Amaranthaceae**
- 1↑ Ovules numerous, placentation axile; leaves alternate **Saxifragaceae**
- 1↑ Ovules numerous, placentation basal or free-central; leaves usually opposite
- 3↓ Sepals free **Caryophyllaceae**
- 3↑ Sepals united..... **Primulaceae**

Figure 123. Branching keys in nested style adapted for use with polytomous couplets, using a count-down method (left) or arrows (right).

A variant the presentation styles (usually nested, but potentially linked keys as well) are “solid keys”, where the line-structure of linked or nested keys is replaced by bold formatting or graphical elements within lines in an attempt to save printing space (Leenhouts 1966, Pankhurst 1991). Naturally, such keys are considerably more difficult to use, which may lead to an increased error rate.

Linked and nested styles have different *advantages* and *disadvantages*. The linked style is more suitable for long keys and polytomous keys, because finding and comparing the corresponding leads in a couplet is more convenient and reliable. Compared with nested keys using indentation, it makes better use of available space. The major advantages of the nested style is that the resulting groups can clearly be seen and that it is much easier for experts to make use of additional knowledge about the species being keyed out. This style is often preferred by taxonomists who wish to express taxonomic structure directly in the key. Furthermore, even where a linked key supports backtracking through additional backpointers, this is more error-prone than the intuitive backtracking in a nested key. In general, nested style is recommendable mostly for short keys of less than a page in print.

In the light of these differences, Metcalf (1954) proposes to use *combination keys*, combining the advantages of both styles. This can occur both ways: by first using a linked style, separating the major parts of the key into small subkeys each of which is small and uses the nested style, or by first using nested style, with linked keys in between to save indentation space. Using indentation and different lead codes (e. g., letters for nested style and numbers for linked style), the resulting keys can be quite intuitive to use.

Finally, a rare but very appealing presentation form of branching keys might be called “graphical style”. Here the graphical elements and illustrations dominate layout and key structure (Fig. 124).

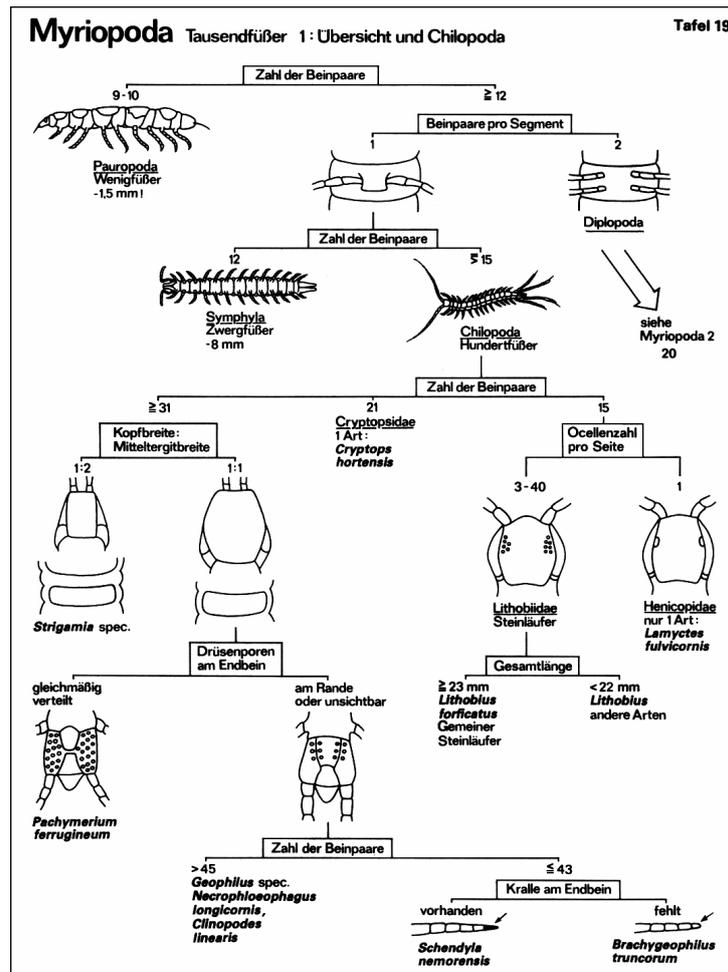


Figure 124. An example of a branching key in graphical style (polytomous question/answer style, after Müller 1985). The combination of text and illustrations is such that either may be missing. Regions of interested are highlighted with arrows (bottom right) and graphical symbols are used for subkeys (upper-right).

Important material on the history and principles of manually constructing branching keys may be found in Voss (1952), Metcalf (1954), and Pankhurst (1991).

Many computer applications support the creation of printable keys. The first such programs appeared over 35 years ago (e. g., Pankhurst 1970a, Pankhurst 1970b) and both the current CSIRO DELTA suite (KEY, KEYQW) and Pankey (KEY3M3, KPRINT, p. 19) provide fully automated tools that generate branching keys based on DELTA data. These tools have various options to influence the style of the key that is being generated. The Pankey suite also provides a program to create keys interactively in a combination of user choices, and DELTA-data based recommendations (KCONI, Pankhurst 1988). Both the CSIRO and Pankey tools are DOS-based programs that can be run under Windows.

The optimality criteria for algorithmic creation of branching keys are discussed further below under “Character ranking and guidance” (p. 267). They are usually a combination of algorithmic and authored guidance (p. 276). The fundamental principle is that minimizing the average length of the key not only minimizes average identification time, but also maximizes the average chance of correct identification (for a fixed and equal probability of answering key questions wrongly, Osborne 1963). Since in practice the probability of error is different for individual questions (and may be estimated based on authored character guidance information), Pankhurst (1991) further points out that the probability of obtaining the correct identification is further maximized by ordering couplets so that couplets with a lower error probability come first.

A new attempt to create and render branching keys is being developed in the Electronic Field Guide project (Stevenson & al. 2003) as a “Key Rendering Suite” (Morris & al. 2003, Morris & al. 2007). By using an XML-based data structure, this may then be transformed using XSLT (Clark 1999) and XSL-FO (Berglund 2006) transformations into multiple display formats such as PDF or XPS for printing, HTML for general internet use, or Wireless Markup Language (WML) for web-enabled mobile phones. Thus this application renders both printable and computer-aided branching keys.

Computer-aided branching keys

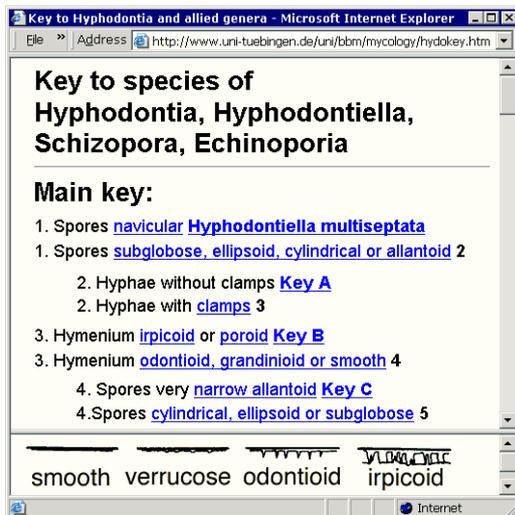


Figure 125. An example for a printable, hyperlinked dichotomous key. Glossary terms and illustrations are presented in a frame window at the bottom (<http://www.uni-tuebingen.de/uni/bbm/mycology/hydokey.htm>).

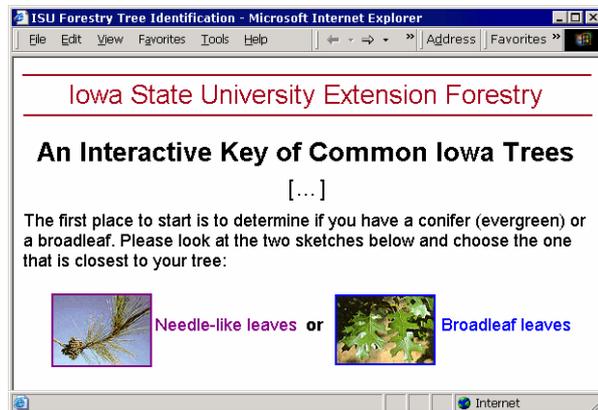


Figure 126. An abbreviated example of an online dichotomous key, split into one question per page (<http://www.extension.iastate.edu/pages/tree/key.html>).

Both the linked and nested styles of branching keys are amenable to presentation as hyperlinked text. Wright & al. (1995) and Morse & al. (1996) found that using hyperlinked branching keys may be slower and less accurate than using printed keys. They mention several potential reasons, including lack of experience with the technology and poor quality of on-screen images that can be easily overcome.

For linked web-based branching keys presenting multiple key statements on a single page (Fig. 114, p. 234; Fig. 125), one further disadvantage not discussed by Wright & al. (1995) and Morse & al. (1996) may be that following the internal hyperlinks may cause some irritation because the targeted couplet is usually not clearly visible as the next focus. Web browsers display no “cursor”. While jumping to internal section headings usually works well because the heading is clearly visible and in most cases the heading requires scrolling so that the hyperlink will position it at the top of the screen, internal links perform poorly when jumping to lines visible on the same page. This could be addressed in the future by enhancing the internal links with JavaScript-based highlighting of the target (and perhaps of the origin, i. e., the previous couplet).

Without such enhancements, the nested style (requiring no internal hyperlinks) may be preferable for small branching keys. However, for large keys the linked style will usually be preferable. It makes the leads of a couplet easily comparable with scrolling, and hyperlinking removes most of the disadvantages of paper-based linked keys (Brach & Hong Song 2006). The definition of a “large key” clearly depends on the average targeted screen technology and individual prefer-

ences as to scrolling or not. A rule of thumb may be that keys exceeding the length of two or three web pages in targeted screen technology should be considered large.

In addition to using hyperlinks for following the leads in linked keys, and jumping to subkeys or taxon information, hyperlinks may also be used to provide supplementary information such as glossary definitions or illustrations (Fig. 125).

An alternative presentation of branching keys available only on computers is to reduce the key to one question per web-page (Figs. 126-127, also Fig. 118, p. 239). This increases the concentration on a single question and may provide space for rich illustrations or explanations. On the other hand, the resulting interaction with the key will be slower and may lead to more difficulties in memorizing the path up to the present place.

Similar to multi-access keys, it is therefore highly desirable to provide a history of the accepted statements or questions answered (Farr 2006, Fig. 127). This history may either be in the form of a natural language statement or may be a “decision map” (Edwards & Morse 1995), supporting both orientation and backtracking using hyperlinks. From the perspective of expert systems, the history may be considered a “justifying module” (Fajardo Contreras & al. 2003).

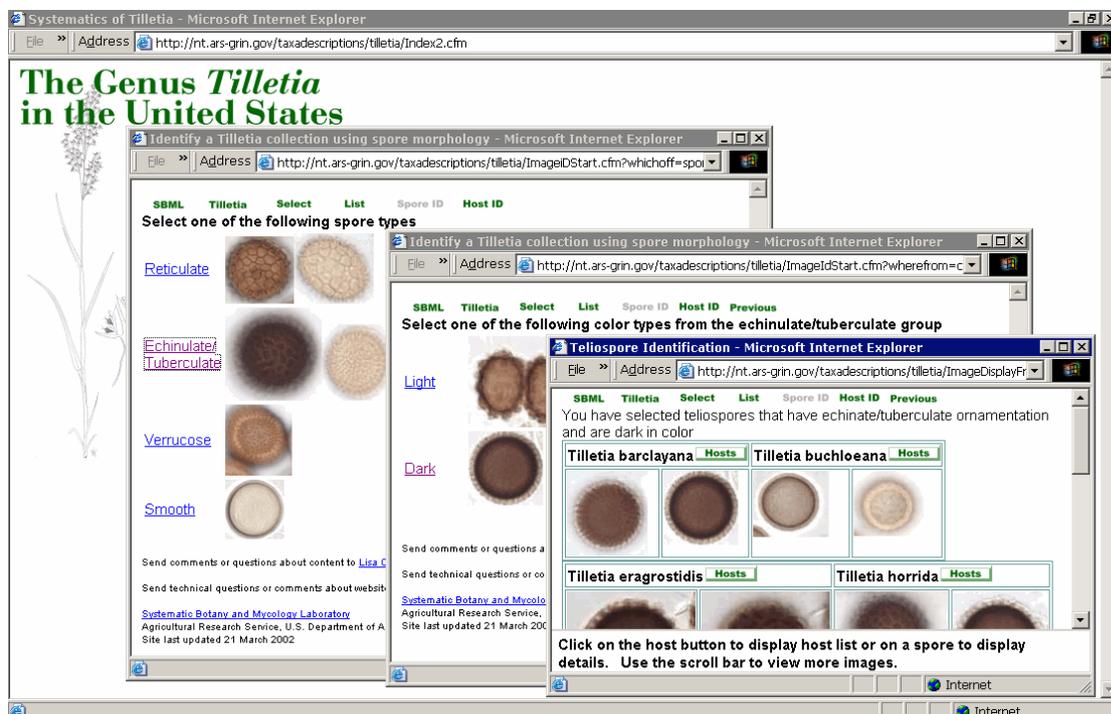


Figure 127. An example of a picture-based branching key with one question per page, supporting the user by tracking the history of previous statements (Castlebury & Farr 2002).

Probably the most sophisticated application for handling branching keys is CBIT Lucid Phoenix (Fig. 128, CBIT 2007b). The Lucid Phoenix builder supports writing new branching keys or digitizing existing printed keys and enriching them with new information (especially illustrations). The player can be run locally or as a Java-applet in an internet browser.

A special feature of Lucid Phoenix is the ability to skip a question (see lower left panel in Fig. 128). In a dichotomous keys both leads must then be continued; this is automatically reflected in the history and the list of “entities remaining”.

Naturally, computer-aided keys may be created algorithmically just like printed keys; the same comments made in the previous section apply here.

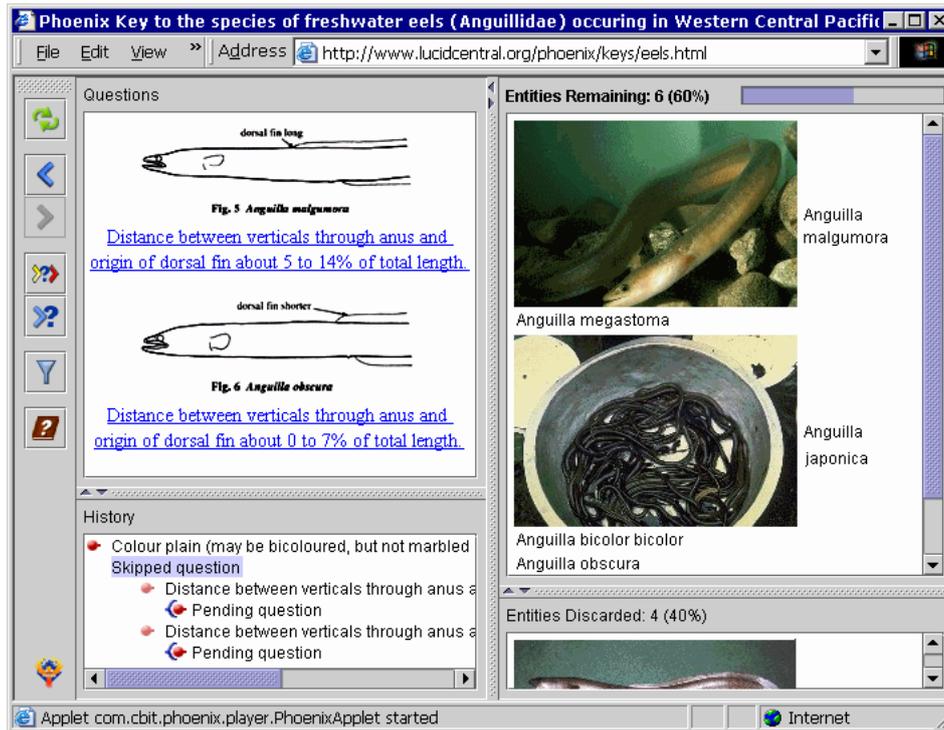


Figure 128. An example of CBIT Lucid Phoenix (version 1, running as Java applet in an internet browser, <http://www.lucidcentral.org/phoenix/keys/eels.html>). It supports skipping of questions, resulting in multiple pending questions.

Printable multi-access keys

The dominant two styles might be called “character-list style” and “taxon-list style” (or “character-formula style”; all terms introduced here). In both styles the key is composed of a list of characters that are arranged such that the user can easily choose the desired characters. In the **character-list style** (Fig. 129) the list of taxa is coded (usually numbered but occasionally abbreviated taxon names are used) and within a list of character states (equivalent to a lead in a branching key) each state has a corresponding list of taxon codes. Conversely, in the **taxon-list style** (Fig. 130), the character states are coded and each taxon is associated with the corresponding states (the “character formula”).

The *character-list style* can successfully be applied to a long list of characters and a moderately long list of taxa. Korf (1972) suggests a usability limit 30 taxa, but in the present author’s experience around 60 taxa remain practical as long as enough convenient characters exists that allow to start with considerably less than half that amount. The choice of the initial character is completely free, although guidance to reliable and convenient characters may be given by means of formatting or arrangement. An alternative guidance method is to provide “lead questions” in front of the key, suggesting suitable starting points under certain conditions (Korf 1972). Perhaps the best guidance is, however, an inherent feature of this key style: It is highly visible which character states are present only in relatively few taxa. By selecting a character that promises fast progress, only a short list of taxon codes must be copied to a piece of paper. The taxon list of states of subsequently selected characters is then compared with this list and taxon codes not found are crossed out on the paper. Thus, subsequent characters may have a much longer list of taxon codes without adversely affecting the speed and convenience of identification. It is obvious that as soon as the number of taxon codes is higher than is practical to memorize, the advantageous mnemonic properties of abbreviated alphabetical taxon codes are probably outweighed by

the inconvenience of copying and comparing them, thus leading to a preference of taxon numbers for taxa for larger character-list style keys.

Building on earlier attempts of other authors like Ogden (1943), printable multi-access keys have been most clearly described and advocated by Leenhouts (1966). Leenhouts uses the name *synoptic key*, which should be avoided (see p. 398). Leenhouts further proposes to improve and elaborate the character-list style by underlining taxon codes mentioned in more than one lead of a character (compare *versicolored pigmentation* in Fig. 129), and placing taxon codes that are unknown or uncertain in brackets.

Conidiomatal characters:		Genera of Thallostromatinae	
Stromatal type		1. <i>Neozythia</i>	
a. acervular	8	2. <i>Dothioropsis</i>	
b. cupulate	3 4 5 6 7	3. <i>Phacidiella</i>	
c. multilocular	2 9	4. <i>Sirozythiella</i>	
d. unilocular	1	5. <i>Acarosporium</i>	
Position relative to substrate		6. <i>Trullula</i>	
a. superficial	9	7. <i>Phragmotrichum</i>	
b. immersed	1 2 3 7 8	8. <i>Staninwardia</i>	
c. initially immersed, finally superficial	4 5 6	9. <i>Barnetella</i>	
Conidial characters			
Pigmentation			
a. hyaline	1 2 3 4 5		
b. brown	6 <u>7</u> 8 9		
c. versicolored	<u>7</u>		

Figure 129. An example of a printable multi-access key in “character-list style” (from Sutton 1980). For each character state the matching results are listed, abbreviated by a genus number. In this example, a multilocular fungus with brown conidia must be *Barnetella*, whereas *Sirozythiella* and *Acarosporium* are indistinguishable with the three characters given.

Genera:	Conidiomatal characters:	Conidial characters
A F I <i>Staninwardia</i>	Stromatal type	Pigmentation
B F H <i>Phacidiella</i>	A. acervular	H. hyaline
B F I K <i>Phragmotrichum</i>	B. cupulate	I. brown
B G H <i>Sirozythiella</i>	C. multilocular	K. versicolored
B G H <i>Acarosporium</i>	D. unilocular	
B G I <i>Trullula</i>	Position relative to substrate	
C E I <i>Barnetella</i>	E. superficial	
C F H <i>Dothioropsis</i>	F. immersed	
D F H <i>Neozythia</i>	G. initially immersed, finally superficial	

Figure 130. An example of a printable multi-access key in “taxon-list style” (or “character formula style”). Each character state has a unique code and each taxon carries an abbreviated formula-like description composed from a sequence of these codes.

Whereas the taxon-list style has similar properties to a computer-aided multi-access key, the *taxon-list style* is usually limited to a few characters, all of which should be conveniently and reliably observable. The style essentially creates a shorthand formula or code for diagnostic descriptions (p. 39), which then can be compared more easily than full-text descriptions. If the list of taxa is sorted by the formula and if the first characters in the formula can be expected to be always assessable, the list of taxa may be longer. The key then becomes a mixture of a branching key for the initial characters, and a multi-access key for the remaining characters in the formula. The limitation to only very few characters can be somewhat reduced by combining a small set of

generally applicable formula-characters with another column containing further diagnostic characters as natural language text (compare Fig. 11 in Leenhouts 1966). The characters used in this additional text are not expected to be consistently used throughout the key. This “character-formula style” is closely related to the tabular keys – see p. 256 – where non-abbreviated text is arranged in a tabular format, to increase the comparability. Some further presentation forms of printable multi-access keys surveyed in Leenhouts (1966) use a similar tabular arrangement to achieve their goal.

Computer-aided multi-access keys

Computer-aided keys may be running locally (on a PC, PDA, etc.) or may require a network connection. Typical examples of the first category are ETI Linnaeus 2/IdentifyIT (p. 19), Diversity-Descriptions Identify (Fig. 131), or Pankey ONLIN7 (DOS-based, p. 19). An intermediate type of applications is able to run both locally and over the internet. Typical examples are CSIRO Intkey (Fig. 132), the Java-based CBIT Lucid 3 (Fig. 133 local, Fig. 134 in browsers) and Navikey 4 (Fig. 135, 136) which can be run entirely by opening a web page (provided a sufficiently new version of the Java virtual machine has already been installed locally). In contrast, Intkey can acquire data sets over the internet, but must first be downloaded and locally installed (since ca. 1995 Intkey is available free of charge).

Other internet-based tools are less feature-rich, but use only standard internet browser functionality. An example of a design that is limited to a small set of characters, but where all characters can be evaluated without sending and receiving intermediate web pages is shown in Fig. 137. A simple design that supports very long character lists is the server-based “DeltaAccess-Perl” (DAP, Fig. 138). This key works in standard internet browsers even if Java and JavaScript are disabled. Quite sophisticated keys can be realized using a combination of server-based programming and client-side JavaScript, as exemplified by X:ID (Fig. 139) or ActKey (Brach & Hong Song 2005, Fig. 140).

It may be desirable to define multilingual branching keys, where each statement has representations in multiple languages. Managing language versions as separate keys is difficult when keys are improved and corrected over time. Especially if a key is only partially translated, it may be desirable to designate default and fallback languages; untranslated parts could then be rendered in a “default language”. Among the available key software, the PICKEY (p. 22) and 3I (p. 21, Figs. 141-142) multi-access keys are the only ones known to the author with multilingual support (compare Lobanov & al. 2005).

Some relatively simple (e. g., SLIKS, Guala 2006) or sophisticated (SAIKS, Alexander 2006a, 2006b, 2006c; Visual Key, Klimov 2001) keys even run entirely in the web browser, using exclusively ECMA or JavaScript to create the user interface. Using client-side code allows for considerably greater responsiveness to user actions than the server-based logic of most other web identification interfaces. This is highly visible in SAIKS (Figs. 143-145) which – by changing the background color of text – gives immediate feedback about remaining and excluded taxa as well as contradictory character states (states that, if selected, would lead to zero remaining taxa). In contrast to SLIKS, SAIKS supports multi-state characters, enabling it to support a “show character” mode, in which the description of a single taxon becomes highlighted (Fig. 144). Clicking on a result taxon calls up a species page or the next subkey (Fig. 145).

The techniques mentioned above are currently limited to small to medium size data sets, but using client-side code with asynchronous server-based data updates (e. g., using AJAX, Aynchronous JavaScript and XML) they could in the future be scaled up to truly large data sets.

An important feature of multi-access keys is whether they assist users by providing recommendations which characters would be most effective to use next. This aspect is discussed in detail “Character ranking and guidance” (p. 267) and “Presentation of character guidance in multi-access keys” (p. 277).

The most extensive comparative list of computer-aided multi-access keys is provided by Dallwitz (2007), with additional details to be found in Dallwitz (2005a), Lobanov & al. (2005), and OConnor & Klimov (2004b). Further important references when comparing multi-access keys are Pankhurst (1991; especially for principles and a detailed account of the history of key development), Edwards & Morse (1995; including other kinds of keys like expert systems and early applications of neural networks), Lindh (2003, in Swedish), Gibaja Galindo (2004, in Spanish), Rambold (2002; especially lichen keys), Farr (2006), Dallwitz & al. (2007), and the link lists http://www.zin.ru/projects/pickey/pic_link.htm and <http://www.geocities.com/rainforest/vines/8695/software.html>.

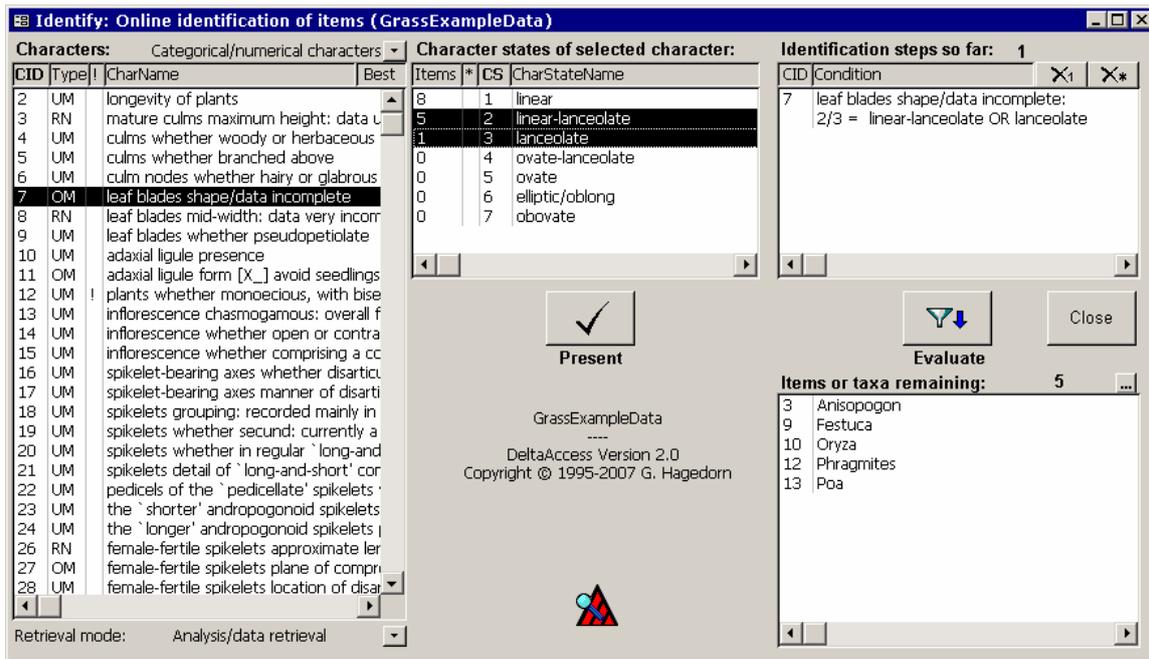


Figure 131. An example of an *Identify* key running in DiversityDescriptions. Multiple states selected (center) indicate polymorphism or uncertainty. The lower center space is intended for resources such as images (not implemented yet).

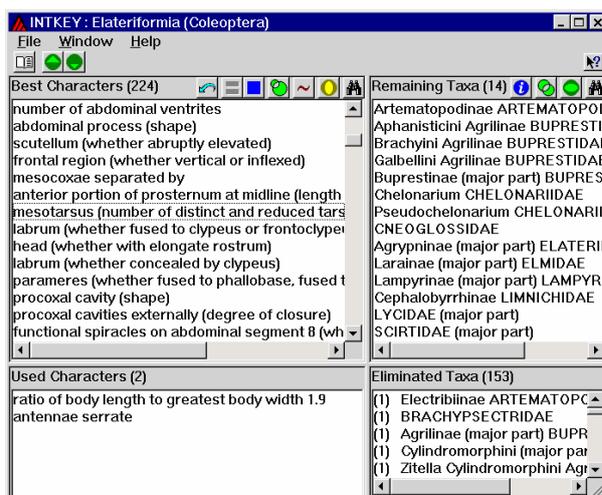


Figure 132. An example of a CSIRO Intkey key running under Windows (after Dallwitz & al. 2007).

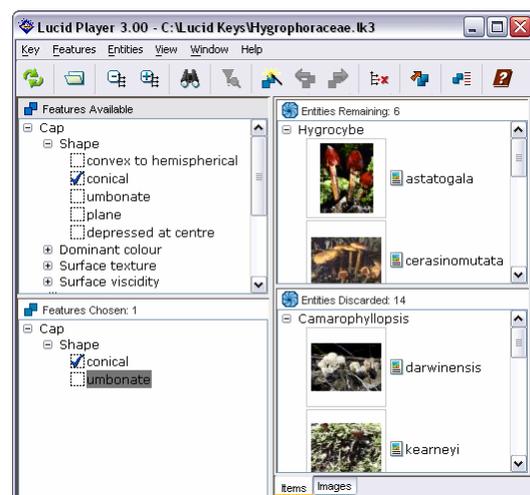


Figure 133. An example of a Lucid 3 key running under Windows (screenshot from <http://lucidcentral.org/lucid3/>).

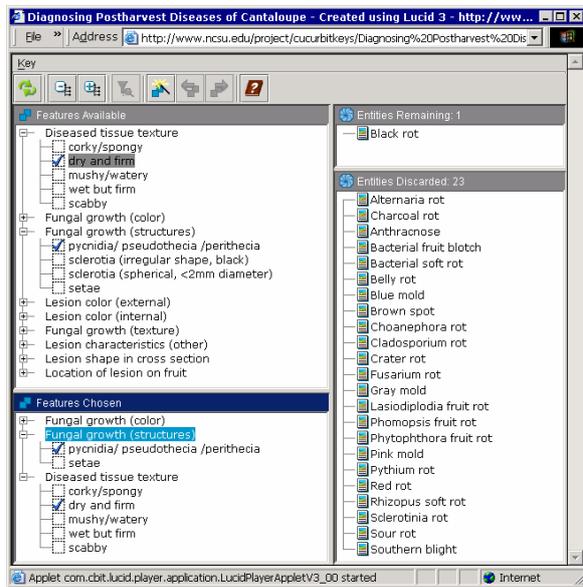


Figure 134. CBIT Lucid 3 may also be used over the internet if Java applets are permitted (from [http://www.ncsu.edu/project/cucurbitkeys/Diagnosing Postharvest Diseases of Cantaloupe.html](http://www.ncsu.edu/project/cucurbitkeys/Diagnosing%20Postharvest%20Dis...html)).

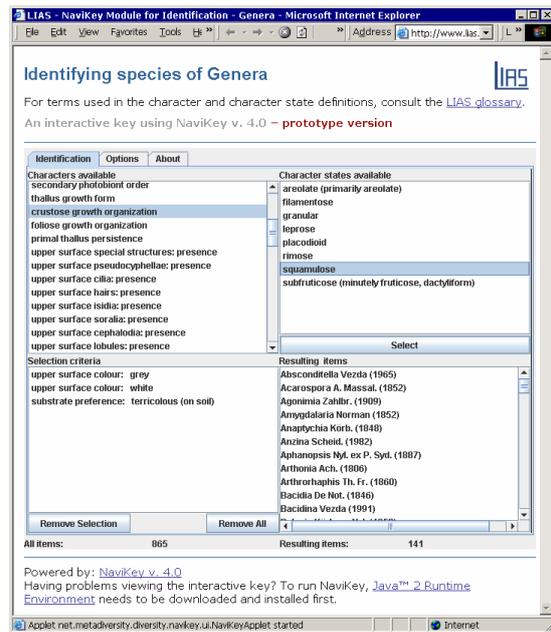


Figure 135. An example of a Navikey 4.09 Java applet running in an internet browser (<http://www.lias.net/Identification/GenusLevel.html>).

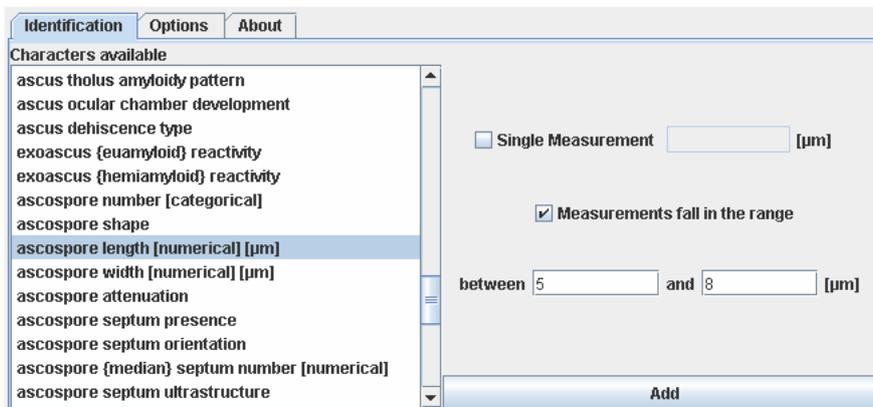


Figure 136. Navikey 4.09 using a quantitative character. Compare Fig. 135 for context and Fig. 155 (p. 266) for DiversityDescriptions.

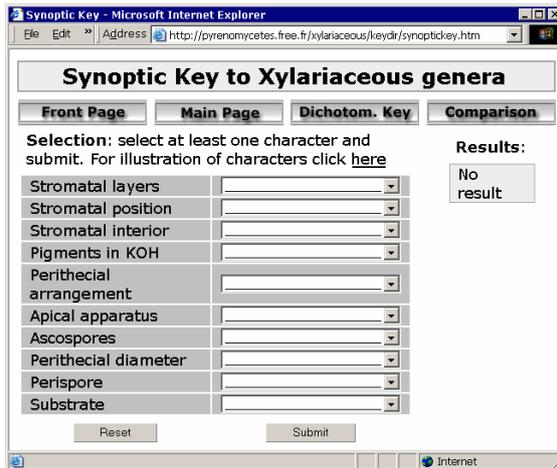


Figure 137. An example of a multi-access key that is limited to a few pre-selected characters. These may be answered in a single step without sending the form multiple times to the web server (<http://pyrenomyces.free.fr/xylariaceous/keydir/synoptickey.htm>). Compare p. 398 for use of “synoptic”.

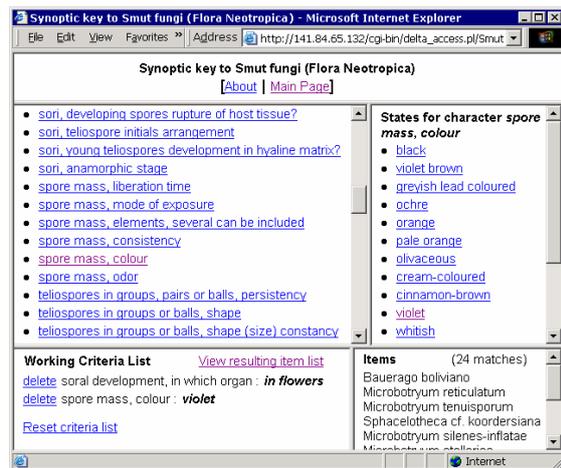


Figure 138. A simple multi-access key using DeltaAccess-Perl (DAP), suitable for a long list of characters. For each character selected in the center left panel the corresponding states appear in the center right panel. Selecting a state adds it to the Criteria list (example accessible from <http://www.glopp.net>).

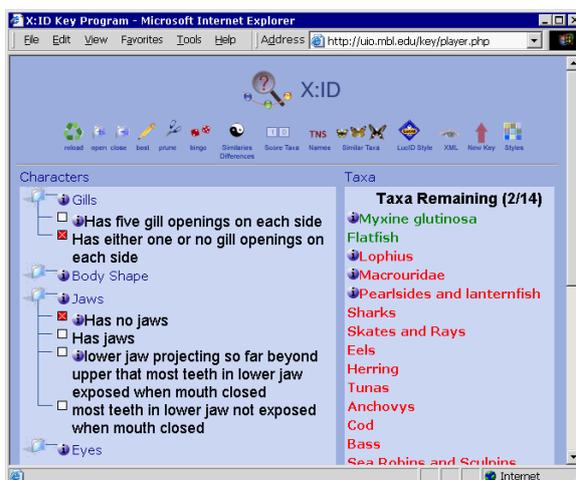


Figure 139. An example of an X-ID key (p. 21), running in the web browser (accessible from <http://uio.mbl.edu/services/key.html>).

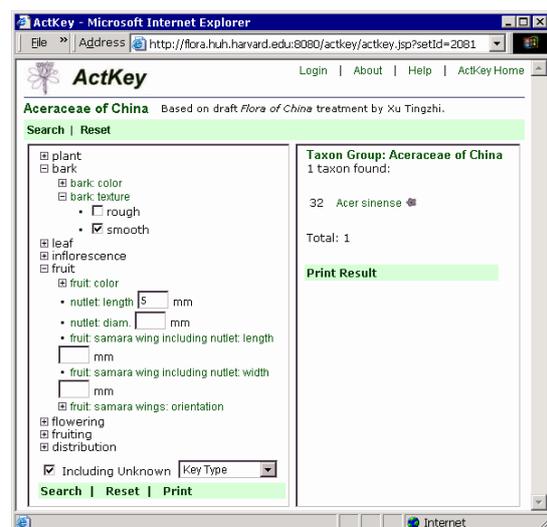


Figure 140. A sophisticated multi-access key using ActKey (p. 20, <http://flora.huh.harvard.edu:8080/actkey/actkey.jsp?setId=2081>).

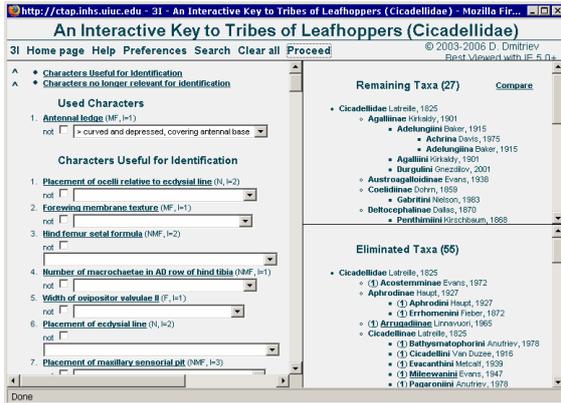


Figure 141. An example of a 3I web key (p. 21). After each evaluation, characters are reordered into categories (used, useful, no longer relevant) and the used characters are ranked by a character guidance algorithm.

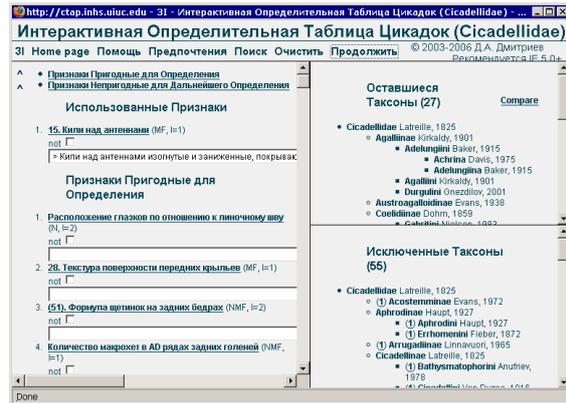


Figure 142. 3I is multilingual, here the key from Fig. 141 is displayed in Russian. Accessible from <http://ctap.inhs.uiuc.edu/dmitriev/3i.asp>.

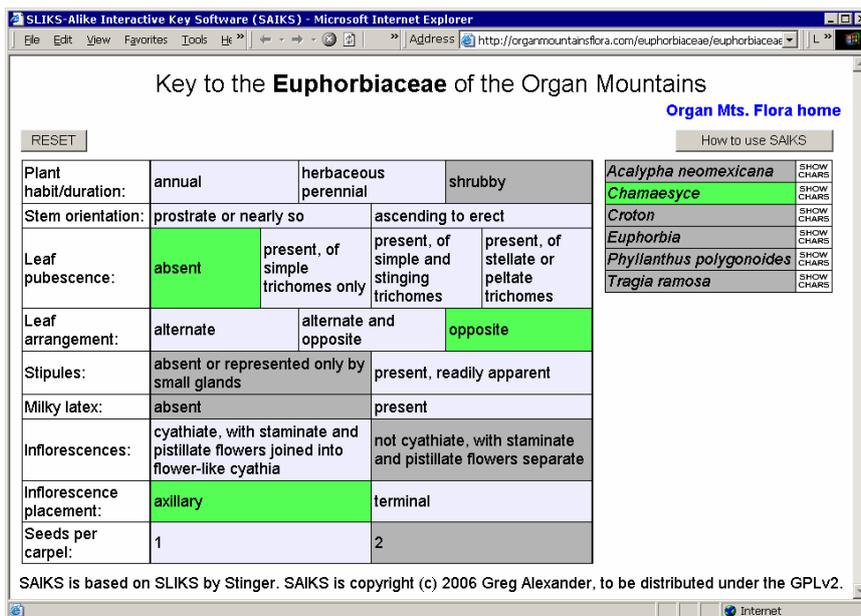


Figure 143. An example of a SAIKS key (Alexander 2006c) in identification mode. Clicking on a character state scores/resets the state, indicated by a change of background color (green/darker background = scored). Immediately, the background color of the taxon list (right) is updated (lighter/green = remaining, darker gray = excluded), and the background of states inapplicable for all remaining taxa is set to dark gray (colors slightly changed to improve gray-scale printing).

Figure 144. An example of a SAIKS key (Alexander 2006c) in “show-character” mode – all character states recorded for a selected taxon are shown.

Figure 145. An example of a SAIKS key (Alexander 2006c) showing a subkey with back-reference to the previous key.

Tabular keys

A tabular key (Figs. 146–147) is a special key style combining properties of branching and multi-access keys. Like a branching key, it is most conveniently used in a predefined sequence (columns from left to right, or rows from top to bottom, depending on the pivotal arrangement of the key). However, because of the tabular arrangement of information, and because information that is relevant in some parts of the key is also supplied in parts where it might be considered redundant (e. g., columns 2–4 in Fig. 146 for the upper species, having large pits in rays), it is also possible to use it as a multi-access key. The arrangement of tabular keys corresponds closely to the taxon-list style of printable multi-access keys (Fig. 130, p. 250). In both types of keys, choosing a different sequence of characters results in losing the advantage of sorting, but at least for relatively small keys it remains a viable option.

The tabular arrangement is very space-consuming so that only a few characters can be arranged in this manner. Usually the entire key is therefore split into many small subkeys, and characters specific to the differentiation of certain taxa may be presented in the form of natural language descriptions rather than in separate columns (Fig. 146, right column).

Normally, the tabular arrangement involves some form of projecting the three-dimensional character \times state \times entity information into two dimensions. A new and interesting variation can be seen in the SAIKS program (p. 251, and Figs. 143–145). Although not a classical tabular arrangement (taxa are in a separate list, thus states can simply be nested in characters), it represents the third dimension through colors. This is practical only when representing a single species at a time, but SAIKS achieves a highly synoptic user experience in “show character” mode, in which all character states of a single taxon become highlighted (Fig. 144). Although the approach is not a tabular key, it has many of the same advantages and allows substantially more characters and taxa to be used in a single key.

Tabular keys are found both in print and as hyperlinked text on the web. The KEY program of the CSIRO DELTA suite is able to create a simple tabular key, where the cells contain codes for each character and state combination (Dallwitz & al. 2000a).

Tabular keys may include illustrations, either through reference, or sometimes directly within the table cells (Fig. 147).

Coniferous wood		Wood without vessels (pores), with tracheids only			Species	Key characteristics
Pitting in rays	Resin canals	Transversal tracheids	Spiral thickenings in tracheid walls			
large (pinoid pits)	present	present	absent	Pinus sylvestris	Transversal tracheids with dentated walls, cannot be differentiated from <i>P. mugo</i> and <i>P. nigra</i> .	
				Pinus mugo	Transversal tracheids with dentated walls, cannot be differentiated from <i>P. sylvestris</i> and <i>P. nigra</i> .	
				Pinus nigra	Transversal tracheids with dentated walls, cannot be differentiated from <i>P. sylvestris</i> and <i>P. mugo</i> , sometimes early-/latewood transition seems more abrupt than in the two other species.	
				Pinus cembra	Transversal tracheids with smooth walls, in general small growth rings, latewood zones always narrow.	
				Pinus strobus	Transversal tracheids with smooth walls, in general larger growth rings, latewood zones narrow, similar to <i>P. cembra</i> .	
small	present	present	absent	Picea abies	Transition from early- to latewood continuous. Resin canals with thick-walled epithelial cells. Bordered pits in axial tracheids generally uniseriate.	
				Larix decidua	Transition from early- to latewood abrupt. Resin canals with thick-walled epithelial cells. Bordered pits in axial tracheids often biseriata.	
	absent	absent	absent	Pseudotsuga taxifolia	Transition from early- to latewood abrupt. Resin canals with thick-walled epithelial cells. Fine spiral thickenings.	
				Abies alba	Transition from early- to latewood abrupt. In radial section the tangential ray cell walls show distinct nodular chains.	
				Juniperus communis	Colored deposits in parenchyma cells, smooth walled ray cells.	
				Taxus baccata	Distinct spiral thickenings.	

Figure 146. An example of tabular key (left columns), combined with short diagnostic descriptions (right column). The preferred usage is from left to right, but by supplying redundant information some degree of “multi-access usage” is supported (excerpt from http://www.woodanatomy.ch/ident_key.html).

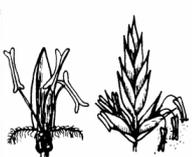
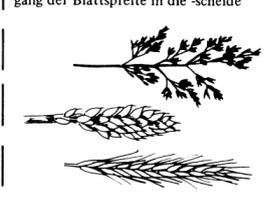
Stengel	Blattstellung	Blattscheiden	Blüten	Blütenstände	Frucht	Sonstiges	Name
rund massiv, knotenlos	oft 3zeilig	z.T. sichtbar blattlos	klein, unscheinbar, mit trockenhäutigem Perigon, mit dem Bau der Liliaceenblüte 	Spirre (z.T. köpfchenartig)	Kapsel 	oft auf feuchten Standorten 	Juncaceae (Binsengewächse)
	oft 3kantig	meist geschlossen	in der Achsel trockenhäutiger Tragblättchen (Spelzen), oft 1geschlechtlich  sehr stark reduziert	Ährchen (oft 1blütig) zu Ähren, Köpfchen oder Spirren vereinigt	Nüßchen 		Cyperaceae (Sauer-, Riedgrasgewächse)
rund, hohl, mit deutlichen Knoten («Halme»)	2zeilig	meist offen 	in Ähren mit Hüll-, Deck- und Vorspelzen. Deckspelzen oft begrannt 	Ährchen in Ähren, Scheinähren oder Rispen	Korn (Karyopse) 	v.a. Wiese 	Poaceae (Gras-, Süßgrasgewächse)

Figure 147. An example of tabular key including illustrations (Haller & Probst 1989).

5.4. Requirement summary

Requirements that may be deduced from the previous sections for an information model to store authored identification keys have been compiled here:

201. The information model needs special data structures for branching keys. These are *required* for authored (or “designed”) keys. They may also be *desirable* to store (cache) algorithmically generated branching keys for fast retrieval.
202. Metadata supporting such a distinction between authored and algorithmically generated keys may be desirable, including information about last update for algorithmically generated keys.
203. Authored branching keys need certain metadata like name, title, description, expected experience level of user (untrained, generally trained, specialist), and information on available languages. The metadata for a branching and multi-access key are generally identical with those of a set of coded descriptive data. For key operation, however, additional metadata may be required (compare requirement 229, p. 276).
204. Dichotomous keys are a special case of polytomous keys and may be stored in a model for the latter. The author of a key may desire to indicate that a key should remain strictly dichotomous; this may be stored as metadata item specific to branching keys. Key builder applications (“editors”) may recognize and either warn or prevent the user from adding more than two leads. The distinction is, however, not considered central enough to warrant enforcement by the information model itself.
205. Combinations of broad branching keys and groups of browsable descriptions/illustrations – commonly found in field guides – can be modeled as a special case of polytomous keys, with a high number of lead choices at the end. However, the information model must support empty lead text and perhaps a metadata item requesting “embedding” of descriptions or illustrations instead of the usual linking.
206. Leads in branching keys may lead to other couplets, entire keys (“subkeys”), or taxon names/descriptions. The choice is a strict alternative in most cases (but see requirements 207 below and 220, p. 262).
207. The “result-and-continue pattern” (compare p. 235) implies requirements that may lead to either supporting both a taxon group and a following couplet at the end of a lead, or presentation metadata indicating that a subkey shall be embedded in a higher-level key. Exactly how to support this situation needs further study.
208. Branching keys may provide for redirection (“cross-linking”, “reticulation”). As a consequence, the information model cannot be limited to a tree, but must support directed acyclic graphs (DAGs).
209. It is desirable that branching keys may support the question/answer and the lead style. The question/answer style requires an additional “question label” at each couplet.
210. The question/answer style may also occur in multi-access keys, requiring an additional “question label” for each character. This item is separate from the question in branching keys, because a couplet in branching keys may involve multiple characters.
211. The text in branching keys (question/answer or lead style) should be free-form text and provide for minimal inline text formatting such as italics, bold, sub- and superscript.
212. Multiple keys for a single set of taxa should be supported (for a plant family, for example, separate keys based on flowers, fruits, and vegetative organs might be desirable).
213. Multilingual support for branching keys is desirable, especially for complex keys intended for ongoing revisions. Additional metadata on default language, fall-back language etc. may be desirable.
214. Media resources (images, etc.) are required at all nodes in an identification key, not only on terminal nodes. At higher nodes they may either illustrate diagnostic features, or support the concepts of “looks like”/“promorphs”.
215. Media resources require context-dependent captions. The same resource may be used at different points in the key, illustrating different character concepts or the entire resulting taxon.

216. Presentation styles of branching keys may be supported as stored preferences in the meta-data, but are not an issue of the required data structures.
217. Tabular keys may have additional requirements (e. g., a fixed set of few characters and sequence of characters determining the sort sequence) that cannot be represented with the polytomous branching key model. Whether this warrants an independent model, or whether it can be included in a model for character guidance needs further study.
218. The confirmation phase (computer-aided choice of similar taxa) may be based on algorithmically determined similarity, or it may be based on manually entered lists of “easily confused taxa”. In the latter case, data structures for this must be presented in the information model.

5.5. Linking multiple keys

A hierarchy of keys (i. e., entry key with one or multiple levels of subkeys; Fig. 149) is standard practice in biology and found both with branching and multi-access keys. Although publishing constraints and difficulty in handling very large printed keys contribute to this pattern, authoring and management factors similarly lead to a preference for well-defined building blocks instead of unwieldy monolithic “superkeys”. Small to moderately sized keys can be authored and reviewed by specific experts, resulting in keys of known quality and known taxonomic concept. In contrast, “superkeys” have a tendency to be of widely different quality in ways that is difficult to document or perceive.

In the simplest pattern supporting this, a key is considered an “identification resource”, regardless of its format (paper, static digital formats like PDF or XPS, computer-aided, etc.). A lead pointing to a subkey thus consists of a display title in appropriate languages and a resolvable URI to obtain the resource.

Occasionally, it may be desirable to offer multiple alternative subkeys to the user. Keys are typically not only defined by their taxonomic coverage, but may also be scoped geographically (e. g., “Orchids of Bavaria”), ecologically (e. g., “high-altitude shrubs”), seasonal (e. g., “winter key”), sexual state (e. g., “vegetative key”) or by lead-characters (e. g., “broad-leafed trees”), etc. Not all subkeys may be available in all desirable scopes. For example, the entry key may be available for the preferred geographic scope, but certain family or genus keys may be available only on a different scope. If, for example, the only available keys are global (being large, unwieldy, perhaps outdated) and a key scoped to a neighboring country, it will be desirable to leave the choice of subkey to the user. Similarly, within a general moth key, the identification certainty of some moth genera may profit from using an anatomical key (requiring a microscope), but a less reliable “field key” may be provided as an alternative.

Multiple keys at a result-lead may either be implemented through the “result-and-continue pattern” (Fig. 148 left; compare p. 235) or by allowing more than one subkey pointer at each result (Fig. 148 right).

“Vegetative key to monocotyledonous families”

[...]

8. [lead text ...] **Sedges (Cyperaceae)**
 9. Flowers or seeds present **Cyperaceae**
 – Only vegetative material present
 **Vegetative key to Cyperaceae**

“Vegetative key to monocotyledonous families”

[...]

8. [lead text ...].....
 – **Cyperaceae (flowers or seeds required)**
 – **Vegetative key to Cyperaceae**

Figure 148. Multiple subkeys may follow after a single lead. Left: using a “result-and-continue pattern”; right: using a list of available keys.

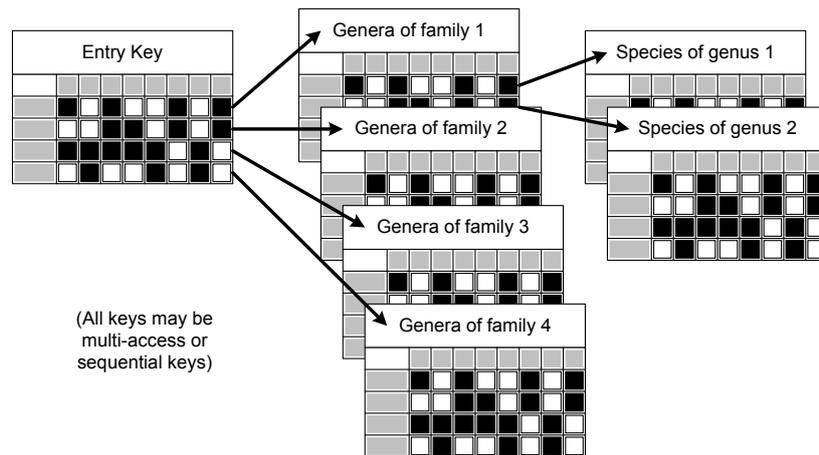


Figure 149. Hierarchically linked keys (visualized as a character state \times entity matrix representing multi-access keys).

Transferring progress information between multi-access keys

To improve the experience of linking keys, it is desirable to transfer information from one key to the next. When dealing with well-designed branching keys arranged in a key hierarchy following the taxonomic hierarchy (family, genus key, etc.), the identification progress used in the higher key will be transferred to a large extent simply by the design of the higher-level key. Although even for branching keys occasionally some information may be lost (especially where the lower taxon is variable with respect to a character), in general this is negligible.

The situation is markedly different for multi-access keys. Here it is quite common that the user records data on convenient characters that remain polymorphic in the following subkey. If in a computer-aided multi-access process these characters are already scored, it is desirable to transfer the information to the next key (Fig. 150).

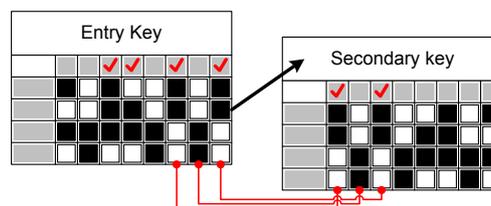


Figure 150. Scores recorded during identification may be transferred from one key to the other where linked keys share common characters.

The simplest form of integrating multiple multi-access keys is to base them on a single data matrix, where for certain taxon groups only a subset of characters (general plus group-specific) is scored (Fig. 151). To structure this process and improve documentation and communication among collaborators, SDD has a special coding status “not to be coded” (Table 16, p. 75), which in the presence of data inheritance (p. 99) may be deduced down the taxonomic tree. The concept of “Progressive revelation” (see p. 270) may then be supported by such explicitly marked or detectably incompletely filled matrices.

The single-matrix approach is used, e. g., in the LIAS project where family keys are handled through taxon and character subset views on a common base matrix. In DiversityDescriptions such sub-projects are implemented as database views and can be used identically to projects based on their own tables (Fig. 152). The characters and items in a subset are adjacent in Fig. 152 for illustration purposes alone, compare Fig. 226 (p. 334) for an illustration of non-adjacent subsets.

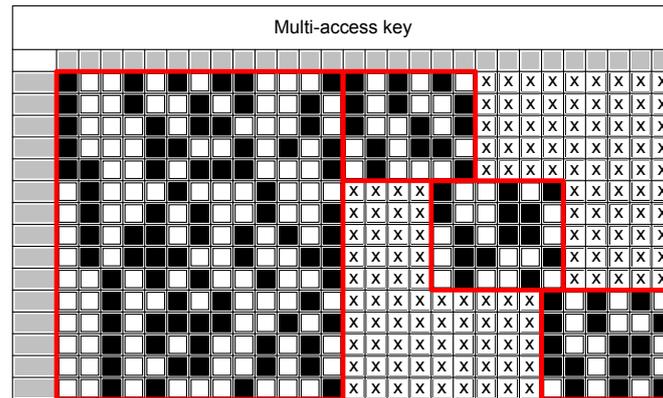


Figure 151. Multi-access key (visualized as a character state \times entity matrix). The sub matrices designated for coding are marked with a thick border; characters outside may have been marked with “Not to be coded” coding status value (symbolized here by an ‘x’).

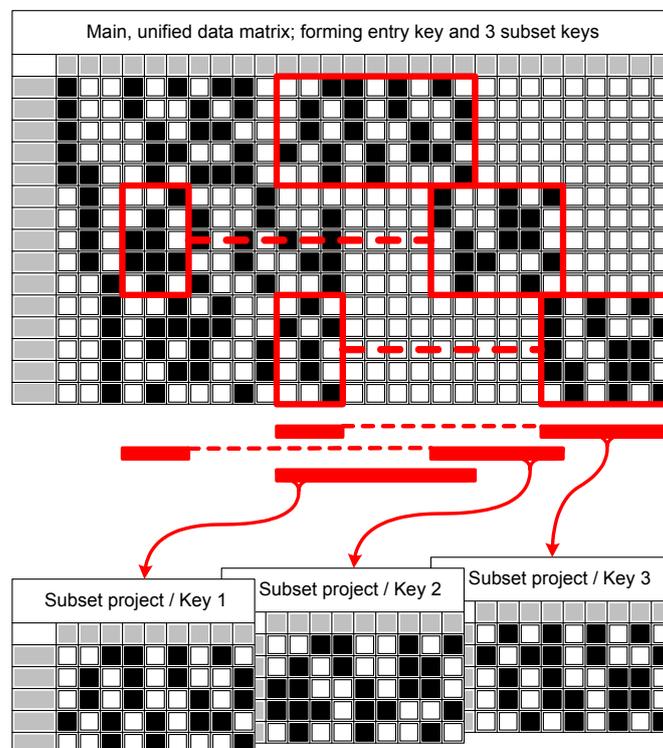


Figure 152. Multi-access key (visualized as a character state \times entity matrix). The areas that are relevant to subgroups can be described as item and character subsets. Database views may be used such that from the user’s standpoint subset projects are indistinguishable from a full set.

In practice, the single matrix approach has serious limitations. The requirement to manage all data in a single coherent matrix makes projects inflexible both technically and socially. Furthermore, very large character sets become quickly difficult to manage.

It is therefore desirable to also offer linking mechanisms between independently developed keys and coded descriptive data (i. e. data matrices). A simple linking is trivial and may simply be a resolvable URI. However, carrying information from one key to another is a serious problem. It either requires both projects to use a common standard for taxa and descriptive terminology (perhaps through some form of taxon, character or state globally unique IDs) or to provide external knowledge about which characters and states are strictly or approximately comparable. These

problems are discussed in more detail in “Federation and modularization of terminology” (p. 180).

Current software examples: CSIRO Intkey offers two generic mechanisms (*Taxon links* directive and *File Display* command) to link to other resources, including between multiple independent keys.

DiversityDescriptions currently supports a subset approach with virtual “view projects”, but no specific linking mechanisms between independent projects. Similar to the Intkey approach, external keys are considered resources and use the same mechanism as images, audio/video files or formatted documents. Resources may be available locally (e. g., for CD-based keys) or on the internet. If multiple identification key resources are present, the display priority of these may be defined. The data structures to link between resources are provided (see p. 322 ff), but the current version of the application supports resources only in a minimalistic way. An improved version of the Identify component that actually exploits the possibilities of the information model is planned for the future.

CBIT Lucid (p. 21) can link multiple matrixes for particular taxa, resulting in a hierarchy of data sets and corresponding keys (pers. comm. K. Thiele). It is thus possible to create a system of primary and subkeys similar to those found in conventional faunas and floras. Identification progress is not carried to subsequent keys.

219. Both branching and multi-access keys may point to another key (subkey) rather than to a taxon. Subkeys often are associated with taxonomic ranks (order family, genus keys) but may be dominated by other scopes (e. g., “shrubs in winter”, “broad-leafed trees”).
220. More than one subkey may be desirable at a single result-lead.
221. Descriptive data for multiple algorithmically created keys (branching or multi-access) may be kept in a unified matrix; in addition support for pointers from one key to independently developed related keys is required.

Transferring progress information between branching and multi-access keys

Considering the relative advantages of branching keys versus multi-access keys, it seems desirable to offer the user a choice of identification interfaces wherever suitable coded descriptive data and key-generation algorithms are available. Under these circumstances, it seems highly desirable to be able to switch between interfaces, preserving as much information as possible while switching.

Such a system would allow starting in a branching key, switching to another branching key (e. g., from flower based key to vegetative key), and finally finishing identification in a multi-access key. Branching keys are often desirable starting points because of the carefully balanced guidance they offer. However, at some point, a branching key may become inconvenient or inefficient (e. g., microscopical information requiring thin-sections) or impossible to follow (e. g., fruit characters when no fruits are present in the material).

Conversely, if users are starting with a multi-access key, but have the impression that little progress is made, they may decide to switch to a branching key. Although a branching key is less flexible than a multi-access key, still transferred information could be utilized by skipping couplets and leads in a branching key that are known to be inapplicable. In the example shown in Fig. 153, the branching key could start directly at question 3 (character 3, state 1 or 2) if a user had scored the characters shown.

Because of the different structure of branching and multi-access keys, transferring information is more difficult than transfer among multi-access keys. To some extent, reasoning may be based on excluded taxon sets. When switching from a branching key to a multi-access key, the set of taxa excluded in the branching key may be treated as a separate kind of criterion and displayed as

such in the history of identification steps. Conversely, when switching to a branching key, a lead where all resulting taxa are excluded based on information from the multi-access key may be excluded, and if only a single lead remains the couplet may be skipped entirely. An example of this “pruning method” can be seen at National Herbarium of New South Wales (2007).

A more flexible method of information transfer requires annotating the branching key in a format that is equivalent to the character states or values used in the coded descriptive data on which the multi-access key is based. Because branching key leads commonly contain complex propositions, such a “markup” requires support for the full range of Boolean operators and nesting (brackets). Upon transfer to a multi-access key, for each selected lead in the branching key multiple states of multiple characters may be scored (Fig. 153). For example, the leads in a couplet may be “Flowers yellow and stamens 5” ↔ “Flowers white or stamens 20” (i. e., “Flowers white (and stamens 5) or (flowers yellow) and stamens 20”). The support of the ‘not’-operator may be desirable, although this can also be translated into a complement-statement using all other states connected with ‘or’. The advantage of supporting ‘not’ would be that the annotation-statement in the branching key is more likely to remain correct if an additional state is introduced in the character terminology.

The markup of authored branching keys with coded descriptive information is a relatively expensive process. It may be desirable, however, since this process automatically creates a sparsely filled data matrix which may then be used as a template for a more completely filled and revised data matrix. Furthermore, the addition of coded information could occur automatically in programs that support semi-automatic creation of branching keys.

Linking branching, authored keys with a descriptive data matrix may further improve the handling of incompletely translated branching keys. Instead of reverting to default and then available languages, the application may offer the option to switch into a multi-access mode at this point. Translating only the “head” or “start” of an authored branching key into multiple languages is often more practical and brings the greatest benefit in guiding users to be relatively close to successful identification.

To the author's knowledge, no current software program implements a tight integration of branching keys with multi-access keys. Several programs, however, offer character guidance mechanisms, which in the case of CBIT Lucid “expert routes” (p. 269) comes close to an integration of a branching key into a multi-access key.

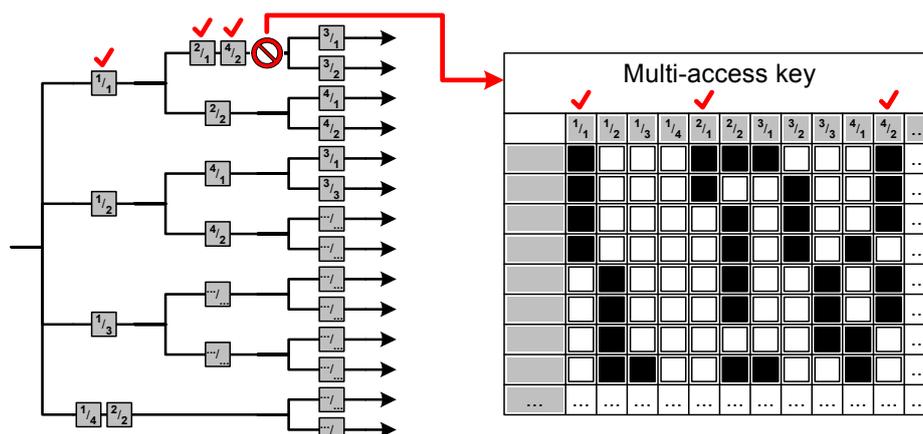


Figure 153. A branching key (left) connected with a multi-access key (visualized as a character state × entity matrix, right). Questions and matrix columns are labeled as character number/state number. At some point in the branching key the user cannot answer the next question (character 3, state 1 or 2) and switches to the multi-access key, transferring the information entered so far. Alternative characters (not shown) may then help to finish the identification. The illustration is a simplification insofar as character and states in the branching key may be connected by ‘or’, ‘and’, and ‘not’.

- 222. Support for transferring information between branching and multi-access keys is desirable.
- 223. Optional support for coded data reflecting the proposition made in the lead of a branching key is desirable. This may have the form of markup of the natural language lead text or of a coded description associated with a lead.
- 224. Support for Boolean operators and nesting is required for both alternatives (markup of natural language lead text or associated coded descriptions).

5.6. Equality criteria and error tolerance

A major question in regard to identification processes is how they respond to

- errors made in data input during identification (by a human user in interactive keys, or as a result of automated processes),
- errors in the knowledge base of descriptions,
- indications of uncertainty added during the identification process,
- indications of uncertainty added in the knowledge base, and
- missing data (completely missing, or annotated using coding status values, see p. 74) in the knowledge base of descriptions.

Such error tolerance (often also called “graceful degradation”, Pankhurst 1993a) may be achieved by using fundamentally error-tolerant methods. Examples are:

- Similarity methods using multivariate statistics (e. g., the MIS system, MIDI (undated), uses Principal Component Analysis);
- probabilistic (maximum likelihood, Bayesian statistics, etc.; see, e. g., Pankhurst 1991 and Fortuner 1993);
- fuzzy logic; the only examples in biological identification known to the author are Kennedy & Spooner (1994) and Winder & al. (1997), but see, e. g., Anagnostopoulos & al. (2003) for human face identification;
- machine learning such as artificial neural networks (e. g., Clark & Warwick 1998, Clark 2003) and other pattern recognition techniques (e. g., Agarwal & al. 2006).

Most current identification keys, however, follow a model of Boolean predicate logic, the logic used in standard database queries and most programming languages. These methods are ideal for exact, non-error-tolerant behavior, but additional methods or precautions for error tolerance may be built in. One may distinguish between:

1) Error tolerance embedded in descriptive data

In natural language and coded data, it may be helpful to include indications of uncertainty or doubt in the descriptive data; this is discussed in detail in “Certainty modifiers” (p. 207).

An extreme form of supporting error tolerance through data is to include statements that are factually false, but often erroneously assumed to be true. Without further precautions, it becomes quickly impossible to distinguish between purposely and accidentally erroneous data. A special form of misinterpretation markers is therefore desirable; compare “Misinterpretation hints through modifiers” (p. 209).

A special method applicable to authored branching keys is that a key may include redirections (reticulations Fig. 116, p. 235) or taxa are keyed out multiple times. This may compensate for either natural variability or frequent misinterpretations of structures or properties.

2) Algorithmic error tolerance in interactive identification

To achieve algorithmic error tolerance, the criteria for equality between query criteria and search results must be modified relative to a standard database query. When comparing values for a given set of variables, the following equality criteria are almost universally accepted:

- Two value sets match if all variable values match (Fig. 154 a).
- Two value sets match if all values defined in the search criteria match the values in the corresponding variables in the knowledge base (Fig. 154 c).
- Two value sets do not match if at least one variable defined in the search criteria does not match the corresponding variables in the knowledge base (Fig. 154 b).

One method of improving error tolerance is to modify the third criterion. Instead of returning only perfectly fulfilled conditions, the algorithm may return a list of results ranked by the number of character variables matching may be returned. An identification program may trigger this automatically if no perfect match could be found, or it may offer an option to explicitly request “similar results”. In Morse (1974) a “variability limit” defines the number of characters which may be in contradiction (not matching).

Whether variables defined in the search criteria are considered to match with missing data in the knowledge base is more contentious (Fig. 154 d). Standard database queries will consider a search clause “Where X=1” to not match a database record with “X=NULL”. During identification this behavior is highly desirable – it would allow identification only if all data are known for all taxa – and probably all identification programs will consider criteria for a character to match all taxa with missing data in this character.

A second analysis of equality criteria may focus on the equality of variables where the values are sets (i. e., the value of a character is a set of atomic values or states). This is the case in most categorical characters in biology. Whether perfect identity (Fig. 154 e), a subset of values (Fig. 154 f), or a non-empty intersection (“overlap”, Fig. 154 g) is required in a search condition varies. Most standard database queries will consider the subset to be the appropriate matching conditions (e. g., searching for Author1, Autor2 will return all titles containing both these authors plus additional authors). For identification processes, the intersection condition is, however, more appropriate. With regard to the information model, it is likely that the difference is best handled either outside of the information model, or in context-specific metadata.

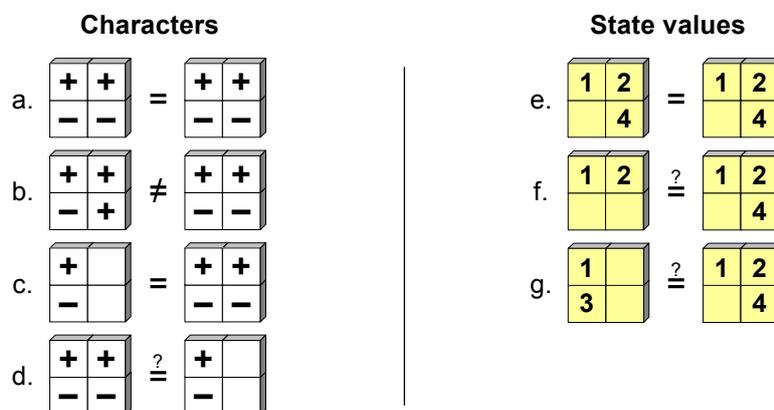


Figure 154. Illustration of equality criteria that are either generally accepted or special to some identification use cases (shown with ‘?’ above equal sign). The left side (a-d) illustrates criteria on the character (variable) level, the right side (e-f) criteria on the character state (value) level. On each side, search criteria are shown to the left, values in the database to the right of the comparison operator. Missing data for a character is shown as an empty box. For example, f. illustrates that in an identification use case it is usually desirable to return descriptions that have three out of four states, even if only two of these can be found in the individual being identified. See text for further explanations.

The criteria discussed so far address only categorical data. For quantitative data similar arguments concerning included or overlapping ranges can be made. In standard statistical techniques, any overlap between two confidence intervals is considered a match. If the knowledge base con-

tains only single values, a comparison is especially problematic. A common option is to add an artificial margin of error to both values. Examples are the Absolute and Percent Error DELTA directives used in the Confor/Intkey programs (see Macfarlane 1993b) and the user-settable error tolerance in DiversityDescriptions (see Fig. 155). Although typically used for quantitative data, the “margin of error” method may also be applied to ordinal categorical data, where adjacent ordinal values may be considered matching (especially if the initial identification failed and similar descriptions are sought).

It is often desirable to offer different query modes for identification purposes and exact retrieval of existing data for editorial or data analysis purposes. In the CSIRO Intkey program, several of the contentious equality parameters are parameterized. For characters the behavior in regard to individual coding status value may be set (Match ‘U’ and ‘I’ in Table 55), and for character values all three equality options shown in Fig. 154 can be selected (‘E’, ‘S’, ‘O’ in Table 55).

The Identify component of DiversityDescriptions/DiversityDescriptions is less flexible than Intkey; it only distinguishes between data retrieval (exact) and identification mode (error-tolerant).

Options for error tolerance may be activated by degree. For example, to accelerate the identification process an expert may choose the standard matching conditions and exclude data containing misinterpretation modifiers.

If error tolerance is activated, it is desirable that a program is able to report which identification results (remaining taxa) are resulting from applying error tolerance (e. g., misinterpretation hints, or one or few characters in contradiction). Furthermore, the character affected for each such taxon may be of interest. This may occur by highlighting result taxa or characters (e. g., through color or icons), or by presenting results based on misinterpretations in a separate list.

Figure 155. Available options in *Identify* (*DiversityDescriptions*) for quantitative measurements obtained from the object to be identified. The options support the identification algorithm in selecting appropriate margins of error. Compare also Fig. 136, p. 253.

Table 55. Character equality parameters (“Set Match”) in CSIRO Intkey.

SET MATCH	Description in Dallwitz & al. (2000b)	Example in predicate logic ⁽³⁾
O ⁽¹⁾	“specifies that two sets of values match if they overlap, that is, if they have any values in common (e. g., 1/2 matches 2/3; 2–5 matches 4–10)” ⁽²⁾	Character=“FlowerColor” AND (State=“blue” OR State=“violet”)
S ⁽¹⁾	“specifies that two sets of values match if one set (usually the values of the specimen) is a subset of the other (e. g., 1/2 matches 1/2/4 but not 2/3; 2–5 matches 1–6 but not 4–10).”	Character=“FlowerColor” AND (State=“blue” AND State=“violet”)
E ⁽¹⁾	“specifies that two sets of values match only if they are identical”	Character=“FlowerColor” AND (State=“blue” AND State=“violet”) AND Count(*)=2
U	“specifies that ‘unknown’ matches any value”	Character=“FlowerColor” AND (Status=“Unknown”)
I	“specifies that ‘inapplicable’ matches any value”	Character=“FlowerColor” AND (Status=“Inapplicable”)
I O U	“default setting for identifications” (for data retrieval O is recommended)	Character=“FlowerColor” AND (State=“blue” OR State=“violet” OR Status=“Unknown” OR Status=“Inapplicable”)

⁽¹⁾ S, O, and E are alternatives and may be combined with U or I.

⁽²⁾ The cited text from Dallwitz & al. (2000b) uses DELTA notation, “1/2 matches 2/3” may be read as “state 1 or 2” matches “state 2 or 3”.

⁽³⁾ The predicate logic examples are written similar to SQL. However the AND clause in Match S would refer to two Character and two Value fields in an inner join (assuming states are saved as records, similar to DiversityDescriptions), and the Count clause in Match E may require a sub-query or another join.

3) Algorithmic error tolerance in automated identification

Wherever complex quantitative data are used in automated identification (p. 231), specialized equality criteria and error tolerance algorithms need to be employed. In many cases the equality criteria will be based on probabilistic or similarity measures, returning result sets with an increasing probability that the true result is among the set. This can be measured using test data sets and test objects to study the effectiveness and quality of the identification algorithms. One measure for this is the “Receiver Operating Characteristics (ROC)” used in Agarwal & al. (2006).

5.7. Character ranking and guidance

In many identification processes, a choice of the next characters is required. In interactive identification scenarios, a human user may request a suggestion about the next character to study, or a sorting of all characters by some measure expressing decreasing utility. Various optimality criteria exist, prefer, e. g.,

- characters that are simple and quick to measure or score (especially in the field),
- characters that are highly available in all kind of material and at most times (e. g., seasons), or
- characters that are expected to provide fastest progress (e. g., those which on average result in the smallest number of remaining taxa).

Interactive scenarios in which a human may request such guidance are:

- in a multi-access key the user may request a suggestion which characters to use next,
- the creator of a branching key requests a recommendation which characters to consider in the next step in the new key, or
- in the confirmation phase of an identification (see p. 232), “check characters” or diagnostic descriptions may be requested for taxa that are similar to the initially identified one(s).

Some closely related use cases exist, in which character selection becomes automatic by simply accepting all characters recommended based on the optimality criterion:

- during automated creation of a branching key, the recommended characters are automatically accepted,
- in the creation of diagnostic descriptions (p. 39) the optimal character set is sought.

Importantly, the criterion of “fastest progress” depends on a current taxon set, which may continuously change during identification progress. For example, in the construction of a branching key after the first couplet, the optimality of the next character is best based on the remaining taxa in either lead.

Character ranking and guidance may be authored or it may be calculated on the basis of algorithmic optimality criteria.

Authored character guidance

Authored character guidance information may be stored as **character-ranking metadata**, **best path**, or **coding status** information.

Character-ranking metadata (“character ratings”) may be used to order characters in a single sequence of characters based on how much they are recommended in general for identification. Examples are the *weight* and *reliability* directives in DELTA (p. 19). A problem when exchanging DELTA data containing such information is that neither *weight* nor *reliability* have a strict semantic definition. “Weight” is intended for general purposes, including especially phylogenetic analyses, and reliability for identification purposes (M. Dallwitz, pers. comm.). Two different convertible value scales exist in DELTA, and these have no interoperable definition. The broad definitions that DELTA offers for weight and reliability provide flexibility for a variety of purposes, but require external free-form text documentation of the implied semantics. Common usage when using *weight* or *reliability* for character guidance is to adjust the values until the

desired effect is achieved in a specific target application (such as CSIRO Intkey, CSIRO DELTA Key, Pankey, or PAUP).

At the SDD meeting in Brazil (see minutes, Hagedorn 2003a), a general consensus existed that interoperable and semantically defined weighting or rating parameters with an interoperable scale should be introduced. This should make it possible to obtain the values in a “questionnaire” style from biologists unacquainted with actual processing software, and to exchange data independently of a specific application or purpose. The SDD discussions are summarized in Hagedorn (2005a). Earlier proposals for a richer character rating vocabulary are: Diederich & al. (1989) and Diederich & Milton (1991), proposing *conspicuity*, *ambiguity*, and *variability*, and Hagedorn (1999a), proposing *availability* and *reliability*. The current SDD schema 1.1 contains the following enumerated values:

- *ObservationConvenience*: How conveniently can a character be observed? This may include a measure of cost of equipment and expendables (such as chemical reagents). Convenience should be rated relative to other methods required for identifications within a taxonomic group. Thus, for example, if microscopic methods are always necessary in a taxonomic group, microscopic characters may be considered convenient within this group. Also, a character may be convenient in one group, but inconvenient in another.
- *Availability*: How available is the character or concept for identification? For example, ratings would be low if a character is available only during a short time in the life of an object, or only expressed with low frequency in populations.
- *Repeatability*: How reliable and consistent are repeated measurements or scorings of the character by different observers and on different objects? This may include both variability of values (frequency of polymorphisms) and variability in how the observations are interpreted. It depends both on precision (quality of being reproducible) and accuracy (closeness to the true value).
- *PhylogeneticWeighting*: A weighting factor expressing the relative weight of a character for the purpose of phylogenetic analysis.
- *RequiredExpertise*: The user is expected to have this expertise level at least.

An unsolved problem with such ratings is that for some it may be desirable to define different ratings for field or laboratory identification. A flexible mechanism is desirable, guaranteeing interoperability for some rating semantics but providing for user-defined extension or subclassing. This mechanism has not been fully developed.

In addition to such interoperable character-ranking metadata, application-specific extensions may also be desirable. For example, one may want to fine-tune the ranking of characters when producing printable branching keys, without unnecessarily “twisting” the general metadata. Application-specific ranking metadata may be used instead of, or in combination with, interoperable metadata. In SDD the general application-specific extensions (“CustomExtensions”) are available in all objects. These extensions document the target application in a required name attribute and allow any application-defined schema inside.

Irrespective of the exact structure of character-ranking metadata, a fundamental problem is that the rating value often depends on a taxonomic group (Hall 1970). Counting the number of anthers in flowers may be convenient in large-flowered groups, but highly inconvenient in wind-pollinated groups with tiny and reduced flowers. To achieve character-ranking metadata scoped within the taxonomic hierarchy, these may be considered a special form of data in a character \times taxon matrix (instead of considering them part of character definitions). Combined with a concept of data inheritance down the taxonomic tree, general metadata may be defined in the root taxon, inherited by all other taxa until (e. g., in a specific family) rating values are adjusted. This process may occur for individual characters, i. e., only the necessary characters need to be adjusted.

Dallwitz & al. (2006) and Dallwitz (2005b) propose for “New DELTA” (p. 20) to introduce *attribute reliabilities* (defined within character data) in addition to DELTA’s *character reliabilities* (which are part of the character definitions). An attribute in the sense used in DELTA (com-

pare Table 3, p. 34) is the set of all states or values for a specific character in a description, i. e., a character value in the terminology used here. If the proposal is understood as a modifier of character data (see below), inheritance along the taxonomic hierarchy would be bound to value inheritance, which may be undesirable. Indeed, the *attribute reliability* proposal is intended to be independent of character data (M. Dallwitz, pers. comm.).

Convenience, availability, and required expertise of a character normally do not depend on the result of a measurement or observation. Exceptions do occur, e. g., size measurement may require different instruments for very small (microscope), medium (ruler), or very large objects (e. g., a tree measured using a hypsometer, clinometer, relascope). Similarly, certain colors or shapes may be easier to record than others. Furthermore, repeatability of a measurement may depend on results. Therefore, a modifier-like approach that modifies existing values may indeed be desirable in addition to a general mechanism (compare “Reliability modifiers”, p. 213).

The concept of character-ranking metadata (“character ratings”) is further discussed in an SDD proposal (Hagedorn 2005a).

An alternative to character-ranking metadata is the “**best path**” (or “best route”, “expert route”) approach. Here, a tree-like data structure stores a sequence of characters that depends on the state selected in the previous character. The necessary data structure is very similar to a polytomous branching key, where each couplet uses only a single character. In contrast to branching keys all or most terminal leads carry a special “continue in multi-access mode” message rather than leading to taxa or further keys.

In principle any branching key contains guidance about the sequence in which characters are recommended to be used for identification. Extracting “best path” information from a branching key to make it reusable (e. g., in a multi-access key), is trivial if the branching key uses only one character per couplet, slightly more difficult where multiple characters combined with ‘*and*’ are used, and may be quite complicated if the full range of Boolean operators is used.

The closeness to a single-character polytomous key is attractive because in many cases character guidance information can be obtained directly from published literature. On the other hand, the approach can become inflexible if a data set undergoes continuous editing and revision. Special problems may arise if a data set integrates data from a wide spectrum of taxonomic groups, obtained from different sources.

The advantage of a best path over a branching key is that an application may allow users to score other characters before entering the best path and after leaving it. Based on the sets of remaining taxa, identification software may be able to skip recommended characters that are redundant based on information entered before entering the “path”. Even if the user decides to skip a character, the software may continue with recommendations, now resulting in multiple characters rather than a single one (compare CBIT Lucid Phoenix, Fig. 128, p. 249).

A “best path” approach is implemented as *expert routes* in CBIT Lucid version 2 (CBIT 2004). Expert routes are labeled, and character states may be scored because information is implied in the route label. Lucid expert routes differ from branching keys in that they contain a provision to include quantitative characters (which then have no effect on the subsequent route). At the moment, the feature is limited to the older Lucid 2 version; the reimplementations in Lucid3 is currently not yet accomplished (K. Thiele, pers. comm.).

Finally, **coding status** information (see p. 74 for further information) stored in the descriptive data matrix may be used to some extent to provide character guidance. Taxa where a given character in the data matrix has certain coding status values (especially the implicit value “no data recorded” and the explicit values “Data unavailable”, “Not interpretable”, “Data withheld”, and “Not to be coded”, but not “Not applicable”) should match any actual character value observed in the object to be identified. This feature is called the “retaining unknowns” feature in Dallwitz (2005b) and Dallwitz & al. (2006). If the object to be identified has three wing spots and no num-

ber of wing spots is recorded for certain taxa, it cannot be excluded that the object belongs to these taxa.

As a result of this, the higher the proportion of these coding status values among the remaining taxa for a given character is, the less separating power does this character have and the less recommended it is. Note that this information is not a sufficient character recommendation in itself. It is valuable to recognize characters that should be avoided, but of little value to rank those characters where the matrix is reasonably complete.

Related to character guidance using coding status is the concept of “**Progressive revelation**”, proposed by K. Thiele (pers. communications and emails to TDWG-SDD discussion list on 2000-02-28). The general “matrix paradigm” assumes that, because of the huge benefits of multi-access identification, it is valuable to record character data beyond the minimum diagnostic set. However, this cannot be justified for specialized (i. e., “apomorphous”) characters that are required within small taxon groups, but not elsewhere. If no data exists in the literature beyond a small taxonomic group, it may be considered a waste of resources to collect data for the majority of taxa in which nobody so far has found these character data necessary. The scenario of a key with progressive revelation is that such characters are initially hidden from users of multi-access keys and appear only once identification has progressed to a group (e. g., a genus) in which they are desirable. The “progressive revelation” concept is closely related to a general user interface design concept called “*progressive disclosure*” (Mandel 1997).

Several models can be designed to handle the data required for this mechanism. However, any such model will essentially be based on a character \times taxon matrix. Hagedorn (2004b) therefore proposes to use the mechanism of coding status values that is already available. Although progressive revelation would already work with a generic coding status (such as the DELTA “unknown”), a special coding status “not to be coded” (Table 16, p. 75) is proposed for SDD to improve documentation and communication among collaborators. The use of coding status values is especially effective if the data matrix supports data inheritance down the taxonomic hierarchy. It is then sufficient to code a character a single time “not to be coded” in the root taxon description, and override this status as soon as the group in which it becomes relevant is reached.

- 225. Character-ranking metadata, expressing various optimality criteria for identification or analysis, are important data elements.
- 226. Character-ranking metadata should be flexible to support various ranking categories (or “topics”).
- 227. Both interoperable and application-specific ranking metadata may be required.
- 228. Support for missing data in general, and specifically coding status “not to be coded”, is desirable to support guidance in character selection. This supports requirement 50, p. 76.

Algorithmic character guidance

Ranking categorical characters according to separating power

Analysis of separating power is based on character usage in a set of taxon or object descriptions. Most current algorithms are in principle based on average information content of a character (i. e., *entropy* of a message set). Examples are the “Best” algorithm in CSIRO Intkey (Dallwitz & al. 2000b), “Best” in Pankey Onlin7 (p. 19) and “dynamic optimization” in Visual Key (Klimov 2001).

For a categorical character with s states, where each state occurs with a frequency p_i in the descriptions, the average information content is:

$$\begin{aligned}
 H &= \sum_{i=1}^s p_i \cdot \log_2 \left(\frac{1}{p_i} \right) \\
 &= - \sum_{i=1}^s p_i \cdot \log_2 (p_i)
 \end{aligned}
 \tag{1}$$

The values of p_i are calculated as the number n_i of descriptions remaining after selecting the i^{th} state, divided by the number n of descriptions that are current before selecting this state. The “current set of descriptions” may be the result of previous identification steps involving character data, or of scoping conditions such as selection of a taxonomic group or geographic region.

The value of H decreases with the number of states and with decreasing evenness of their distribution (Fig. 156). The highest value is achieved with uniformly even distribution, i. e., all states occur with the same frequency. The information content of characters with many states is higher than the content of characters with two states, but evenness is even more important (see examples in Table 56).

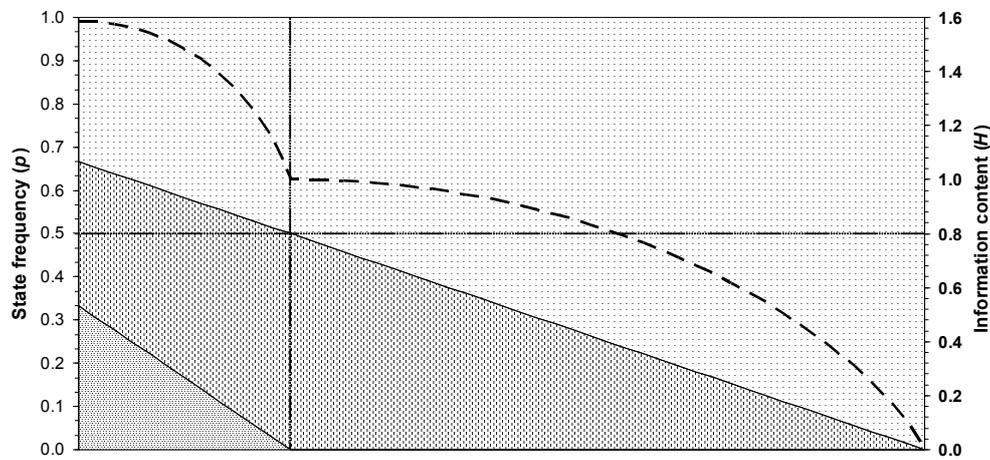


Figure 156. Effect of unevenness on H values. On the x-axis increasing proportions for the top-most state are plotted. The diagram combines possible frequency distributions for three and two states. Starting with an evenly distributed three-state-character, two states increase their frequency at the expense of the third until becoming an evenly distributed two-state-character (inner dotted grid lines). The frequency of the topmost state is further increased until it is the only remaining state. The cumulative frequency of states (shaded areas) is plotted against the left axis, information content (H -values) against the right axis (dashed line).

The importance of evenness is best illustrated in an example. If a large data set contains $n = 1$ million taxon descriptions and all characters evenly halve the number of taxa remaining, each identification would require about 20 identification steps (i. e. $\text{ceiling}(\log_2(n))$). If, however, each character would split the results in a 0.2:0.8 proportion, identification would minimally be completed in 9 steps, but maximally require 62 steps (i. e. $\text{ceiling}(\log_{\text{base}}(n))$, with $\text{base} = (0.2)^{-1}$ and $(0.8)^{-1}$, respectively). The exact average length of all identification paths is difficult to calculate (partly because of the ceiling function, i. e., rounding to next higher integer). However, it is much higher than $40 = (9+62)/2$, because the larger proportion of taxa always remains in the longer identification path.

Sorting the characters using Equation 1 thus ranks the characters based on how fast identification would advance.

Table 56. Examples of values of information content H (Equation 1) for binary and multistate characters, without and with overlap among state frequencies.

States (s)	2	2	3	3	4	2	2	3	4
Frequencies (p_i)	0.5	0.9	1/3	0.9	4 ×	0.6	0.9	3 ×	4 ×
	0.5	0.1	1/3	0.05	0.25	0.6	0.9	0.5	0.5
			1/3	0.05					
Overlap	No	No	No	No	No	Yes	Yes	Yes	Yes
Information (H)	1.00	0.47	1.58	0.57	2.00	0.88	0.27	1.50	2.00

In practice, the frequency of a state will often be 0 and the term $p_i \cdot \log_2(1/p_i)$ becomes “ $0 \cdot \log_2(1/0)$ ”, in which $1/0$ is undefined. It is convenient to handle this by defining a precedence for the multiplication with 0, resulting in the expected value of $H = 0$ (a state with no taxa cannot contribute to identification progress). However, depending on the software used, Equation 1 may have to be calculated using case logic (Equation 2). This problem will no longer be shown in the following equations.

$$H = - \sum_{i=1}^s \begin{cases} 0 & \text{if } p_i = 0 \\ p_i \cdot \log_2(p_i) & \text{otherwise} \end{cases} \quad (2)$$

Overlap and missing data

Equation 1 was originally developed in an information messaging context where different signals are always strict alternatives. In the practice of biological descriptions, a single taxon may express multiple states, either because of variability between individuals of a taxon, or because of variability and other situations that may occur within individuals (compare Fig. 33, p. 94). As a result, the sum of state frequencies calculated as (*descriptions containing state*)/(*total number of descriptions*) may be > 1 .

Furthermore, missing data effectively also create an overlap. As discussed in “Equality criteria and error tolerance” (p. 264), the true value of unscored characters may be any state. Thus, the set of descriptions with missing data for a character (containing no information at all, coded only with explicit coding status values, or coded with a coding status value plus character state data) must be added to each state-specific result set (Table 57, last column).

Table 57. Example calculation for overlap between result sets involving missing data. The result frequencies of states (p_1, p_2) already overlap with each other or with the set of descriptions containing a coding status value (p_c). The sum of frequencies is further increased by adding the set with missing data to each state set.

Description	Variable	Measured p	Calculation	Final p
Neither character data nor coding status recorded	$p_u =$	0.1		
Coding status: “Data unavailable”, “Data withheld”, “Not to be coded”, etc.	$p_c =$	0.2	$p_m = p_u + p_c =$	0.3
State 1	$p_1 =$	0.3	$p_1' = p_1 + p_m =$	0.6
State 2	$p_2 =$	0.5	$p_2' = p_2 + p_m =$	0.8
Sum-of-frequencies:		1.1		1.7

If overlapping p -values are allowed (Fig. 157), Equation 1 only partly behaves in a desirable way (Table 56, right). For characters with two states, characters with no overlap between states are always ranked higher than those with overlap. However, characters with more states are ranked, even with overlap, better than characters providing the same partitioning with fewer states, but

overlap. Clearly, the two-state character partitioning the set in half is preferable to those with three or four states at $p = 0.5$, but the latter are preferred in the ranking. Furthermore, for characters with many states, increasing overlap initially increases the value for H . The effect is minimal for characters with three states, but highly relevant for characters with four or more states (Fig. 158).

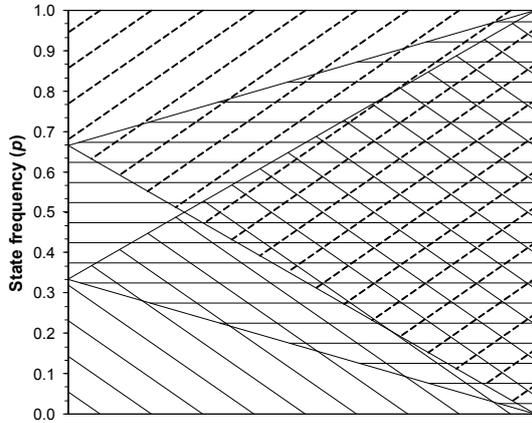


Figure 157. Area diagram illustrating three states with 0 to 100% overlap. Frequencies remain evenly distributed throughout.

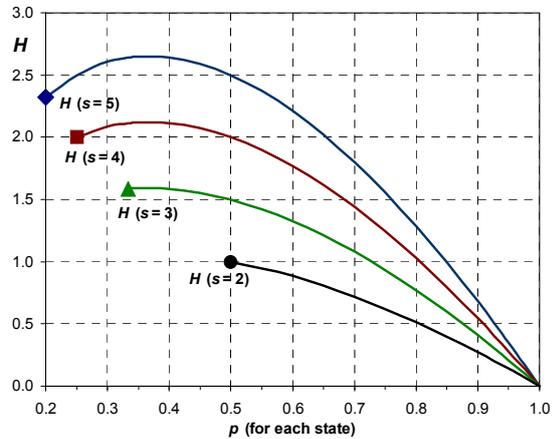


Figure 158. Effect of overlapping p -values on H for 2 to 5 states (s) with equal frequencies (p). Each line starts at zero overlap with overlap increasing to the right (compare Fig. 157 for $s = 3$).

One conceivable approach to reduce the effect of overlap is to separately calculate a frequency p_m (m for *missing*, i. e., descriptions where the character is not coded or where it has an equivalent coding status). All other state frequencies are then calculated as if the descriptions with overlap or missing data would not exist, i. e., they are based on $1 - p_m$ and do not change with increasing overlap. To discourage the presence of missing data, H may then be multiplied with $(1 - p_m)$. A similar proposal (albeit not for an entropy function) was made by Pankhurst (1991). The result is a collection “entropy” functions linearly decreasing from an initial H in direct proportion to the increasing frequency of descriptions lacking data for a character (Fig. 159).

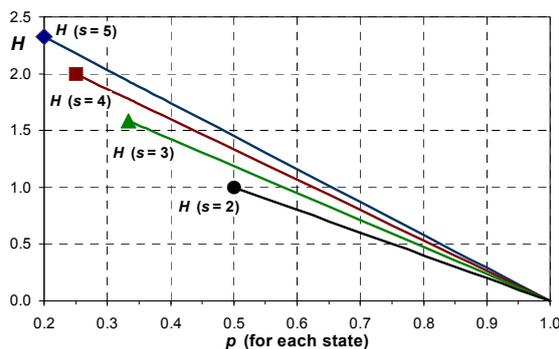


Figure 159. Linear correction for missing data by calculating H for coded data and multiplying with proportion of data coded ($1 - p_m$). This is comparable to Fig. 158, assuming all overlap is a result of missing data.

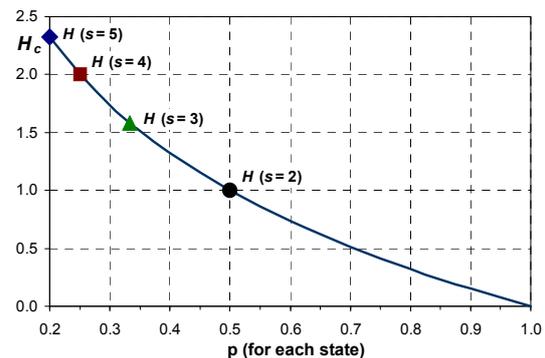


Figure 160. Entropy function H_c , correcting overlap by dividing through sum-of-frequencies for even state-distribution. Compare Fig. 158 for uncorrected values.

The p_m -method described above addresses only the problem of missing data. In a deduction from first principles, considering the average length of a key, Dallwitz (1974) arrived at a new equation. For the case of neither character nor item abundance weighting, this “BEST” function can be rewritten in the terms used in this thesis as Equation 3, where H is divided by the sum of all frequencies ($\sum p_i$). Without overlap (as in classical entropy-based scenarios) $\sum p_i = 1$, and thus $H = H_c$. Increasing overlap reduces the value of H (Fig. 160).

$$H_c = - \frac{\sum_{i=1}^s p_i \cdot \log_2(p_i)}{\sum_{i=1}^s p_i} = \frac{H}{\sum_{i=1}^s p_i} \quad (3)$$

Uniform values and improper subgroups

As discussed above, the situation that all character values in the remaining taxa are uniform (a single state, or always the same state combination) makes a character worthless for identification and should correspond to an information content measure of zero. Both H and H_c fulfill this condition and reach zero asymptotically as the frequencies approach this situation.

An additional special situation is that the frequency for some but not all states is 100%. Dallwitz (1974) and the DELTA standard call this an “improper subgroup”. In a branching key this situation is highly undesirable, because at least one decision will not further the identification at all. On the other hand, if a character with four states has, e. g., $p_i = 0.33, 0.33, 0.33, 1.0$, it may be more desirable than other characters with highly uneven state frequencies.

Profiting from applicability rules (character dependency)

Most coding status values are equivalent to completely missing information. The status value “not applicable” may, however, be treated specially in those cases, where it is also covered by a character applicability rule (“character dependency”). Doing so substantially improves the performance of a character guidance algorithm.

Character applicability rules express a relation between a controlling and a controlled character. In a given description, the controlled character is either: Inapplicable unless a specific state in the controlling character has been scored (“*Applicable-if*”) or applicable by default, unless a specific state in the controlling character has been scored (“*Inapplicable-if*”, compare “Character applicability rules”, p. 76).

During identification, character applicability rules are evaluated based on the data already entered for the object that is being identified. For the purpose of identification, a dependent character may be in three conditions:

- a) it is *inapplicable* if data for the controlling character exist and an applicability rule evaluates such,
- b) it is *applicable* if data for the controlling character exist and an applicability rule evaluates such,
- c) it has *unknown* applicability if the state of the controlling character is unknown (empty or coding status entered).

During data entry, unknown applicability would be interpreted as inapplicable for the *applicable-if* rule and applicable for the *inapplicable-if*-rule. For the purpose of identification it is, however, desirable to distinguish this condition.

Characters known as *inapplicable* may be excluded from the list of available characters, or may be ranked lowest (e. g., by applying the highest possible cost). The latter method has two advantages: Firstly, it allows the user a chance to still score them and by doing so highlight a scoring error either in the controlling or in the controlled character. Excluding the inapplicable characters incorrectly assumes that scoring controlling characters is always correct. Secondly, it

avoids special treatment of these characters in the ranking algorithm, which does not have to implement additional methods for character exclusion.

Characters known as *applicable* require no special handling and may be ranked normally.

Characters with *unknown* applicability are in principle informative because they may allow inferences about the controlling character, reversing the direction of evaluation. For an *applicable-if*-rule, this inference can always be made; for an *inapplicable-if*-rule, it can be made if only two states exist. How this information may be used, depends on the purpose of character guidance:

- When constructing branching keys, characters with unknown applicability should be treated like characters known to be inapplicable, i. e., excluded or ranked worst. When using a character with unknown applicability, one of the lead propositions would otherwise be required to explicitly include the option of inapplicability. For most users this would be difficult to decide and is therefore undesirable.
- When ranking characters for multi-access keys, the frequency of taxa for which a character is coded or inferred as inapplicable may be treated equivalent to a state frequency. In theory the frequency p_m may consist of explicit inapplicable coding status values, implicit inapplicability (no coding status, but known through applicability rule), and truly missing data (if the controlling character is unknown). However, differentiating these situations requires very time-consuming evaluation of many descriptions. The author recommends two possible shortcuts:
 - Simply treat 50% of p_m as truly inapplicable. That is, the corrected proportion of missing data is calculated as $p'_m = 0.5 p_m$.
 - If more precision is desired, the p_m of the controlling character may be used. To simplify this process, the sequence of ranking characters may be organized such that controlling characters are calculated first, and the p_m values of these are cached. Then

$$p'_{m, \text{controlled}} = p_{m, \text{controlling}} \cdot p_{m, \text{controlled}}$$

In both cases, the informative proportion of inapplicable characters may be calculated as $p_{s+1} = p_m - p'_m$. Character-ranking equations may then be calculated by summing from s to $s+1$.

Ranking quantitative characters according to separating power

Quantitative characters are often very efficient in minimizing the set of remaining descriptions, especially in multi-access keys that support a direct input of values. However the reasoning developed above is applicable only in two situations:

- For quantitative integer characters recorded as individual values rather than statistical measures each value may be treated like a categorical state.
- If a mapping to categorical character states (“Mapping univariate continuous measurements to categories”, p. 66) exists, a quantitative character may be ranked on the basis of this.

Developing a ranking for quantitative characters with data expressed as statistical measures (central values, ranges, variance measures, etc.) is difficult.

The strategy employed by Intkey is reported by M. Dallwitz (email to DELTA-L discussion list, 18 May 2005): “*In Intkey (unlike the key-generating program Key), key states are not explicitly defined by the author. Numeric values are used directly in the identification process. Integer-numeric characters are actually treated as multistate characters, with each integer value corresponding to a state, up to an arbitrary maximum, which is currently 100. Values above that maximum are lumped together in a single state. This method allows non-contiguous sets of integers to be treated correctly. E. g. if a taxon is recorded ‘5 or 10’, then it would not match a value ‘6’ entered in an identification. If the lumping of large values causes unacceptable information loss, the author has the option of treating individual integer-numeric characters as real-numeric in Intkey. – For calculating the separating power of real-numeric characters, ‘key states’ are automatically generated from the endpoints of the recorded ranges. For example, if there were 4 taxa recorded 2-3.3, 2.4, 3.1-4.2, and 3.1-4.5, the key states would be: up to 2 / 2-2.4 / 2.4-3.1 / 3.1-3.3 / 3.3-4.2 / 4.2-4.5 / 4.5 or greater. – For efficiency, the definitions of the key states are*

generated only once, and stored for later use. The assignment of taxa to the key states is done when a 'best' calculation is required."

Pankhurst (1991) advocates a search for gaps between value ranges (minimum/maximum or any available statistical range measure), and ranks these gaps based on the ratio of gap width to combined width of adjacent ranges (Equation 4). This method is promising if only few taxa remain. The chance to find gaps decreases with increasing numbers of taxa. Furthermore, the method does not support a co-ranking of categorical and quantitative characters.

$$\frac{\text{gap}}{\text{range}}, \text{ where: } \begin{array}{l} \text{gap} = \text{upper}_{\min} - \text{lower}_{\max} \\ \text{range} = \text{upper}_{\max} - \text{lower}_{\min} \end{array} \quad (4)$$

The problem of automatically finding appropriate divisions in quantitative values is also addressed by Thiele (1993), who describes a new method to categorize quantitative data. This method has been implemented, e. g., in the "Morphocode" program (Smets & Laboratory of Plant Systematics 2003) running under Mac OS X 10.2 or higher.

Combining algorithmic with authored character guidance

A major problem with algorithms based on the information content of existing data is that some characters may be very informative, but difficult or expensive to observe. The assumption made by the algorithms discussed so far that data are readily available is often not valid. It is therefore desirable to modify the character-ranking algorithms discussed above to include authored information on observation convenience, availability, required expertise, etc. (compare "Authored character guidance", p. 267).

Balancing the relative weight of authored and algorithmic rankings is not trivial. Dallwitz (1974) proposes an algorithm ("BEST") in which balancing and other factors are parameterized (Rbase, VARYWT, Reuse, etc.). This algorithm is used in the CSIRO DELTA programs such as Intkey and Key.

A special form of guidance by the "BEST" algorithm is support for preferential treatment of certain taxa. This has not yet been discussed. The BEST parameter is "Abase", and the taxon-specific DELTA directive is called "Item abundance". It is slightly misleading to interpret the name literally: The actual abundance of an item in a geographic region is only one possible factor influencing the decisions about prioritizing taxa in a key. Whereas beginners may profit from frequent species coming first, practitioner often know the most frequent species from memory and may perhaps desire them to come last, preferring medium abundant ones. Thus, "Item abundance" represents the intent that, for a given audience, a taxon should be keyed out earlier than others.

229. It is desirable to store parameters of character guidance algorithms (like DELTA VaryWt, Rbase, Reuse) as key-metadata. These data are not descriptive data, but specific to a given algorithm and key (authored branching key or multi-access key data set).
230. It remains inconclusive whether the information model must support metadata on taxa resulting in a preferential treatment in a key. Such data could express the intent that some taxa are keyed out faster than others. If supported, the model should allow different values for different audiences and key algorithms.

Alternative algorithms

Alternative algorithms have been used such as "separation coefficient" or "dichotomizing value" (Radford & al. 1974). They are not presented in detail here, since the advantages of methods based on information content seem to be generally accepted. Dallwitz (1974) and Pankhurst

(1991) are useful references to study earlier, alternative algorithms. A very recent study of alternative guidance algorithms, including a controlled performance study, may be found in Gibaja Galindo (2004).

Presentation of character guidance in multi-access keys

A distinction should be made between the mechanisms of guidance in character selection and how the guidance is communicated to the user. Some dependency exists insofar as some methods provide a global order of characters whereas others (such as “best path”) only provide a small set of “best characters”, without ranking the remaining ones.

Character guidance is usually prompted by some user request, i. e., the user explicitly asks for suggestions. It may be displayed in a separate window (as in X:ID) or it may modify the standard identification interface (most programs). Once character guidance is invoked, it may be automatically updated with each identification step if it adapts to identification progress.

Following the example of Intkey, character guidance often occurs by rearranging the character list in the sequence in which they are recommended to be used. This display has the advantage that a user may easily override the sequence by choosing a character further down the list. It is also possible to limit the number of recommended characters that are being displayed. For example, although Intkey by default shows all characters, it allows the user to limit the number displayed. This is unproblematic as long as the removal of “lowly recommended” characters is reversible (e. g., by returning to a character display arranged by some concept hierarchy (p. 125)). Otherwise, limiting the available characters impedes the ability to employ previous knowledge about rare character states to accelerate and simplify identification.

A different form of presentation may be selected for those characters that no longer have any separation power within the group of remaining taxa. Such characters may be removed from visibility (called variously “eliminated”, “hidden”, or “pruned”). The process may be either automatic as identification progresses, or upon an explicit request.

CBIT Lucid (p. 21) and Visual Key (p. 19), in addition to removing redundant characters, also perform a similar analysis of states and remove redundant character states. Although this may accelerate the identification progress, it may aggravate previous identification errors and lead to further mis-scoring (Dallwitz & al. 2006). Beginners are very likely to choose among the available states, even if none of the remaining states truly fits the object, and even experienced users may be misled about the semantics of a character state if the neighboring states are missing. The Pankey (p. 19) multi-access key “Onlin7” performs a similar analysis, but only “flags” states that become redundant with identification progress. In many identification use cases such “flagging” (perhaps using a different or dimmed color) of redundant states will be preferable to complete removal.

6. Use case analysis

6.1. Introduction

To achieve further insight into the requirements for a descriptive data system, “use cases” are collected and discussed in the following.

A use case may be viewed as a task that an actor (i. e., human, institution, or software agent) can perform. Use case modeling can be used for different purposes and with different level of detail. Here it is employed to obtain a broad to moderately detailed inventory of system functionality. The following use cases are not intended to formally describe system behavior or to col-

lect implementation milestones. Future work may try to develop the present use cases to the level of detail required for this.

The use cases are presented using UML. Similar to other UML diagrams, generalization relations between use cases are indicated by a large triangle pointing to the more general use case. The derivation of specialized use cases from general ones is used in the following to provide the desired level of detail without losing the general overview.

If several use cases include a number of common steps in their scenarios, these steps are modeled as a separate use case that is included in the other use cases (relationship labeled with «include»). Similarly, if a number of steps form an optional extension in a use case, the optional part is presented as a separate “extension use case” (relationship labeled with «extend»). Some use cases are abstract generalizations that are not expected to be used directly (i. e., only the specialized cases will be implemented). Such use cases are called “abstract”; their name is written in italics. See the section “UML use cases” in Methods (p. 23) for further information on the modeling techniques and notations used.

In the following sections first potential roles and agents for use cases are discussed, and then the use cases. The large number of use cases has been grouped by knowledge management categories: information acquisition, review and interpretation, retrieval, and application (Watson 2002). Which arrangement of use cases is considered “natural” depends to some extent on whether the user is a data provider or consumer. The arrangement chosen here is perhaps most natural to a scientist who collects descriptive data for later use in identification and analysis. A consumer may perhaps prefer a different sequence, starting with information retrieval (including identification) and followed by information application and analysis.

The primary goal of this section is to reconsider the analyses and requirements presented so far from a different perspective and to create a high-level framework into which these and future requirement analyses may be integrated. The present form was achieved after much rearrangement and it may be reconsidered as new use cases are being added. Already at the present time it is known that additional work

Because the use case analysis largely reconsiders what has been discussed in the previous chapters, the collection of numbered requirement statements is mostly stopped here.

6.2. Roles and agents (use case actors)

A potential list of roles and agents interested in descriptive data could look like:

- Data entry/revision roles:
 - Responsible experts:
 - Professional scientist
 - Amateur scientist
 - Student (including pupils in school)
 - Typist role (working for and under supervision of data entry roles)
- Data retrieval roles:
 - Professional scientist
 - Amateur scientist
 - Student (specializes into primary school to university)
 - Policy maker
 - Government agent
 - Agricultural agent
 - General public

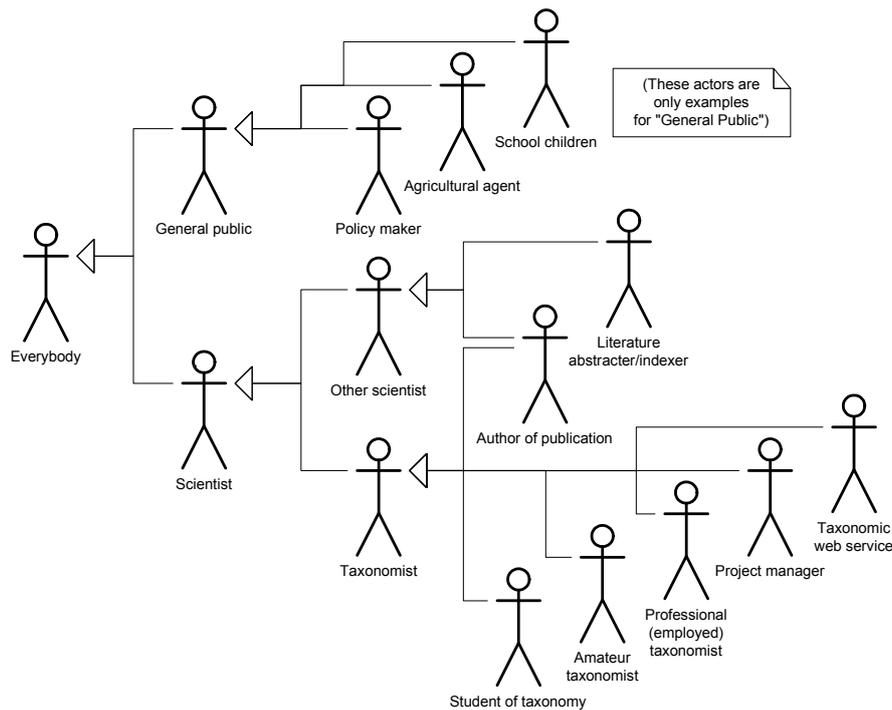


Figure 161. Potential UML use case actors and their hierarchy. Note that most of the following diagrams use only Taxonomist and General Public (see text).

These and other actors and their hierarchy are depicted in Fig. 161. However, in most of the following UML use case diagrams only two agents have been used:

- **Taxonomist**, and

- **General Public**, for the following reasons:

1. In a specimen collection management system the professional (i. e. employed) scientist (curator of the collection) has a special position, because the validity of part of the information depends directly on the institution. No such distinction between the professional and the amateur or student scientist exists for descriptive data. The relevant distinction is that of expertise, which does not depend on employment status. Expertise is difficult to assess and in the case of biological descriptive data is extremely specific to taxonomic groups. Academic grades or years of experience with a group can be useful indicators, but no clear-cut distinctions proved to be viable in modeling. Therefore all “expertise roles” have been combined in a single agent type **Taxonomist** in the UML use case diagrams.
2. For similar reasons the data retrieval roles have been combined to **General Public**. Expertise is difficult to measure and highly dependent on the area of specialization. Furthermore, agents may be interested in aspects like biotechnology, medicine, etc. Cases where expertise is relevant exist, especially in relation to different linguistic registers that may be defined in terminology for different audiences. However, this is a general feature applicable to any use case that involves communication with or reporting to a use case agent. It was therefore not considered helpful to introduce expertise in the use case diagrams.
3. The distinction between a typist and a responsible expert is relevant, but was omitted to simplify the diagrams. No separate use-case for typist-only action has been identified. Many data entry or data revision use cases may consist of a typist phase and a quality control phase (although in practice often the scientists perform both roles themselves). This would be well worth modeling in a more detailed use case model, but the current model does not distinguish these phases.

In addition to Taxonomist and General Public, the following agents have been occasionally used to highlight certain special roles or user communities:

- **Author of publication**
- **Literature abstracter/indexer**
- **Other Scientist**
- **Project manager**
- **Web service**

6.3. Information acquisition

Acquiring the descriptive data content itself and information about the terminology used to express these descriptions is perhaps the most critical step in managing descriptive data. The disparity between the information already acquired and the amount that would be desirable to acquire is huge (see Fig. 1, p. 14). Most biologists will agree that it is much larger than the difference between, e. g., current information retrieval and application procedures and those that could be imagined as desirable.

The following section will discuss information acquisition in a chronological order that would be required when starting a new project: Defining project metadata, defining the terminology, and accumulating descriptions.

Project management

The term “project” is used here for a container that ties descriptive data and terminology used together and provides metadata concerning intellectual property rights (IPR), licenses (e. g., Creative Commons or Science Commons), scope (taxonomic group, geographical range, perhaps seasonal applicability), version numbers, dates of initiation and updating, etc. of a descriptive data set.

Some project management use cases (Fig. 162) like project creation, deletion, backup, and restore are trivial. If an application works document-oriented (one document per project), or if projects are identical with named databases in a DBMS, deletion, backup, and restore may already be covered by standard file manager or database manager tasks. However, in other cases the granularity of stored information will encompass several projects (e. g., a single set of database tables holding multiple projects). File or database backup and restore will still protect against hardware failure or global operator errors, but project delete, backup, and restore use cases need also be implemented on a project-level granularity.

The export and import use cases are closely related to this. The exchange format may be a fairly complete representation of descriptive data like the SDD format, or more selective formats like DELTA or NEXUS. If the exchange format is capable of preserving all information (the SDD format offers this ability with the provision of application-specific extensions), the backup/restore use cases may be derived from “export/import to exchange format”.

The most complex use case in project management is the provision of project metadata. The two aspects of project metadata:

- informing about the project itself, and
- summarizing information about the terminological and descriptive content

are often difficult to separate and are therefore not modeled as separate use cases. For example, on the one hand editors of a project must not necessarily be identical to editors of individual descriptions, on the other hand project contributors will often be simply the sum of all content creator roles (e. g., author, editor, contributor, or translator). Similarly, a last-update date of the project would be the most recent last-update of any content item. It may be desirable to be able to mark some information as being or to be automatically generated. Several issues of project “envelope” information have been discussed at the SDD meeting in Paris (see Hagedorn 2003b).

The importance of project and data set metadata is often overlooked (e. g., the original DELTA relied entirely on external documentation for this). Farr (2006) argues for the importance of communicating coverage and scope of identification keys.

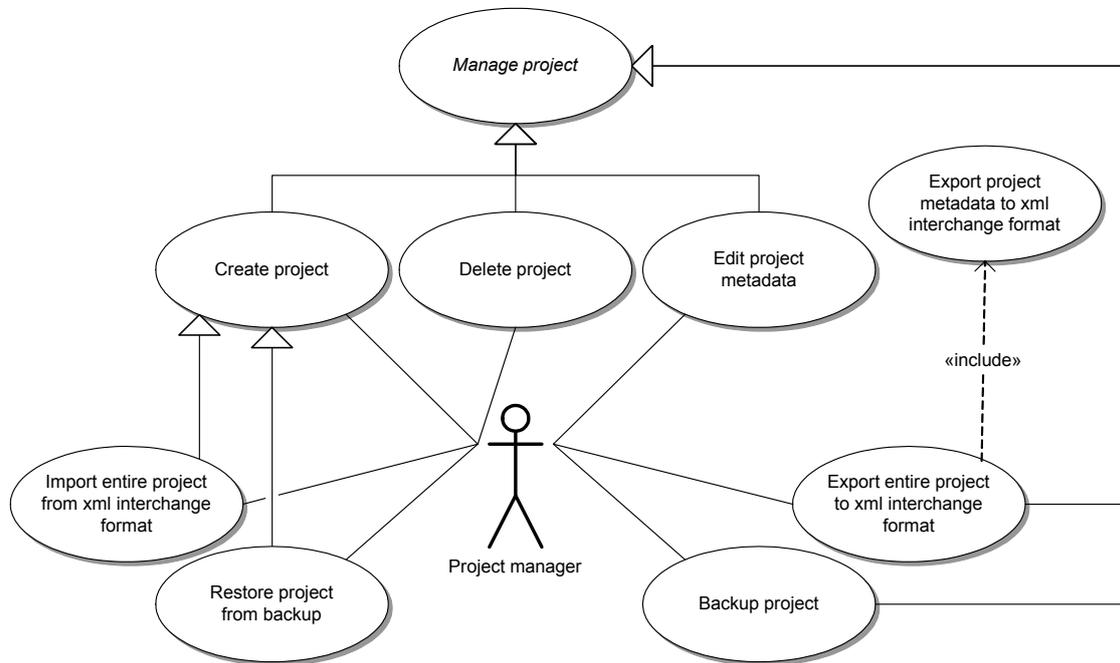


Figure 162. UML use case diagram for project management.

231. Support for metadata on the level of data sets (or “projects”) is desirable. This may include intellectual property rights (IPR; including authorship, ownership, copyright, and licenses), coverage and scope (taxonomic group, geographical range, perhaps seasonal applicability), version numbers, initiation and modification dates, etc.
232. Some data-set-level metadata may be calculable from individual objects (contributors, last modification, coverage) others not (editors, scope, etc.). Flagging data as having been, or to be automatically updated may be desirable.

Definition of terminology

Defining the terminology provides a major part of the overall information model in which the descriptive data are expressed. It is a peculiar feature of descriptive information models that a major part of the information model (or “schema design”) is thus left to the content developer (see also section “Structured descriptions and the concept of terminology” on p. 42). The base model itself defines only general structures and data analysis types from which specialized data types are derived by the domain expert (taxonomist). The reason for this is that much of terminology is specific to taxonomic groups. At least the compositional terminology must be defined separately. Unfortunately, even definitions of potentially generally applicable properties and methods terms are often specific to taxonomic groups, or to geographical or scientific “schools”.

The definition of terminology has two aspects:

1. *Ontological terminology*: Scientific knowledge and generally usable definitions of terms may be expressed as ontological statements. These statements may be arbitrarily complex. For example, an unlimited number of morphological or anatomical part hierarchies may be defined,

and the complexity of terminology does not have to be limited by practical considerations of applicability of audiences.

2. *Operational terminology*: A restricted and constrained version of the terminology may be defined for data entry and identification purposes.

Ideally, these two aspects should be handled by a single model, but it has proven difficult to find a model that handles both aspects equally well. The use case discussion therefore treats the two aspects as separate specializations of “Define terminology” (Fig. 163).

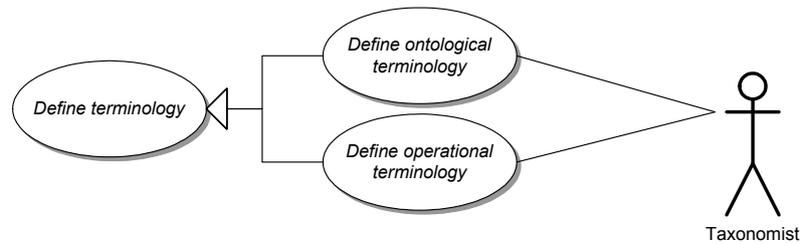


Figure 163. UML use case diagram showing that terminology may be defined separately as fundamentally true ontological statements, and as operational terminology to be used under existing limitations. All three use cases are *abstract* (shown here in the following with italic use case names).

Multiple languages or audiences

In principle, a terminology can also attempt to define the terms used in “prose-like” natural language descriptions. The advantages and disadvantages of such descriptions have already been discussed above (p. 39). One of the problems of natural language descriptions is communicating in multiple languages or with multiple audiences (school children, scientists, etc.). This will be discussed first, before addressing ontological and operational definitions of terminology.

One strategy to express descriptions in several languages is the development of automated translation systems based on natural language. Despite some problems (e. g., terms are frequently ambiguous and context-sensitive, and information may be implied in one language and has to be expressed in another language), the relative simple nature of scientific descriptions would appear a promising target for automated translations. Major problems are, however, that natural language descriptions often contain errors or inconsistent usage and that – embedded in the simple, structured descriptions – notes may use the full expressiveness of a language. Translation systems constrained to structured terminology will then only allow partial translations. Semi-automatic translation systems are of little interest to the taxonomic community. Firstly, the resources to check thousands of descriptions are lacking and secondly, most biological descriptions generated in digital format are seen not as a final product, but as a first version that the author team hopes to improve gradually through revision and criticism from outside. No implementation of an automated translation system for descriptions is known to the author.

Instead, a long tradition exists in the DELTA and NEXUS communities to express descriptive data through symbolic codes. Each term defined in the terminology has such a unique code associated with it and this code is used in the object or class descriptions. If appropriate labels and wordings for each supported language or audience (such as school children, university students, etc.) are provided, natural language descriptions can be generated from the coded descriptions. Similar to automated translations, the specialized notes that are not covered by the general terminology cause a problem and require manual translation. However, they are clearly identified in a structured data format rather than being embedded in unstructured text. If only relatively few such notes exist, the dynamic nature of descriptions is lost only for these parts.

Although the original DELTA format supports automatic translations for characters and states, much had to be expressed as free-form text in notes. Proposals to introduce additional elements to

increase the expressiveness of coded terminology (“coded comments” in “New DELTA”, p. 20; modifiers in DiversityDescriptions and SDD, see “Modifiers”, p. 189) substantially reduce the amount of free-form notes requiring manual translations.

An important question is, whether the codes should be semantically meaningful (e. g., “blue” representing the color blue) or not (e. g., “27.66xx”). Semantic codes have advantages during the debugging of applications, but should otherwise be avoided. When the terminology evolves, e. g., when many different shades of blue are added, it is often not possible to keep the intuitive semantics of the code and the true semantics of the definition synchronous. The confusion caused by such an “impedance mismatch” between assumed semantics and true semantics of a code may cause considerable errors and delays during application development, outweighing the original advantage during debugging.

The extreme case of a system with semantically loaded codes would use natural language phrases in one selected language directly to store the descriptions. If the system controls the syntax and provides uncontrolled containers for free-form text notes, the system would appear like a natural language description, but without the translation problems noted before. In this case the advantages may be considered to outweigh the disadvantages noted above. Especially, such data could be directly reported at least in one language, and would still be useful if knowledge of the terminology has been lost (e. g., in a partially conserved archive). However, before deciding to design such a system, two other aspects must be considered:

- Since such a system can never work with single words, it must be guaranteed that phrases are always unique. Two phrases may start identically, and one code may be a part of another phrase, but the longer phrase may never be identical to a combination of phrases allowed by syntax of the system. This behavior is very difficult if generality (usability in different languages, e. g., Chinese) is sought.
- Because codes and representation are coupled, changes to the terminology (schema evolution, refactoring) require modifications to all descriptions expressed in the terminology. This is often not possible if data are federated.
- The system would have a non-symmetric behavior regarding report generation and translation. In a coded system, improving and refactoring the terminology can be achieved with a single set of methods. If one language is also used for coding, this preferred language would need other methods than the translations.

Ontological terminology

The definition of use cases for ontological statements (Fig. 164) is relatively preliminary, since no existing applications could be studied in detail and relatively little experience exists. Some sources of information are:

- The “Plant OntologyTM Consortium” (<http://www.plantontology.org/>) plans to cover two vocabulary domains: plant structure and plant growth and development for selected, genetically and agriculturally important plant species. It uses three relations: “is-a = type-of”, “part-of”, and “develops from” (e. g., “seed coat develops from integuments”), creating directed acyclic graphs (DAG, p. 234). Early vocabularies were narrowly focused on molecular model species, but recently more general vocabularies begin to emerge (e. g., Pujar & al. 2006 for stages, Ilic & al. 2006 for structure). However, these remain to be optimized for use in a molecular context, compare the synonymy problem discussed on p. 159.
- The Prometheus description model (p. 21) concentrated on guaranteeing that all used descriptive terminology must be properly defined. The published information is yet insufficient to elaborate the use cases further and it is difficult to assess in how far the intended definitions are accessible to automated reasoners.
- Many other projects address the problem with a dictionary or glossary approach without formal machine-readable ontologies, e. g., the online version of Radford & al. (1974):

<http://www.ibiblio.org/botnet/glossary/>, the LIAS glossary (Ryan & al. 2005), or the U-PLanT and OpenKey projects (Greenberg & al. 2005, 2006).

Ontological descriptive terminology may in the near future be primarily used to improve the internal consistency of the operational terminology, and as a backdrop to operational terminological descriptions (e. g., by generating dynamic dictionary or glossaries that can be referred to by the operational definitions). The intended future use of an ontological vocabulary is its consumption by semantic software agents (Berners-Lee & al. 2001). Much may change in this area, since the definition of web languages for semantic and ontological statements is under very active development. Only recently the first versions of RDF and OWL have been elevated to w3c recommendation status (RDF: W3C 2007, Klyne & Carroll 2004, Manola & Miller 2004, Brickley & Guha 2004; OWL: McGuinness & van Harmelen 2004).

The topic has already previously been discussed in “Fundamental aspects of description models: Descriptive ontologies”, p. 131.

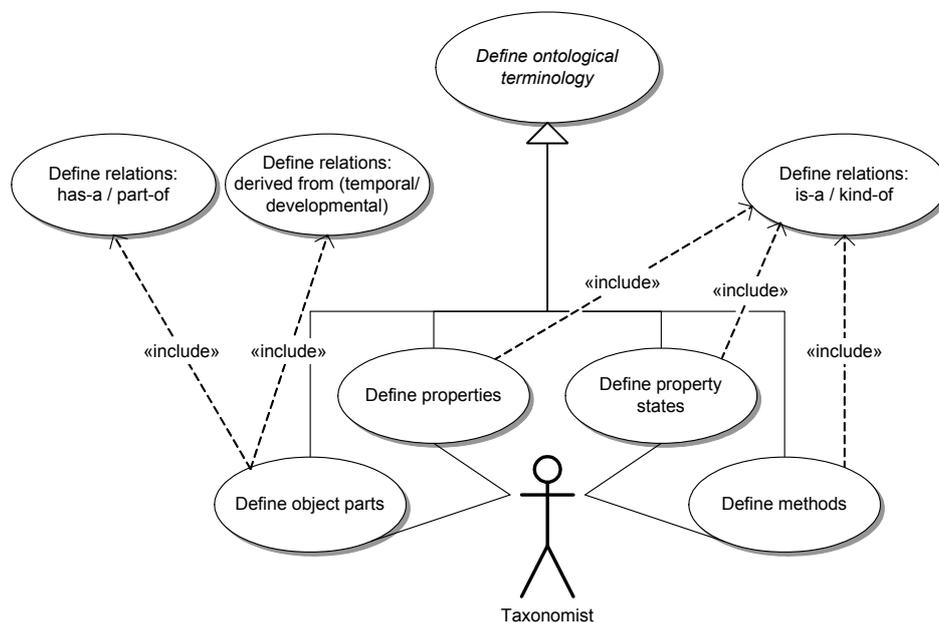


Figure 164. UML use case diagram for the definition of ontological terminology. A composition hierarchy for object parts (using has-a = part-of relations) and generalization hierarchies for properties, property states and methods (using is-a = kind-of relations) are defined separately.

Operational terminology

In contrast to ontological definitions, all existing applications and approaches to descriptive data have methods for the definition of operational terminology (“character definition”). The use cases presented in Fig. 165 are based on a character model (see “Description storage models”, p. 104) extended with concept hierarchies (p. 125) and modifiers (p. 189) as currently proposed for SDD (p. 20). Other models (e. g., “Character decomposition models”, p. 116) would result in a different collection of derived use cases for “Define operational terminology”.

The central operational entity is called “character”. A character can be understood as a, variable, property or “data container” containing free-form text, categorical data, or quantitative value. The enumeration of categorical values is defined as categorical states, the various kinds of univariate statistics for quantitative values as statistical measures. Characters may be arranged into one or several conceptual hierarchies (compositional, methodological, by property, etc.). Character dependency rules (general dependency or specific applicability rules; compare “Analysis of character applicability”, p. 304) can be considered a special form of such character hierar-

chies. Note that the «include» relation denotes that creating a hierarchy is only possible after characters have been created).

Modifiers are a separate terminological entity for frequency (e. g., “rarely”, “usually”; p. 206), certainty (e. g., “probably”, “perhaps”; p. 207), or other modifications of states or measures (e. g., “strongly”, “when young”). They can be defined independently from other elements. A separate use case defines which modifiers are applicable to which character.

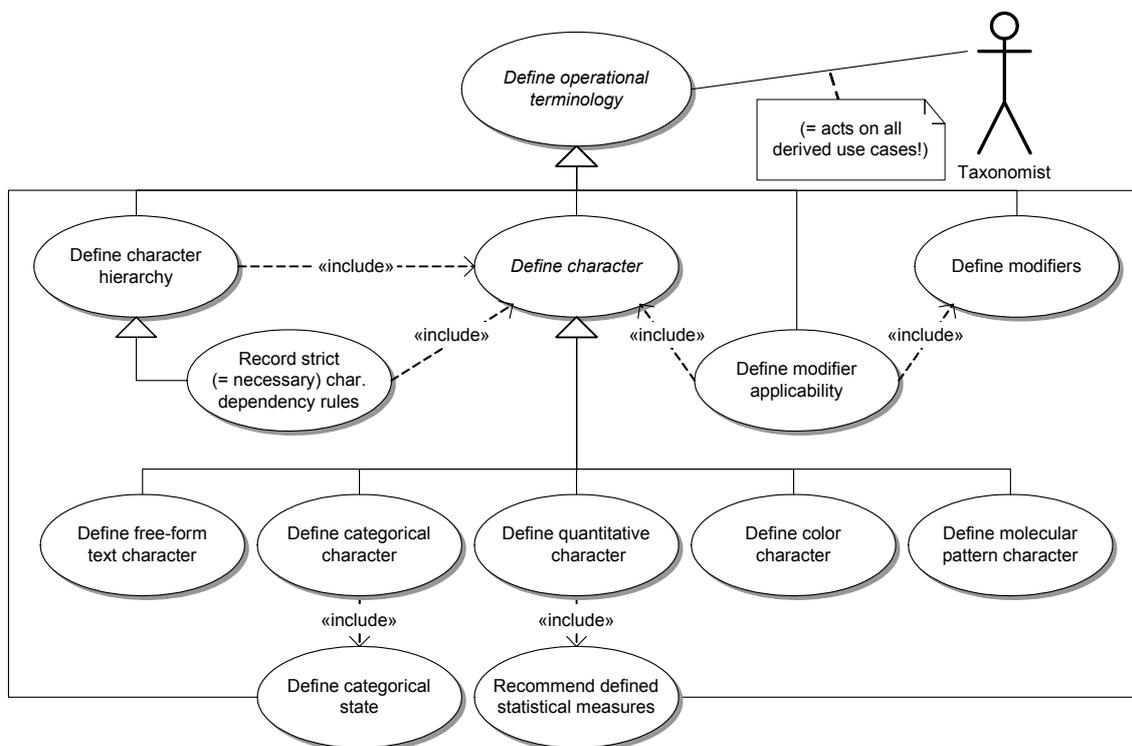


Figure 165. UML use case diagram for the definition of operational terminology. The definition of specialized color and molecular pattern characters are Examples of further specialized character types.

Short text labels are often insufficient to communicate terminological concepts to untrained information consumers. Attached images and other media can help, but often considerably more effort is needed to achieve exact definitions. Each specialized use case of the operational terminology (character with derived kinds of character, character hierarchy, modifiers, states, measures) has a number of common included use cases (Fig. 166, 168, and 169), covering both simple and elaborate definitions:

1. Each use case includes the definition of a label. Optionally, label abbreviations, and natural language wordings may be defined.
2. Beyond the definition already expressed in the label text or media associated therewith, elaborate definitions may be associated.
3. An association of media resources may be included in each use case.

In Fig. 166 the association of media resources is drawn as an independently included use case. Often it will be modeled as an indirect inclusion, through including the association use case in both the label and the elaborate definition use case (Fig. 167).

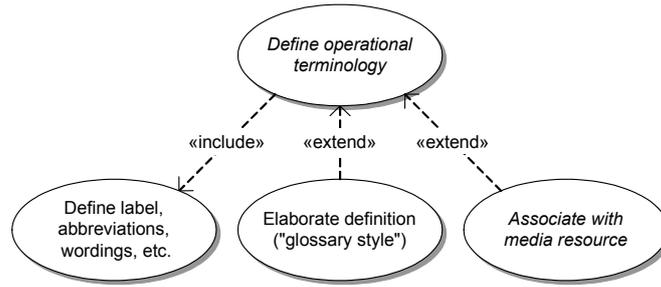


Figure 166. UML use case diagram showing the inclusion and extension of use cases to define “label, abbreviations, wordings”, “elaborate definitions”, and to “associate terminology with media resources”. The relations shown here affect all use cases derived from “Define operation terminology”, such as characters, states, modifiers, etc.

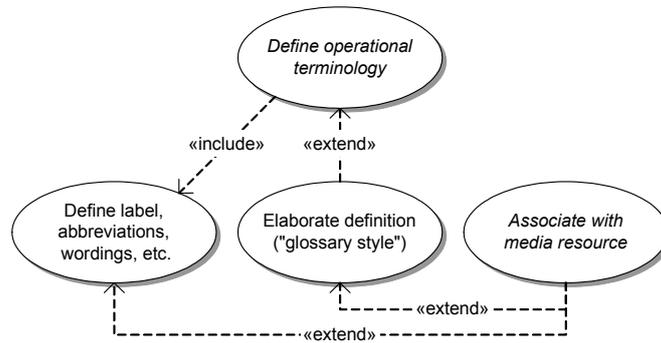


Figure 167. UML use case diagram showing a modification of Fig. 166. The association of media resources is now considered an extension of the label and the elaborate definition use cases. This solution is proposed in the SDD model.

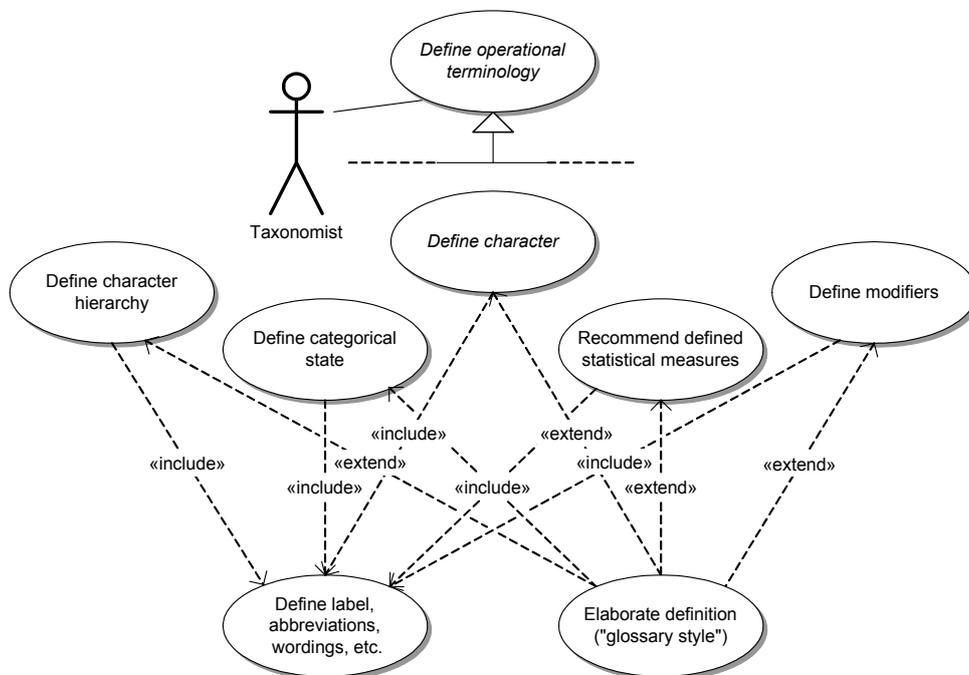


Figure 168. UML use case diagram for the inclusion of label and glossary use cases in some operational terminology use cases. This diagram explicitly shows the left two use case relations already implicitly present in Fig. 166. The generalization hierarchy is only hinted at the top; compare Fig. 165 for details.

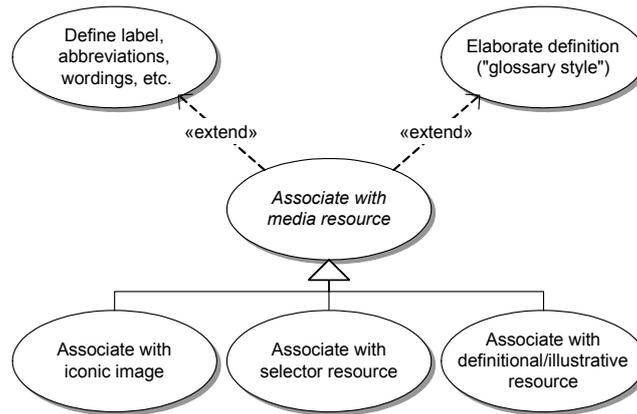


Figure 169. UML use case diagram showing the specific use cases of the abstract “Associate with media resource” (compare Fig. 167).

The explicit versions of the diagrams are shown to clarify the use of an include dependency relation to a generalized use case from which specialized use cases are derived. Note that most modifiers and all statistical measures may well be associated with iconic or definitional image resources, but the use of selector images is unlikely for these objects. However, no reason exists to prevent this from occurring (Fig. 169).

Coding status and statistical measures

The definition of coding status values expressing reasons why data are missing (compare p. 74) and statistical measures (compare Fig. 32 and section “Standard aggregation methods”, p. 85) may be considered special use cases. Methods analyzing, manipulating, and querying descriptive data must be aware of much of the semantics of the different coding status values and statistical concepts. This is different from categorical states, where the same methods must only be aware of general comparability and information embedded in the data type.

For a fully extensible solution the system designer might define a symbolic algebra and a sufficient number of semantic concepts to enable the taxonomist creating an operational descriptive terminology to base new statistical measures and coding status values on these agreed concepts (Fig. 170, left). Alternatively, one might limit the terminology to a fixed vocabulary with system-defined semantics may be created (Fig. 170, right) which cannot be extended by the taxonomist.

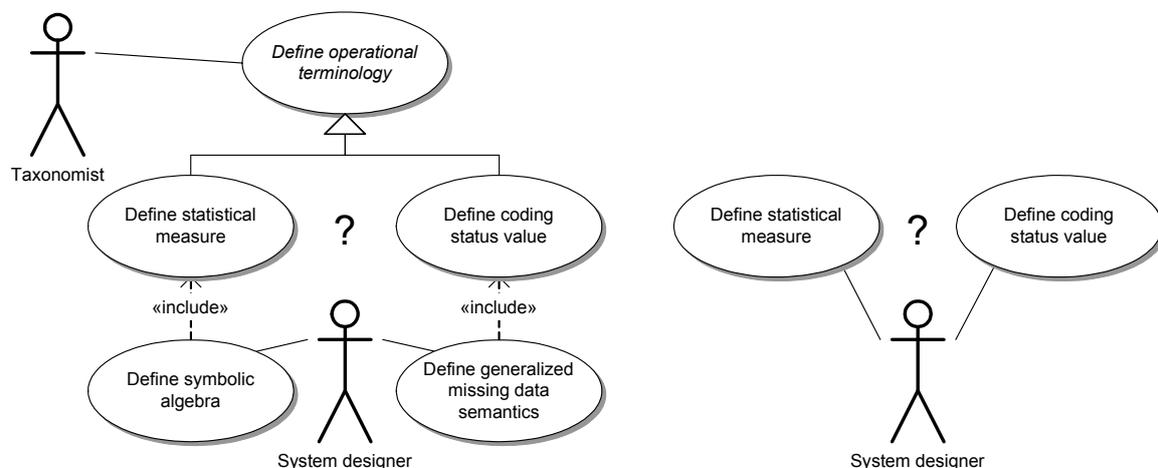


Figure 170. UML use case diagram showing options for defining coding status values and statistical measures. On the left an extensible vocabulary is defined by a taxonomist as part of the operational terminology, on the right a fixed vocabulary is defined by the system designer.

Originally, the extensible approach (Fig. 170, left) was attempted in SDD. Over time, however, it was realized that the complexities of pursuing this were so great, and the disadvantages of a rich but fixed vocabulary so relatively minor, that the system-defined vocabulary was preferred.

Federated and shared terminology

Use cases regarding distributed and shared terminology (Fig. 91, p. 182; Fig. 94, p. 185) have already been shown and discussed in “Federation and modularization of terminology” (p. 180).

Descriptions

Descriptive data can be obtained by:

1. Direct entry of descriptive data based on a newly devised exact terminology. This is the mode with the longest tradition in the application of computer methods to descriptive data. All DELTA- or NEXUS-based programs currently employ this mode (using local application form or online-forms over the internet, see, e. g., Hagedorn & Rambold 2000). It is discussed in the following section “Data entry of coded descriptions”.
2. The digitization of existing (printed legacy) data like flora, mycota, or fauna volumes, resulting in natural language digital text. This text may then optionally be marked up to provide additional structure, or it can be converted into structured data formats. Only relatively few natural language descriptions have so far been digitized and marked up and the author is not aware of projects studying the conversion of natural language with markup into coded description data. This is discussed in “Digitization and markup of descriptions”, p. 293.
3. Recording the process of identifying objects (Fig. 177). This is an entirely new process proposed in this publication; it is discussed in “Recording identification data”, p. 295.

A special aspect of descriptive data is that raw data and repeated observations may already be recorded, rather than synthetic, processed information. This will be discussed at the end under “Recording repeated original observations”, p. 296.

Data entry of coded descriptions

Data that are fully defined through a structured terminology that is available inside a descriptive data management system are called “coded descriptions”. This refers to the fact that the terms defined in the terminology are not taken at their natural language “face value”, but are codes referring to exactly defined entities.

Definition of blocks of coded description

Descriptive data may be viewed as atomic statements (e. g., “petals blue”) that can be organized by various principles into blocks of data. The most important organizing principle for such blocks is the definition of the real-world entity that is being described. In the model proposed here this may be either an individual (observed or collected) object represented through an object identifier (such as a specimen number or code) or a class of objects represented through a class name (taxon name). Probably any model for descriptive data will have some block structure and use case dichotomy that is based on these two elements (Fig. 171). Note that although object and class descriptions in theory are fundamentally different, in practice they are distinguished only by their definition (compare section “Are different models for individual and class descriptions needed?”, p. 361 for a discussion of this).

In addition, further organizing principles are citations (or “references”) of publications that previously published descriptive data, and the team of authors and editors that is responsible for the creation of a block of descriptive data. These elements may either be nested within the primary block structure or used as additional defining elements for the primary block structure. The SDD model (p. 20) chooses the latter option and allows an unlimited number of data blocks (called descriptions).

The citation of a publication is occasionally uncritically considered a “data source” for a description. This may indeed be appropriate in the future when referring to digital coded descriptions based on a standardized terminology and exchanged through data exchange formats like SDD. However, currently publication citations usually refer to printed or digital natural language descriptions. Viewing these as “data sources” is misleading, since the person creating the coded descriptions must translate from published natural language terminology to the coded terminology that has been defined in the project. This process involves a sequence of analyzing the usage of terminology in the published descriptions, comparing it to known reference descriptions (i. e., organisms that are described in the publication as well as in coded terminology – usually only implicitly because the translating person knows the organisms well), and then creating a strategy to translate the published descriptions to coded descriptions. Often errors or inconsistencies are detected in the published descriptions during this process, and appropriately coding of the resulting uncertainties is part of the coding strategy. The sequence of analysis and synthesis always creates original descriptive data, and similar to original scientific publications, the citations are supporting evidence and not “data sources”.

The problematic relation between publication citations and the changes necessary during data entry may be one reason why most descriptive data application are currently designed primarily to handle the third case (no references) and can handle the other cases only by adding unstructured information in free-form annotations. DiversityDescriptions attempts to improve this and links explicitly to collection or reference databases; however, the lack of interoperability with reference or collection applications makes this a difficult process that currently still requires custom programming.

To analyze the specialized procedures based on the definitional elements of descriptions (object ID, class name, publication citation, and author team), UML use case diagrams are not ideal.

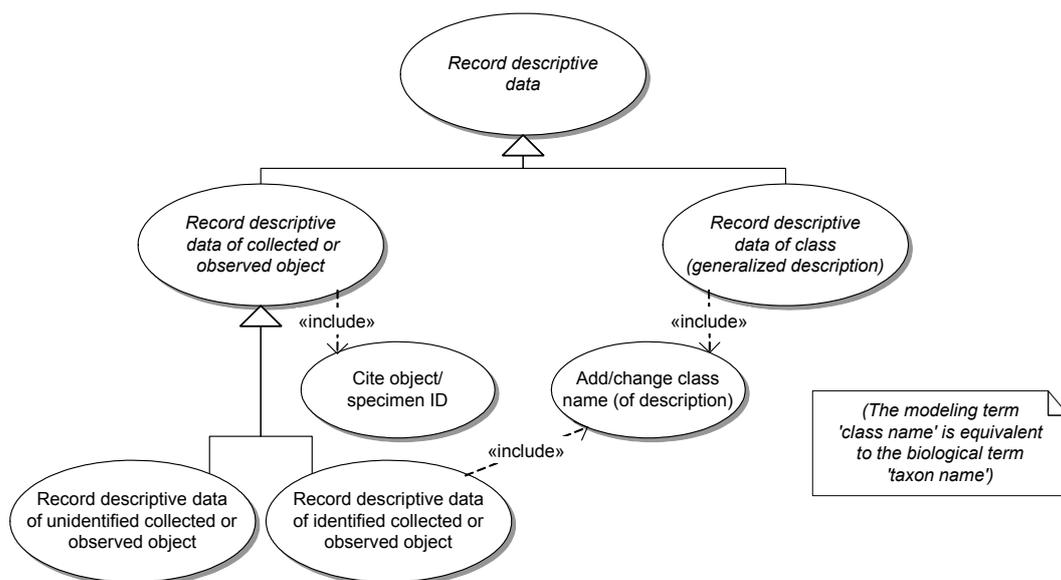


Figure 171. UML use cases derived from “Record descriptive data” for individual objects and class descriptions. The former may be unidentified or identified to a class name.

Each combination has to be modeled as generalizations with multiple inheritance, leading to a multitude of specialized use cases. In addition to the cases resulting from the definition of class name (i. e. taxon name) or object identifier (already shown in Fig. 171 above), at least those derived from recording and citing published data are also shown (Fig. 172). Deriving specialized use case for presence or absence of the author team of a description is considered less interesting. Any description has an author team, only it may or may not be recorded in the data.

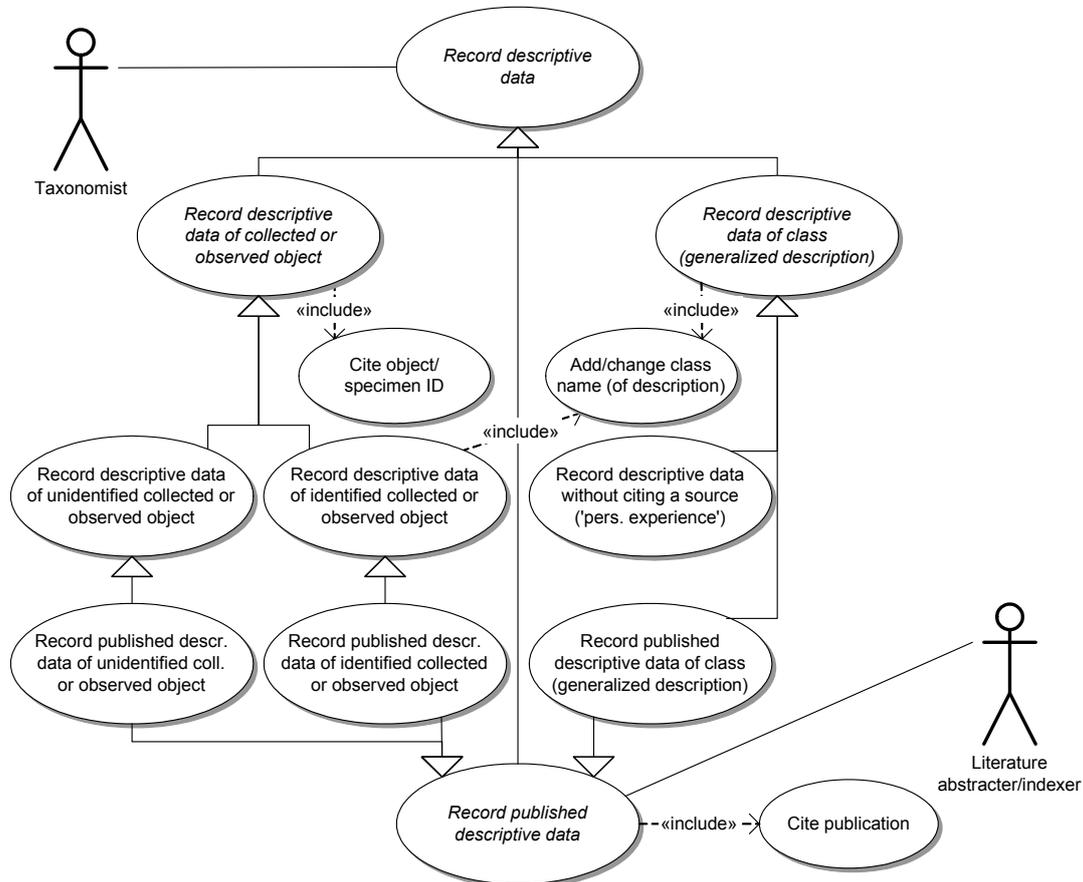


Figure 172. UML use case diagram extending the interactions already shown in Fig. 171 by those derived from the “Record published descriptive data” use case. Four basic abstract use cases, three included use cases, and six concrete use cases (corresponding to the check marks in Table 58) are shown.

Only six out of eight possible combinations of the three definitional elements are considered relevant (Table 58 and Fig. 172). An unpublished description without object ID or class name is considered data garbage, i. e., at least one of “class name” or “object identifier” must be present. The citation of a publication is always considered optional.

Table 58. Tabular presentation of the possible combinations between object ID, naming/identification, and publication status.

	– Object ID present –		– Object ID absent, –	
	Identified	Not Identified	Class named	Class unnamed
Published: yes	<input checked="" type="checkbox"/>	<input checked="" type="checkbox"/>	<input checked="" type="checkbox"/>	<input type="checkbox"/>
Published: no	<input checked="" type="checkbox"/>	<input checked="" type="checkbox"/>	<input checked="" type="checkbox"/>	<input type="checkbox"/>

(Object identifications and naming of classes may be expressed in the same data item “class name”.)

It may be surprising that descriptions of unidentified objects are being published. This situation occurs, e. g., when molecular sequences records are deposited in GenBank, EMBL, or DDBJ, that could not yet be identified. DNA may even have been obtained from environmental samples so that no object identifier of a voucher specimen or strain is present. Similar cases occur in phytopathology, where a description may refer to an unidentified organism causing a disease, or in publications requesting identification help.

The examples above include cases in which descriptions report neither a scientific taxon name nor an object ID, which seems to contradict the requirement that at least one of “class name” or “object identifier” must be present. However, these cases are relatively rare and a non-scientific name will always be present. It is common practice in biology to give these classes temporary names (often called “laboratory names”) to be able to refer to them. Thus, class names may include valid taxonomic names, but are not limited to them. Descriptions for non-taxonomic (non-phylogenetic) class concepts like “algae”, diseases names (one organism may cause different diseases on different hosts or under different conditions), or preliminary laboratory names may also be created. Supporting the explicit absence of class name and object identifier is considered an unnecessary complication of system design, since it would require the implementation of additional query mechanisms (based on, e. g., system-internal identifiers of descriptions) to allow users to retrieve and edit blocks of descriptions.

Character data entry

So far only the definition of the unit of description has been discussed, but not the entry of character data. The term “Character data” is used here to summarize all available combinations of the operational terminology (i. e., including categorical or quantitative data with or without modifiers, unconstrained text information and notes, or coding status values). Consequently, character data entry is a very complex use case. It is not discussed in detail here, since little insight can be gained from use case diagrams beyond what has already been discussed in the section on “Operational terminology” (p. 284).

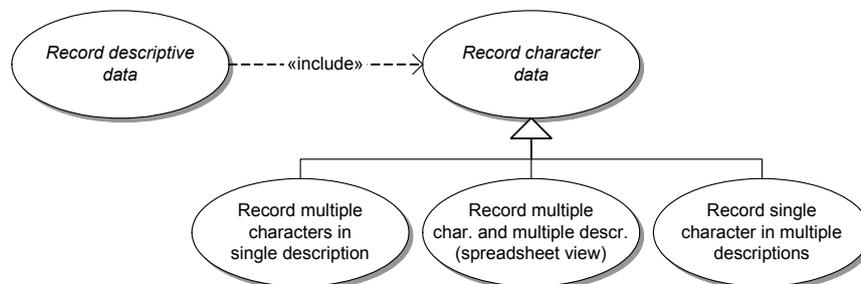


Figure 173. UML use case diagram showing various modes of recording character data.

Recording character data can be modeled (Fig. 173) as an included use case of the basic abstract use case “Record descriptive data” (i. e., recording characters is part of all six specializations shown above in Fig. 172). The specializations focus on the fact that different sequences of actions are possible. Each results in a specific editor design and each one has advantages and disadvantages. The three use cases presented in the diagram are discussed in use case scenarios with annotations:

Use case scenario: “Record multiple characters in a single description”

1. Instead of working on all characters one may select a character subset
2. Select a description or start a new description
3. Select a character (first, next in sequence, or free choice from set of characters)
4. Enter data for the selected character

5. Loop to 2 until all available descriptive data are entered for one description

Notes: This is the most common method. It has the advantage of focusing the actions on a single description. It is the only meaningful method when an object (i. e. specimen) is directly studied. It usually works well for class (i. e. taxon) descriptions, since most information is organized by taxon (like natural language descriptions).

Use case scenario: “Record multiple characters and multiple descriptions (spreadsheet view)”

1. Select a description
2. Select a character
3. Enter all character data
4. Move to the next character or to the next description, repeat 3

Notes: The free navigability allows more user choices, as well as errors. At least the description dimension of the spreadsheet matrix should be freely selectable or should be sortable by different criteria (taxonomic name or hierarchy) so that related descriptions can be seen next to each other. A spreadsheet view is implemented, e. g., in the new CSIRO DELTA editor for Windows. A major disadvantage of this method is that it is difficult to adequately present the internal character structure (multiple modifiers, states, or statistical measures) in a cell in a character × descriptions matrix. The solution chosen by the CSIRO DELTA editor is to display a DELTA-coded text summing all states (e. g., “1/4<comment>/5”) inside the matrix cells, requiring detailed knowledge of the DELTA format.

Use case scenario: “Record a single character in multiple descriptions” (compare Fig. 174)

1. Select one or several descriptions based on class name or object (specimen) ID
 2. Select a character
- (For categorical characters:)
- 3.a Select one or several states
 - 4.a Select whether to add or delete these states from all selected descriptions
- (For quantitative characters:)
- 3.b Select a statistical measure and enter a value
 - 4.b Add this to all selected descriptions
- (For free-form text characters:)
- 3.c Enter text for a free-form text element (data or annotation)
 - 4.c Add this to all selected descriptions

This method is especially useful when a data set is restructured or refactored (e. g., after amending the terminology). Furthermore, some information sources are organized by character rather than by taxon. Frequent examples are geographical distribution, host lists (for each country or host a list of taxa is given), or chemical substance data (for each substance a list of taxa or even specimens studied is given). A typical problem of this use case is that it does not easily support the creation of new descriptions (i. e., at least the description definition must already exist). Implementations may offer a mode in which after selecting a class or object name (taxon or specimen), descriptions are automatically created.

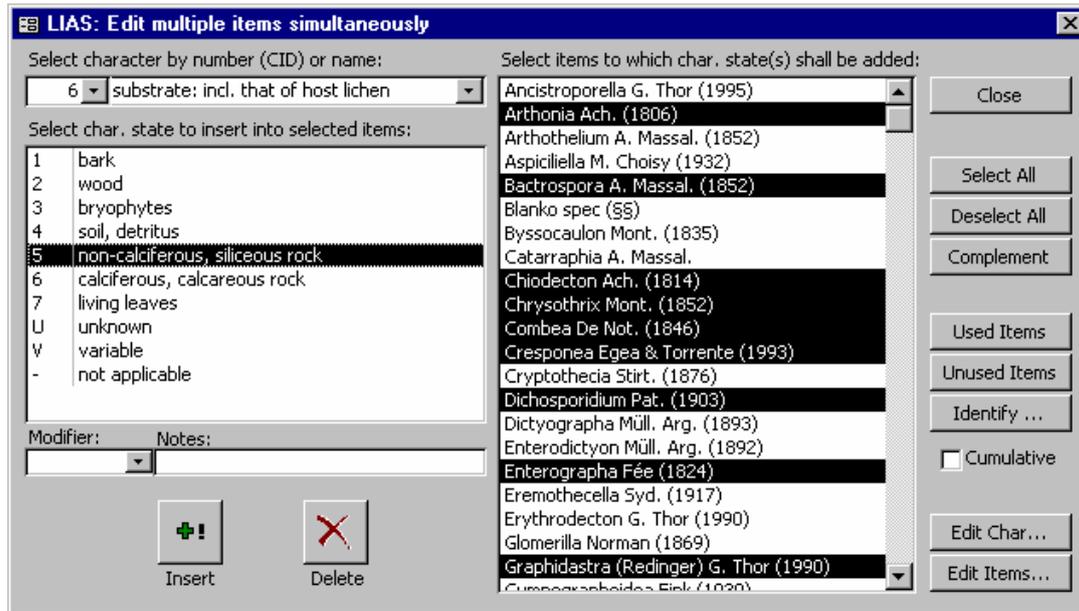


Figure 174. Example for a multi-description editor (from DiversityDescriptions).

Digitization and markup of descriptions

An operational terminology that is defined to enter data directly as coded descriptions (depending on the codes defined in the terminology) may also be used to markup natural language descriptions (Fig. 175). Since the initial digitization (typing or OCR) of natural language descriptions or keys is assumed to occur outside of the descriptive data system, the existence of these data is modeled as a precondition constraint in the use case diagram. Digitized texts may then be marked up using various methods. In SDD it is planned that all elements of terminology are available for markup use, including the structure, property, and method concepts not normally used in coded descriptions. The lead statements (or question/answer-combinations) of branching keys can in principle also be marked up like fragments of natural language descriptions. In the SDD concept an alternative method of separately recording coded statements that correspond to the branching key statements is proposed. A problem with this is that implied brackets and Boolean operators (not, and, or) are frequently found in keys, but rarely in natural language descriptions. Support for Boolean logic considerably complicates the information model and all analysis based on it. The question whether Boolean logic should be supported in any form of natural language markup, only in the special version of coded statements associated with key statements, or not at all therefore needs careful consideration.

Markup will often remain incomplete. In some projects only the major structures will be identified, not characters, states, modifiers, or values. Manual markup through human editing is one option, but the goal is often to achieve at least partial markup through automatic text classification procedures. Examples are:

- The “Terminator” software (Diederich & Milton 1993b, Diederich & al. 1999, Fortuner 2002) for the Nemisys project.
- Taylor (1995): Flora of New South Wales and Flora of Australia, Vol. 19; achieving extraction of 60-80% of character/state pairs.
- Markup of Flora of East Africa descriptions (TDWG 1999b, Kirkup & al. 2005).
- Catapano & al. (2006): TaxonX schema applied to digitized AMNH documents.
- GoldenGATE (<http://idaho.ipd.uka.de/GoldenGATE/>): Supporting both automatic NLP-based markup and manual intervention to improve markup (Sautter & al. 2007).
- Vanel (2004) developed a natural language parser called FloraParse and tested it on data from the Flora of China project.

- Cui & al. 2002, 2006, Cui & Heidorn 2007: automated conversion of natural language descriptions into a structured format using “MARTT (markuper of taxonomic treatments)” system, tested using the online versions of Flora of China and Flora of North America (both available on the eFlora platform, Brach & Hong Song 2006).

Related to the markup of descriptions is the markup and restructuring of branching keys found in digital or digitized documents. Approaches to this topic include Brach & Hong Song (2006) and the CBIT Lucid Phoenix program (CBIT 2007b).

If markup is sufficiently complete, it may be valuable to convert it to coded descriptions (Fig. 176).

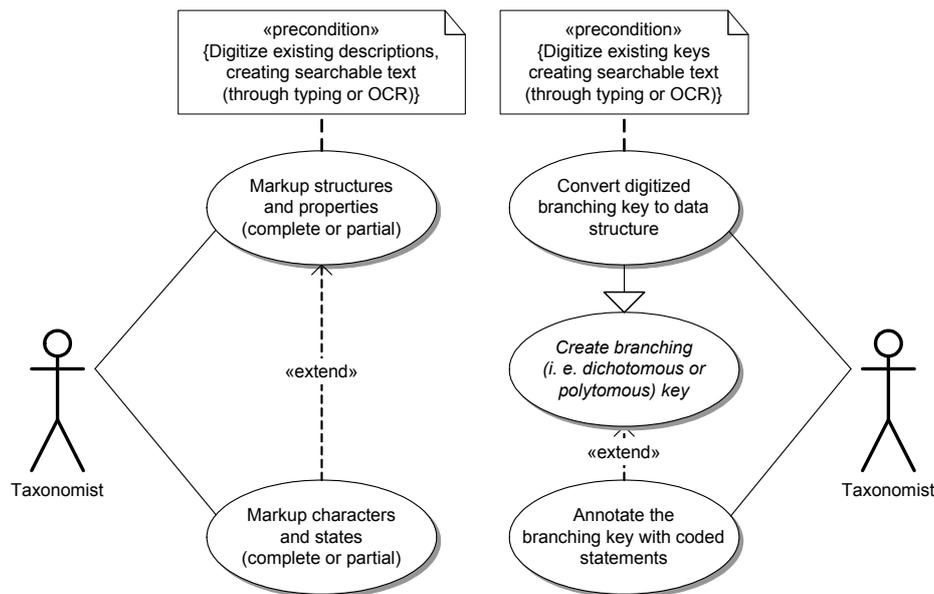


Figure 175. UML use case showing markup of legacy descriptions and keys. The use cases for natural language descriptions and for branching (i. e. dichotomous/polytomous) keys have a parallel structure. The task of converting printed descriptions or keys to digitized text format is very general and not expected to be part of a system. It is therefore modeled as a constraint (i. e., a note shape containing the constraint text in “{}”).

Conversion of digitized data to coded descriptions

Natural language descriptions and branching identification keys

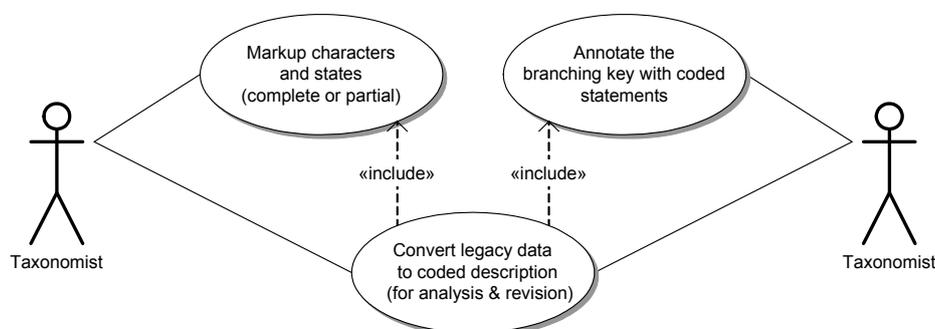


Figure 176. UML use case diagram showing an extension to the cases described in Fig. 175. Once the legacy data have been fully marked up, they can be used to create independent coded descriptions.

Coded descriptions can be generated both from natural language descriptions to which sufficient markup has been added (Fig. 176, left) and from identification keys containing markup or additional statements in coded terminology (Fig. 176, right). In the case of branching keys the resulting character \times class matrices will be only sparsely filled. However, many matrices resulting from natural language descriptions will also be relatively sparse, because the original descriptions have been abbreviated a) for all information considered “implicit” in the taxonomic scope and b) the remaining descriptive information is only reported insofar as it is diagnostic within the scope. Information not diagnostic in the scope may nevertheless be relevant when using data in a wider scope (e. g., for vegetative identification of plants).

Storing converted markup data (natural language or identification key) as a new block of coded data results in a duplication of data, which is generally undesirable. A static conversion is not a requirement for analytical processing of data (the coded form can easily be generated dynamically). However, if data are to be revised (data added, data not considered trustworthy deleted, or changed due to reinterpretation), a conversion to coded data is often beneficial. Whereas marked-up data (key or natural language descriptions) are usually expected to remain true to their source, the converted data can provide the basis for repeated revision and analysis cycles.

As mentioned above, the conversion of markup of digitized natural language description to coded descriptions is yet uncommon because very little data with such markup exist.

Data tables

A special case of source data are printed or digital data tables containing descriptive information. Digital descriptive table data may originate from self-designed simple description databases or, e. g., from user-defined description columns in the collection management software BIOTA.

If the data are well-organized, providing a structure approximately on the level of character or below, tabular data may be converted to coded descriptive data (or entered using one of the editor use cases described above). However, data tables may provide only a coarse structure (e. g., plant data organized into: “flowers, fruits, leaves, other”). The table cells then essentially contain fragments of natural language description. A similar situation occurs if data are organized approximately on a character level, but the description fragments in the table cells are highly variable or contain rich annotation text. In these cases a conversion to a coded description may not be the best strategy. Instead, the tabular data can often be converted to natural language descriptions, with markup preserving the structural information provided by the original table. The markup editor of the descriptive data application may then be used to selectively add further structure to the new descriptions.

Recording identification data

It is possible to store the information that is collected in the course of identification processes as descriptive data. Such data result in strongly incomplete, but usually diagnostic descriptions (p. 39). The process would require no additional effort when using identification tools. CSIRO Intkey already provides the functionality of converting identification criteria to descriptive data.

So far this data collection method has no practical relevance, but it may have great potential for the future. The major problem is one of data integration. A set of identification criteria and together with an accepted name has little relevance. It does not have the authority of a carefully revised “monographic” data set created by an expert and many identifications will be erroneous. However, if in the case of specimens in natural history collections, the data remain tied to specimen accession codes, these data would become a valuable data source.

The use case for recording descriptive data during the identification of collected objects is shown in the left-hand side of Fig. 177. Note that “Record identification data” is derived both from “Record descriptive data of collected or observed object” and from “Identify an object” (multiple inheritance).

Furthermore, the recording process may be coupled with a process issuing accession codes for identification data. If it would become good scientific practice that published identifications need to be accompanied by accession codes of permanently available identification data, this may be a solution to the problems with varying taxon concepts. Scientific journals or molecular databases (GenBank, EMBL, or DDBJ) could require the submission and publication of identification accession codes. Clearly, this may not be appropriate in cases where most identifications are “immediate recognitions” (i. e., the researcher would consult neither printed not computer-aided keys), but once computer-aided identification tools have become standard practice, in all other cases it would be a negligible extra effort with a huge benefit. See also “Future relevance: A proposal to record identification data” (p. 369) for an in-depth discussion.

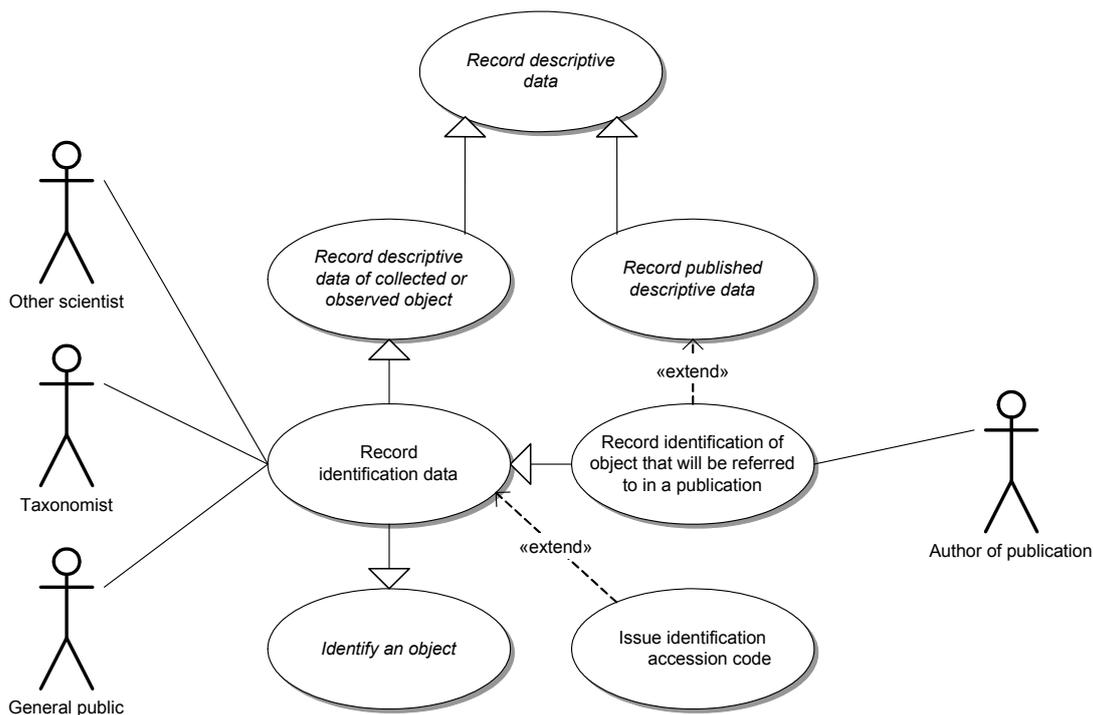


Figure 177. UML use case diagram showing the recording of descriptive data in the context of identifications. Note that the author on the right and the scientists on the left may be the same person in different roles.

233. To improve communication about identification processes, the detailed descriptive data created during identification processes may be permanently stored in “Identification-Banks” and made citable by issuing globally unique “identification accession numbers”.

Recording repeated original observations

The scientific process of obtaining descriptive data may be subdivided into the collection of original observation data and the subsequent processing and interpretation that creates synthetic information. Only the latter type of descriptive information is normally published in print, and most current descriptive data applications focus on this synthetic information. To improve the efficiency of biodiversity research and the collection of descriptive data, it is, however, highly desirable to model the entire scientific process. Thus the recording of descriptive data discussed so far may be specialized into the recording of:

- The original descriptive observation together with information how it was obtained and which fragments were obtained together (paired observations).
- Information that influences the process of obtaining a generalization. This includes interpretations, arguments causing removal of data like artifacts, recognized errors, removal of outliers (based on value alone), etc.
- The synthetic data that are the result of the aggregation/generalization process.

The most notable difference between original and synthetic information is that original information may occur repeatedly (multiple leaves, spores, fruiting bodies, algal cells of a single species and specimen). For a genetically homogeneous unit on a specimen (a specimen in a natural history collection may contain several units, e. g., several lichens growing on stone, or a parasitic fungus on a host plant) these can then be aggregated and summarized using descriptive statistical measures, like sample size and mode (the only measures for data on the nominal scale), median and extremes (data at least on the ordinal scale), mean, variance, etc. (data on interval and ratio scales).

Importantly, the recording of descriptive data may be a mixture of recording original observations for some characters (from which synthetic information is automatically generated), and recording only the synthetic information for other characters. For example, if spore measurements are obtained with the help of digital image processing (and no exchange format exists between the image processing software and the recording of descriptive data), usually only the summary statistics will be manually transferred.

A simple model of recording repeated observations and optionally storing aggregating data calculated from these (without recording additional interpretations, outlier removal etc.) is shown in Fig. 178. After recording repeated observations, the taxonomist communicates with the use case “Aggregate repeated observations”, creating the aggregated data. These may either be dynamic and used for analysis or report-generation purposes, or may actually be stored as data. In the latter case, the aggregation use case uses “Record generalized or aggregated data” as an extension. Alternatively, the taxonomist may communicate directly with “Record generalized or aggregated data” if the aggregation occurs outside of the system. See also “Raw data and data aggregation” (p. 83) for an in-depth discussion.

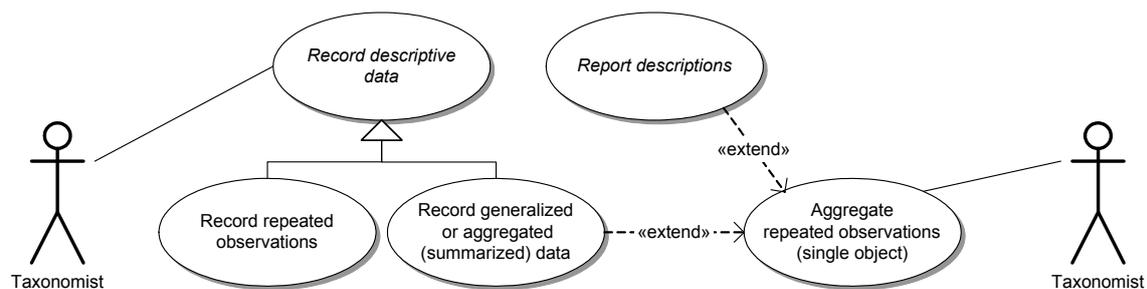


Figure 178. UML use case diagram showing recording of repeated original observation data. The use case “Aggregate repeated observations” interacts as actor with the recording of the aggregated data.

6.4. Information retrieval

The various forms of descriptive data discussed so far (natural language description without/with partial/with complete markup, coded descriptions, branching keys) are structured to various degrees and require different query mechanisms for information retrieval. A common topic of all information retrieval is the:

Selection of language and audience representations

Information retrieval is the area where the diversity of actors is most relevant. Even though summarized in most of the previous and following diagrams as “General public”, policy makers, agricultural agents, students, etc. (Fig. 161, p. 279) have different needs. In the SDD model (p. 20), differences between these actors are addressed largely through the audience mechanism (see “Multiple languages or audiences”, p. 282). Therefore, any use case that involves text labels or wordings includes the selection of audiences (Fig. 179). To simplify use cases where human interaction is, at least initially, not desirable (e. g., presenting an initial web page) it is desirable that the information model allows the project editors to define a default language and audience.

The algorithm for choosing the appropriate language and audience is not entirely trivial. It requires matching a preference sequence of requested languages (explicitly selected, internet browser language preference where an internet browser is involved, project defaults) with the set of available language/audience combinations. If the result set is empty, the first available language/audience for a given label/wording may be chosen. Audiences may be considered nested within languages (i. e., language choices are given priority. However, culture may be expressed as part of language (en-US, en-UK, de-DE, de-AT), and it may be desirable to give audience priority over culture. For example, American school children will probably be better served with British English definitions written with a school children audience in mind, than with the American English definition written for university-level expertise. In SDD it is possible to define an expertise level for audiences, improving the options for machine reasoning about the appropriate audience.

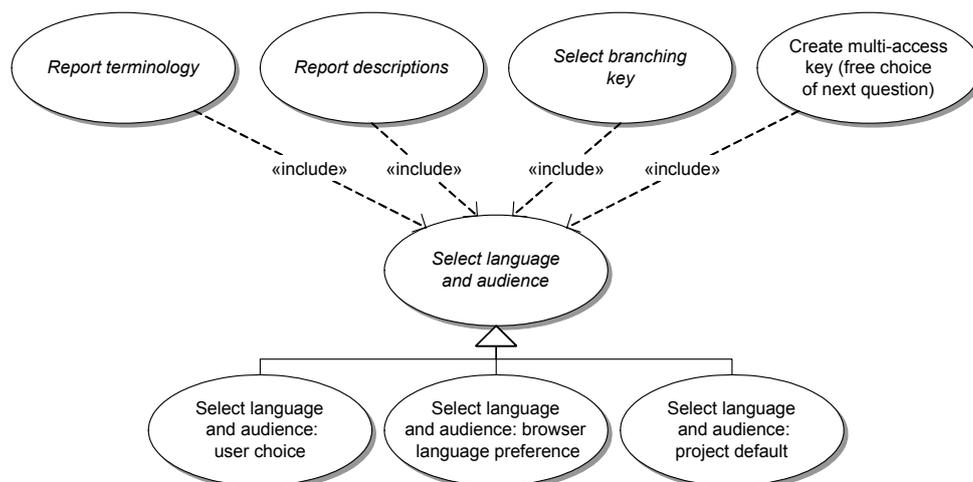


Figure 179. UML use case diagram showing a sample of use cases that include language and audience selection. Although the selection is required, it needs not be a manual selection if a project has defined a default selection.

Selection of branching keys

Many systems will store multiple branching keys so that a human actor first has to select the appropriate key (Fig. 180). The simplest selection will be a human choice based on the label of the entire key. Some care may have to be taken to be able to recognize entry and subkeys. However, users will often already recognize a taxonomic subgroup (e. g., a family in a plant key). In these cases, communicating the arrangement of keys to the actors is more important than denoting a single “entry key”. In most cases, the arrangement will be closely linked to the taxonomic hierarchy and an integration of the key selection use case into this hierarchy may be desirable. As discussed in “Linking multiple keys” (p. 259), keys usually have additional, non-taxonomic

scopes (geographical, ecological, methodological, etc.) so that multiple keys for a taxon must be expected.

The process of finding branching keys based on classes (taxa) that are keyed out may be based either on explicit key-metadata documenting the coverage of the key, or it may be automatically discovered, by analyzing the taxa keyed out in the key (Fig. 180, “Find branching key (based on classes keyed out)”). The major drawback of the latter method is that it requires recursive analysis of all subkeys, which may be impossible if some subkeys refer to external resources.

Taxon-based selection of keys (branching as well as multi-access) is especially relevant, because existing knowledge about major taxonomic groups (e. g., “duck”, “heron”, “bat”) is often difficult to express in terms of equivalent descriptive statements. Activating this knowledge thus not only helps during key selection, but also accelerates the identification progress. For example, a sophisticated identification software might be able to determine both an appropriate branching key, and the closest node in that key that still covers all taxa requested to be covered (right-most use case in Fig. 180). If the class (taxonomic) hierarchy is sufficiently complete and detailed, a hierarchical node and a list of taxa can be converted into each other (which may involve some widening of the taxon-selection, perhaps decreasing identification speed, but not quality).

A search for keys based on a list of classes (taxa) may also occur as part of another scenario, i. e., if an application supports switching between branching and multi-access keys (compare “Transferring progress information between branching and multi-access keys”, p. 262). In this scenario an identifying actor may have reached a certain point in a multi-access key, but then desires to switch to a branching key because selection of further characters seems to make little progress (use case not shown for this).

With regard to the additional scopes possible for identification keys, an interesting use case is the selection of a “virtual” key, which in reality is dynamically created from a larger data set to exactly suit the requirements specified by an identifying actor (“roll-your-own”). The selection of key characters may be based on character concepts (e. g., “reliable field characters”) or on separate databases containing checklists or distributional data, to select only characters and taxa appropriate for a given purpose. This approach can even be used for very small geographic areas like wildlife parks or other protected areas (e. g., as in the FRIDA-based keys, Nimis & al. 2005a, 2005b, Nimis 2007).

The resulting keys may either be used as computer-aided keys, or downloaded, printed and laminated by users (e. g., as in Haber 2006 or the “Rapid color guides” produced by the Chicago Field Museum, Illinois, USA; <http://fm2.fieldmuseum.org/plantguides/> or <http://www.fieldmuseum.org/animalguides/>).

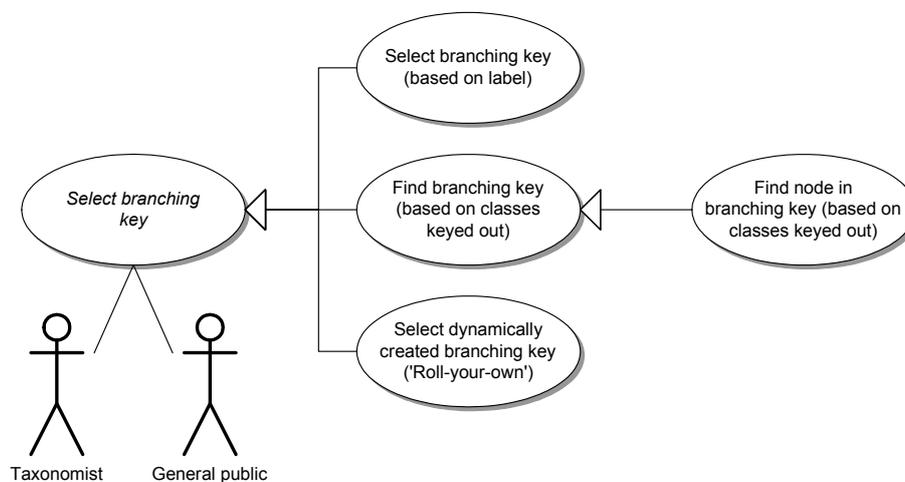


Figure 180. UML use cases specializing the abstract selection of a branching key.

Querying container level metadata

Both natural language and coded descriptions may carry metadata information on the container level. Examples are name or accession code of the described class or object, agents involved in creating the description, IPR data, dates, etc. In addition, some descriptions bear information about coverage and scope (geographic, seasonal, ecological, etc.) of descriptions. Although the class name is of primary importance, the other container-level data items are relevant for information retrieval as well (Fig. 181).

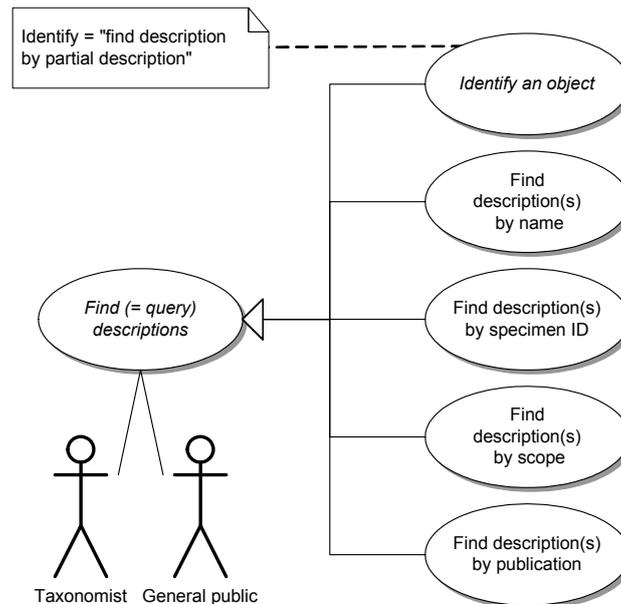


Figure 181. UML use case diagram showing identification as a specialized use case of queries to find descriptions.

Querying natural language description data

Natural language descriptions may be queried with methods designed for free-form text or with specialized processors making use of semantic knowledge derived from markup and the corresponding terminology definitions (the natural language syntax of biological descriptions is highly reduced so that query engines will gather little information from the syntax). In natural language descriptions with partial markup this connection to the terminology has already been made partly explicit, so that many ambiguous situations can be avoided. The use case is trivial so that no diagram is presented.

Querying coded description data

The most common form of information retrieval are queries against descriptive data stored in a structured format. Such "coded descriptions" are highly structured and can be conveniently retrieved using standard database querying techniques. The most frequent application of this is organism identification. Since identification use cases are a mixture of data retrieval and analysis, they are discussed in detail (p. 308) after the following major section, which includes the analysis use cases involved in identification.

6.5. Information review and interpretation

All data review, interpretation, and analysis use cases are modeled to implicitly start with a query (Fig. 182). This query may either be separate process, selecting a set of descriptions further processed in a following review or analysis, or it may be an essential part of the analysis.

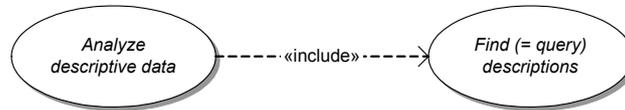


Figure 182. UML use case diagram showing that all analysis use cases include the selecting of descriptive data in a query (see Fig. 181 for non-abstract query use cases). The result of the query may, however, be the entire data set.

Analysis of data quality and completeness

A central use case of information review is that descriptive and terminological data are reviewed, errors are corrected, and missing data added. The use case “Add/change existing descriptions” reflecting this is trivial in itself; its implementation will usually not be different from the original data recording (Fig. 173, p. 291).

The relevance of this use case is that these actions often go hand in hand with data analysis and review. Many data analysis use cases may branch into editing use cases. Data analysis, data entry, and revision often occur in cycles (Fig. 183). Importantly, analysis of descriptive data often reflects upon the quality of the terminology and leads to revising the latter. The terminological editing use case is analogous to “Add/change existing descriptions”; Fig. 187 further down gives an example.

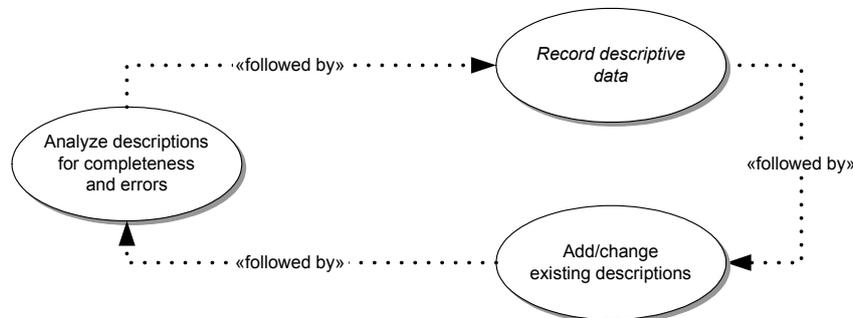


Figure 183. Diagram showing a common cyclic sequence of analysis and editing use cases. The “followed by” relation is not standard UML.

Examples of use cases to check data quality and completeness (Fig. 184) are:

- Test the presence of characters that have been declared mandatory or inapplicable.
- Analyze the usage of characters and states in descriptions.
 - Which terms defined in the terminology have never been used?
 - Which terms are rarely used? Are these terms specific to nodes in the taxon hierarchy? Does the terminology already contain generalized concepts that would also cover these?
 - Where is the usage of characters correlated (Fig. 185)?
 - Where is the distribution of ordinal categorical data unexpectedly discontinuous (Table 59)?
 - Where is the distribution of quantitative values unexpected (outliers, Fig. 186)?

Such tests may be restricted to constrained character sets or to branches of the taxonomic tree or to taxon sets defined by ecological criteria (if these are present as characters, this would be a character correlation analysis).

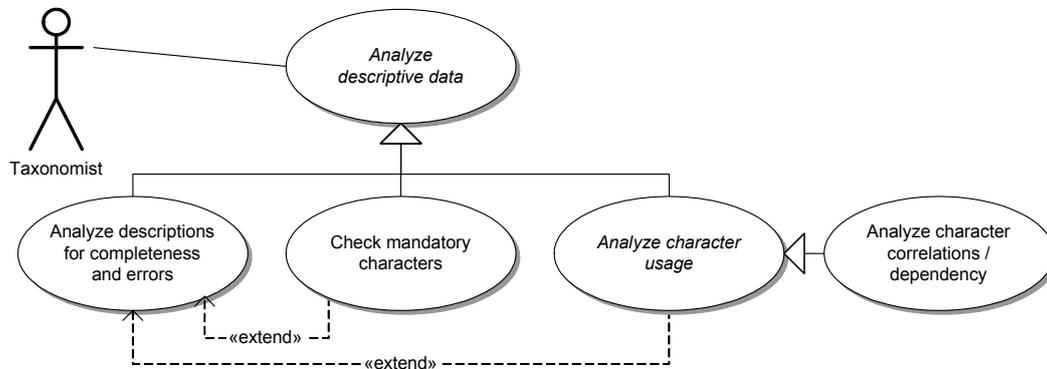


Figure 184. UML use case diagram showing examples of data quality analysis.

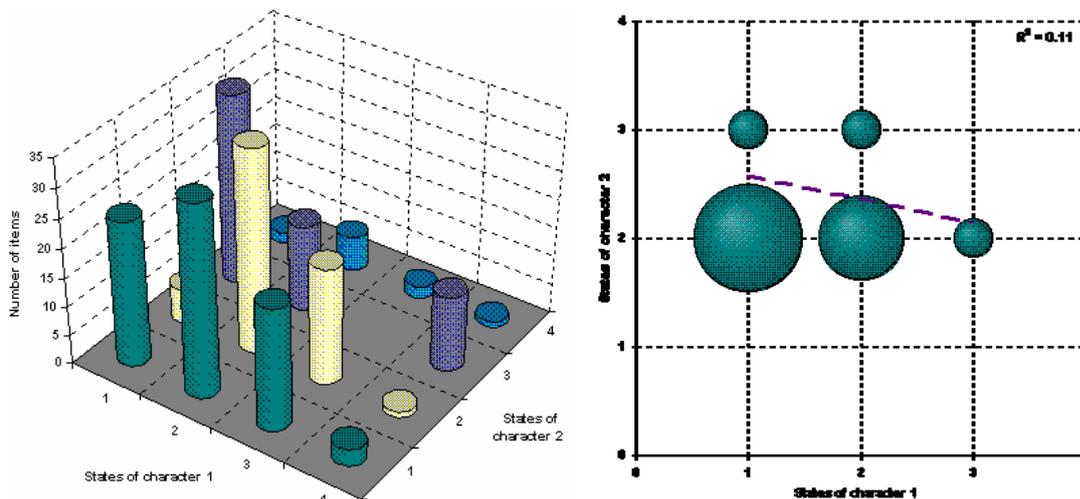


Figure 185. Visualizations of character relationships may also be used for quality control, especially to find outliers or seemingly impossible character state combinations. The figure shows 2 character state usage cross-tabulations graphs as generated by DiversityDescriptions. A bubble graph chart (right side) can display the same information as the more conventional 3D-bar graph. It can be easier to read, especially if many states are involved or if correlations of ordinal characters are expected. The area of the bubbles is proportional to the number of items which possess both states of two characters. See the DeltaAccess (DiversityDescriptions) user guide (Hagedorn 1999a) for further examples.

Table 59. Example of a quality control report for linearly ordered categorical characters (> two states). The items are arranged in order of their states and by Genus. The report highlights outliers (“Genus1 E”) or discontinuities (“Genus2 M”). Further, many experts for a group have a visual concept of similarity of the species they know. A review of this report may allow an expert to compare and visualize his concept of the species with the data recorded in the information system.

Character 1 By Genus	State 1	State2	State3	State4
Genus1	A	A		
	B	C		
				E
Genus2	N			
	M			M
	O	O		
Genus3	Y	X	X	
				Z

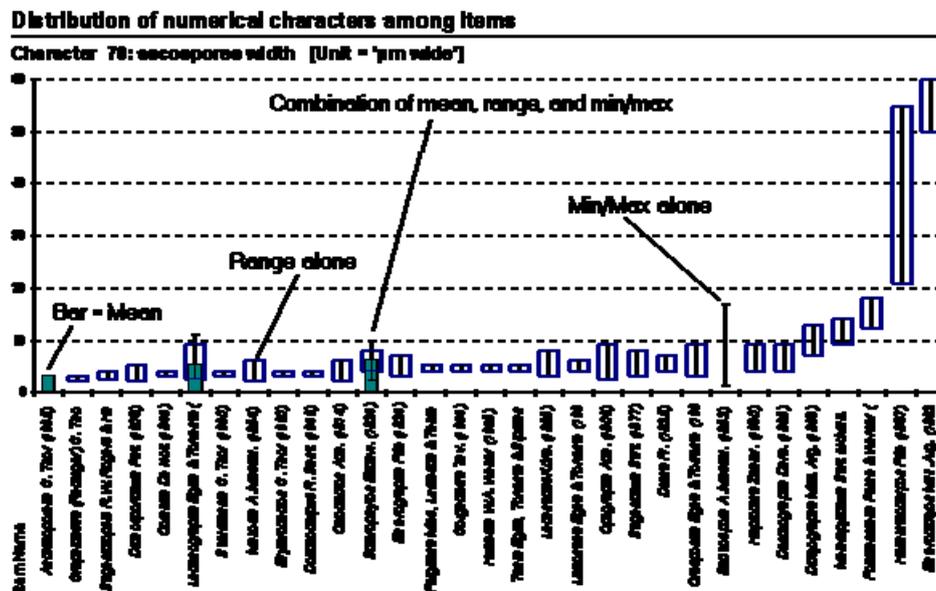


Figure 186. Example of a possible quality control report for quantitative characters (generated by DiversityDescriptions, except for the annotations). The graph displays mean (bar), normal range (box), and extreme values (lines). The items are sorted by the mean (or the middle of a range parameter, if missing). Outliers can be directly detected, and experts can identify unexpected similarity of taxa.

Analysis of character correlation

The analysis of character usage, correlation, and dependency is not limited to quality control measures. It is one of the central analysis techniques when attempting to use descriptive character data for other purposes. Most statistical analysis models (including phylogenetic analysis) assume that the character data (i. e., dimensions of descriptor space) are independent and identically distributed (“IID”-criterion). In reality, the expression of many characters is correlated, because of:

- Physical constraints. *Examples:* size and weight of any object; size of a container and number and size of contained objects (e. g., fruiting body containing spores).

- Different observation methods measuring the same property. *Examples:* object color, color of pigments embedded in object, chemical name of pigment, gene coding for the pigment.
- Properties are genetically correlated (“pleiotropy”). *Examples:* many enzymes are involved in multiple biochemical pathways; changes may influence multiple characters.
- Natural selection acts in parallel on multiple properties. *Examples:* different properties of eye-structures will be correlated.
- Character expression is correlated because of a common phylogenetic ancestry of organisms. *Examples:* Oaks (*Quercus*) are a largely tropical family that does not shed leaves. The European temperate oaks also have no active shedding of leaves although living in an environment where such behavior is a selective advantage for other species of deciduous trees.

On the one hand, knowledge of character correlations may assist data analyses that rely on an i.i.d. assumption by either refining the character terminology or by reducing the data set to include only a subset of (low-correlated) characters (Fig. 187).

On the other hand, character correlations point to interesting evolutionary or functional processes that may not yet be understood. Given a large data set, even simple statistical correlation methods may substantially improve the understanding of character space (see, e. g., Rambold & Hagedorn 1998). A major problem is, however, to distinguish character correlations purely based on common ancestry (as in the *Quercus* example above) from natural selection causes based on function and co-evolution. This can be achieved by methods that correct character correlation for a known phylogenetic history (see, e. g., Huelsenbeck & Rannala 2003 for a new Bayesian method and a review of other methods).

One problem of character correlation studies is that methods are usually suited either for quantitative or categorical data, and that joint analyses are difficult.

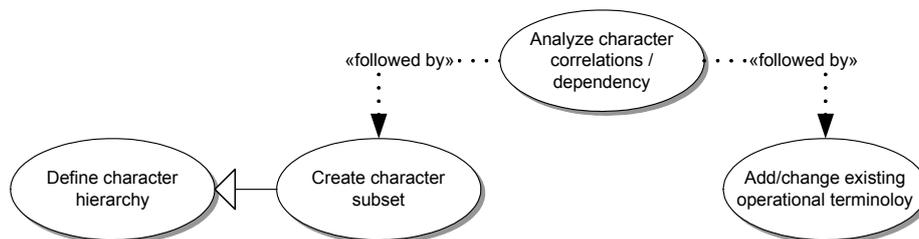


Figure 187. Diagram of potentially sequential process steps, showing that analysis of character correlations may either result in the definition of character subsets, including only one of a set of highly correlated characters, or in a refinement of the terminology itself. The “followed by” relation is not standard UML.

Analysis of character applicability

A special form of correlation between characters is character dependency, where some values of a controlling character fully define the values or applicability of another character. The most frequently implemented case is character applicability where values in a *controlling* character, if present in a description, define the *controlled* characters as inapplicable (i. e., they cannot have a value for logical reasons, see “Character applicability rules”, p. 76) for this description. Example: leaf shape cannot be observed in the absence of leaves.

Most character dependency situations can be deduced from logical reasoning, but character usage analysis can help to detect them (Fig. 188). This knowledge can be recorded as character dependency or applicability rules once it has logically been verified. Dependency rules do not add information content to perfectly correct descriptive data sets. However, most data sets contain factual and coding errors so that the addition of verified character dependency information provides an opportunity to check for a violation of these dependency rules (in separate analysis tasks, or as assistance during data entry).

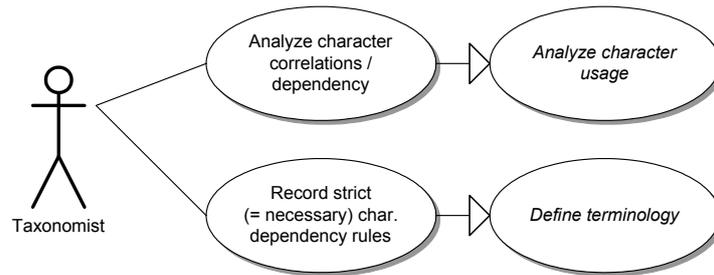


Figure 188. UML use case diagram showing the analysis of character correlations. Character dependency is a correlation recognized as strict or logically necessary that is permanently recorded as part of the terminology.

Aggregating descriptions

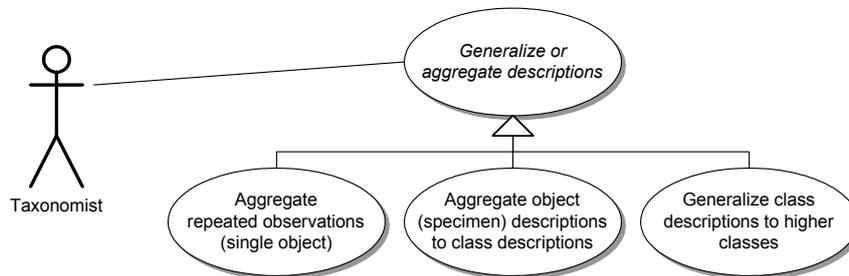


Figure 189. UML use case diagram showing the specific use cases of “Generalize or aggregate descriptions”.

A central feature of many descriptive data management systems is that they are able to automatically create new, “synthetic” or “dynamic” descriptions on the basis of existing descriptions (Fig. 189).

- Repeated observation on parts of an individual object (e. g., the measurement of 100 spores obtained from a single specimen; see “Recording repeated original observations”, p. 296) can be summarized using univariate descriptive statistics (mean, variance, standard error, etc.)
- From all object (i. e. specimen) descriptions that are identified to belong to one class (e. g., species or subspecies), a class description is generated. If later an object is re-identified, dynamically generated species descriptions will automatically change. This may involve calling a service (possible a web service of an external collection data base) to “Retrieve the current identification of object/specimen” (Fig. 190, left-hand side).
- From class descriptions, the generalized descriptions of classes further up in the class hierarchy may be generated. Like class descriptions based on object data, the use case to dynamically aggregate lower class descriptions to higher class descriptions, may have to invoke an update mechanism for the input data (Fig. 190, right-hand side).

Repeated measurements of the same property on the same object may be required by the methodology to compensate for instrument noise (example: fluorescent DNA quantification). This is ignored in use cases and the information models since it is rare and usually such measurements will be reduced to a single value.

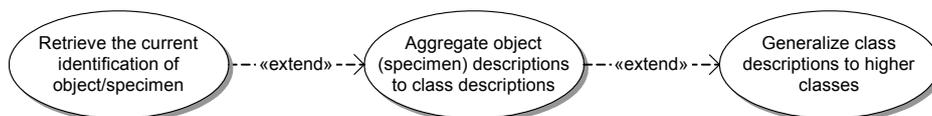


Figure 190. UML use case diagram showing a possible cascade of use cases invoking other use cases to update information before aggregating it.

Creation of class hierarchies

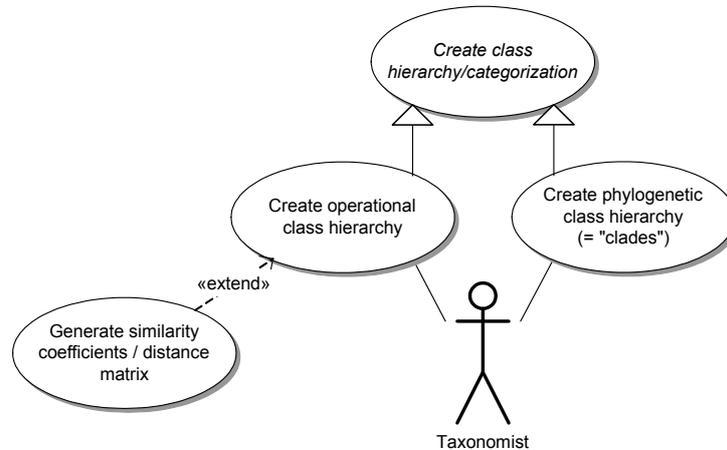


Figure 191. UML use case diagram showing the categorization of objects through data analysis.

Descriptive data (morpho-anatomical as well as molecular) are also extensively used in the definition of taxonomic classes (e. g., species and higher ranks) and class hierarchies (e. g., genera in families in orders). A classification or categorization of biological organisms may be operational or phylogenetic (Fig. 191). Operational classifications always depend on a somewhat arbitrarily chosen guiding principle. This may be overall similarity, or some guiding principle like ecological adaptations (succulents, climbers, etc.) or symbiotic status (pathogen, parasite, hemiparasite, pollinator, predator, prey, etc.). Some classifications formerly considered taxonomic are now maintained as operational classifications (bacteria, algae, protists) because of their history and continued usefulness in structuring the diversity of life.

An operation classification based on overall similarity may be supported by an analysis procedure generating similarity coefficients or a distance matrix.

Very few operational classifications are *not* based on descriptive data; examples are classifications reflecting the mythological significance of organisms and medicinal classifications.

Phylogenetic classification is unambiguous in that it attempts to reflect the evolutionary history of life on earth. In contrast to operational categories, phylogenetic categories have a predictive value. Provided appropriate inference methods have been used (see, e. g., Felsenstein 2004), data not yet included in the process of categorization are more likely to conform to the phylogenetic hierarchy than with any operational hierarchy. Since inheriting a character from ancestors is more likely than independent and convergent recreation, new data are likely to follow the same pattern as previously analyzed data.

Phylogenetic classification is almost exclusively based on descriptive data. The only information about evolutionary history is the fossil record, which in most organism groups is poor and fragmentary.

Analysis of character evolution

A special form of character analysis is the phylogenetic reconstruction of character evolution. Here the evolution of a single character is compared with the general phylogenetic hierarchy of the studied taxa. A simple form of doing this is to display (usually graphically) a phylogenetic tree and overlay selected character information by means of color, symbols, or different line formats in the tree (Fig. 192).

More exact statistical methods are available as well. One form of studying correlation between characters and the phylogeny might be to perform a Mantel matrix correlation test on two distance matrices, one based on distances in the phylogenetic tree, another based on a single character. Other methods are implemented in the “Component” program (Page 1993, Page 2001b). “Component” compares two trees, so the selected character would be used to create a single-character operational categorization (Fig. 193). It was originally designed to study co-evolution by comparing host-parasite phylogenies. If organism interactions are considered descriptive data (compare p. 30), the study of co-evolution is yet another analysis use case (not shown).

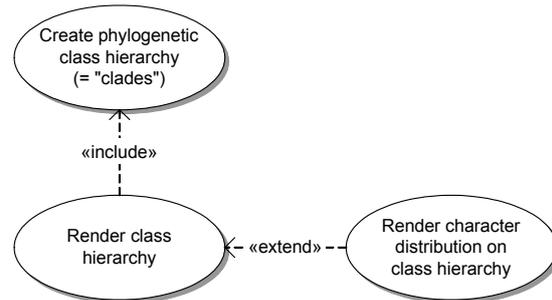


Figure 192. UML use case diagram showing the review of character distribution superimposed on a phylogenetic hierarchy.

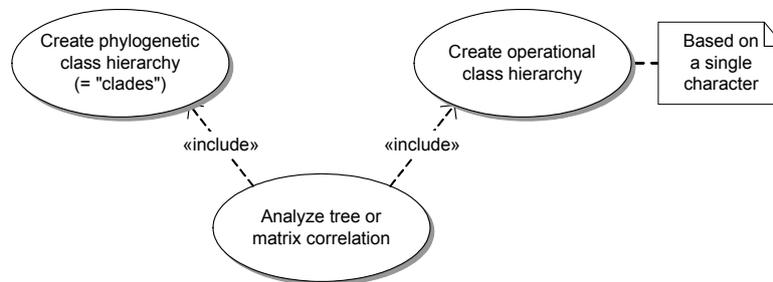


Figure 193. UML use case diagram showing the analysis of character distribution by tree correlation.

Creation of diagnostic subsets

For a given group of descriptions (e. g., a taxon, results after a given identification step, or an entire data set), all characters can be removed that are no longer able to differentiate between members of this group. A description that is thus constrained is often usually called “diagnostic” (p. 39) and contrasted with a “taxonomic description” containing all available information. However, a description is always diagnostic to some set of taxa or objects; the qualification “diagnostic” is a function of the set, not of individual descriptions.

It is possible to reduce the number of characters further, to achieve a *minimal diagnostic character subset* using a find-best-next algorithm (see above), or by performing and evaluating an exhaustive search (Pankhurst 1983). The abstract use case “Select diagnostic description subset” refers to the selection of either the total or the minimal diagnostic subset (Fig. 194).

Associated report-generation use cases are shown further down (Fig. 207, p. 316).

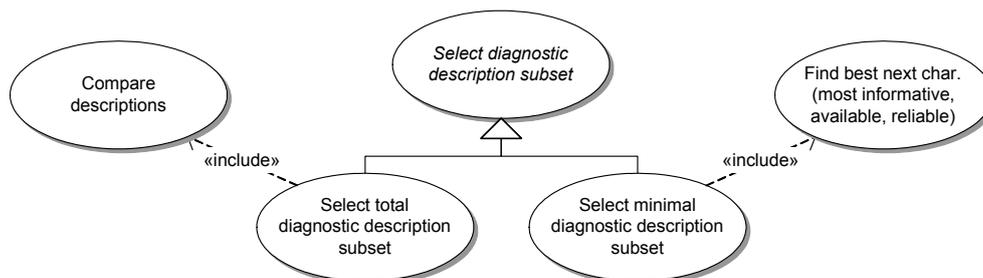


Figure 194. UML use case diagram showing creation of diagnostic subsets.

6.6. Identification

Class (i. e. taxon) identification may be described as a query for the class name of an object based on partial information about the description of the object. Identification using descriptive data is a peculiar type of information retrieval in that the amount of data used (perhaps dozens of characters) during the retrieval process by far outweighs the amount of data actually retrieved (the class name = taxon name). Furthermore, identification typically includes analytical processes, such as calculating class similarities, or character guidance measures. Because of this mixture of information retrieval (p. 297) and analysis (p. 301) identification is discussed under its own heading. See the previous chapter “Identification methods” (p. 229) for further background information on identification.

The process of identification is analyzed here from the perspective of typical computer-aided identification tools (Intkey, Lucid, etc.). A re-analysis of use cases from the perspective of expert systems may be useful (see especially Fajardo Contreras & al. 2003).

Identification keys

Identification primarily uses special query methods based on descriptive information commonly called “identification keys”. The major types of keys are the branching key and the multi-access key (Fig. 195). In branching keys the identification path is fixed and the user is guided along this path. In contrast, multi-access keys allow free choice which identification criteria (characters) are to be used in which sequence. More information about these and other forms and variations of identification keys is found in “Structural classification of identification keys” (p. 233).

The identification process may be facilitated by restricting the result sets based on information available at the description-container level (Fig. 181, p. 300), especially information about taxonomic and geographical scope. If a geographical scope is given, identifications would preferably use a matching scope, but a description from a different area may also be useful and an application may offer the user such data in addition to data strictly matching the scope.

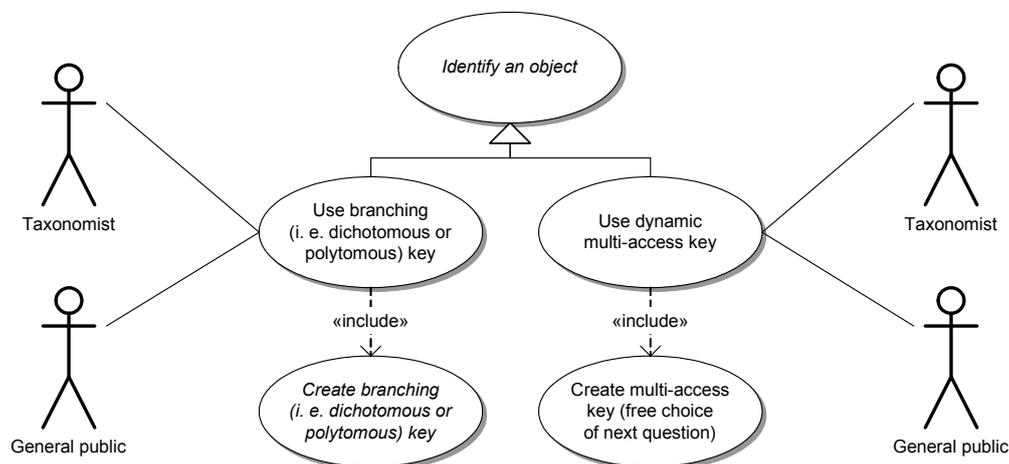


Figure 195. UML use case diagram showing the identification of an object through branching or multi-access keys.

At any point during the use of keys, terms may be looked up in the glossary part of the terminology (Fig. 196). This may be enabled both for natural language phrases present in the statements in branching keys, and for coded terminology entities. In the first case the lookup is based on a string comparison with glossary terms, in the latter case it is preferably based on explicit links pointing to glossary entries that the designers of the operational terminology have defined.

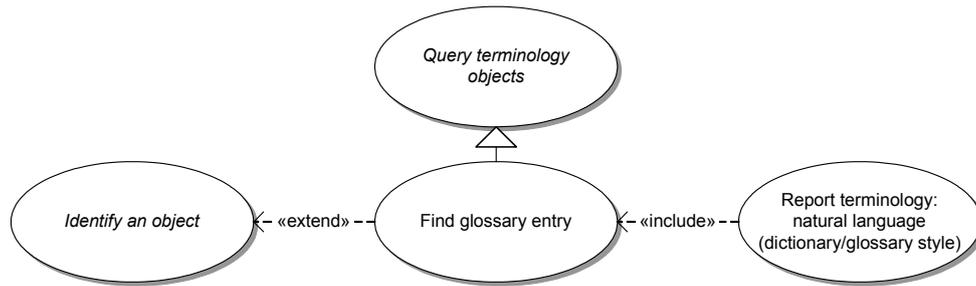


Figure 196. UML use case diagram showing the extension of identification with lookup and report generation of terminological definitions.

Switching between branching and multi-access keys

An application may not only allow an initial choice between branching and multi-access keys, it may also allow starting with a branching key and switch to a multi-access key when the user cannot make a decision in the branching key (Fig. 197). Under certain circumstances it is possible to transfer at least part of the identification progress from one key into the other. Switching options may be provided through menu or other commands, or simply dynamically integrated into the rendering of the key itself (e. g., adding “cannot decide this question” to each key couplet in a branching key). The topic is discussed in more detail in “Transferring progress information between branching and multi-access keys” (p. 262).

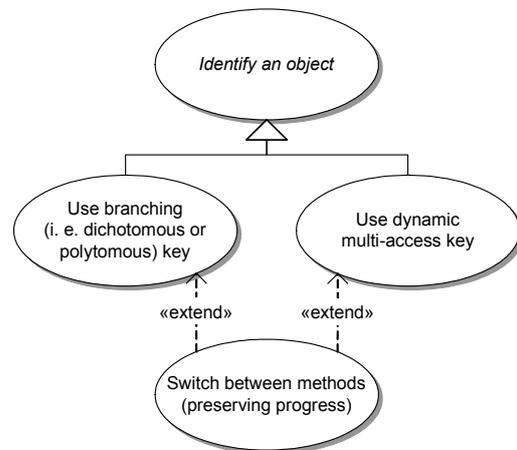


Figure 197. UML use case diagram showing switching between different identification methods (user interfaces).

Confirmation of identification

It is very desirable for an identification system to offer a confirmation or confirmation phase (compare p. 232) after the class (i. e. taxon) name has been identified. Initial identification results, especially when the user is inexperienced, are often erroneous. Questions in branching or multi-access keys may have been misinterpreted, and occasionally even coding errors exist in the descriptions. Two dominant confirmation approaches exist:

- Represent the class (i. e. taxon) as detailed as possible (Fig. 198 left). This can be achieved by generating a natural language description, offering images or other media resources (preferentially those that have not been used during the identification process), or by rendering a complete species page (see p. 319).
- Broaden the identification result set (Fig. 199) by increasing the error tolerance of the identification process (e. g., using wider margins of error), find taxa partially matching the identification criteria (i. e. allowing some character statements to be incorrect), or by including similar descriptions using statistical similarity methods. To improve the results, a similarity search should be dynamically generated and constrained to those characters answered during the identification process. The broadened result set may then either simply be presented to the user for browsing, or a method may analyze which questions have not yet been answered and are diagnostically most appropriate within the broadened result set. Essentially, this creates a dynamic identification key (“check key”, Payne & Preece 1977) and it may be presented as a branching or multi-access key to the user (Fig. 198).

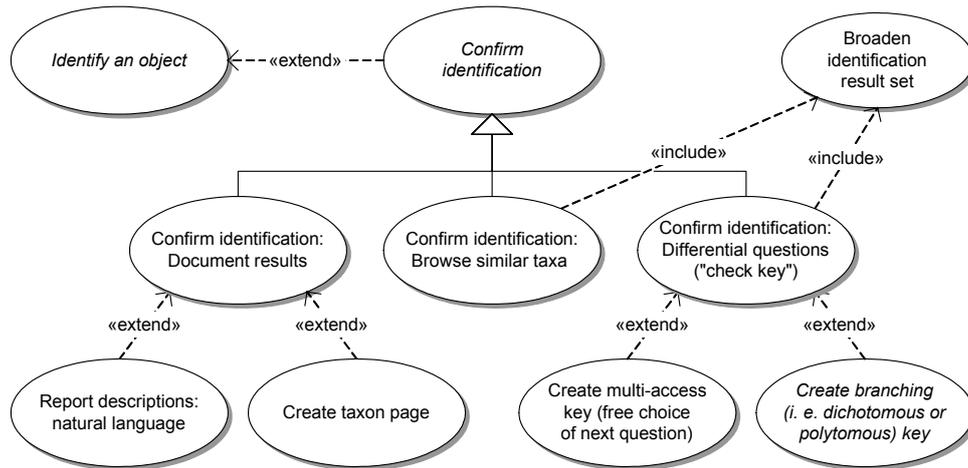


Figure 198. UML use case diagram showing the confirmation phase after an initially successful identification.

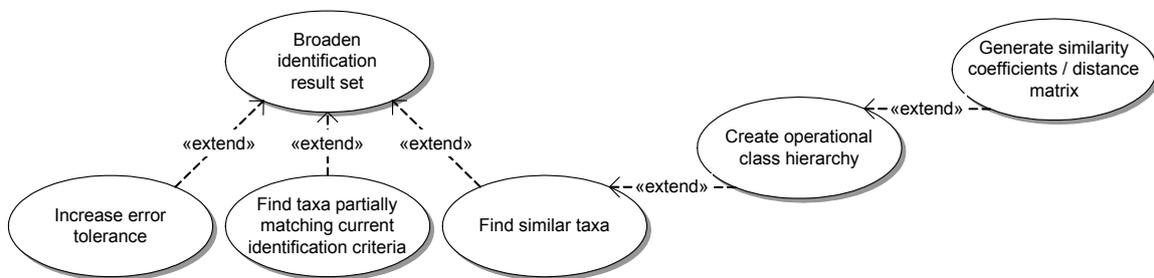


Figure 199. UML use case diagram showing potential methods to broaden the result set of an identification.

Failure of identification

If the identification process fails to result in a class name, or if the initial result is rejected after a verification phase, the following possibilities exist:

- The person making the identification made errors, for example:
 - the object may have been observed not carefully enough,
 - a question in the key has been misunderstood/misinterpreted,
 - an error was made in handling the identification system (scoring wrong check box, following wrong lead number in branching key).
- The data set on which identification is based is erroneous; although it contains the correct results some data are wrongly entered.
- The class that would be the correct result is not covered by the identification system. In this case:
 - the class concept may have already been described and is covered by other (not necessarily digitized) descriptive data, or
 - the specimen belongs to those species that have never been described or named.

As pointed out in the introduction (Fig. 1, p. 14), the majority of organisms on earth belong to the last category. However, the taxonomic and geographic distribution of biodiversity knowledge is extremely uneven. In many cases (e. g., mammals or European insects) the other possibilities are much more likely. To avoid the problem of describing the same taxon multiple times (compare Fig. 2, p. 14), the identification system should provide a way to check or review negative identification results (Fig. 200).

When checking potential reasons for a negative identification result an obvious start is a review of the identification criteria used. Often the user will even be aware that certain criteria have a higher likelihood of misinterpretations than others and may delete these criteria.

Interestingly, the next use case “Find taxa partially matching current identification criteria” often helps both with errors made by the person making the identification and with errors in the data set itself. It could take the form of reducing the set of criteria by creating multiple subsets that contain not all criteria. The number of criteria removed from the identification process could be increased until the identification result set is no longer empty. Many other algorithms are possible. On the user interface the positive identification results could be listed together with the criteria that originally excluded them from the result set.

Unfortunately the use case “Find related identification resources” will in practice be rather vague. Currently this will probably take the form of browsing through printed or digital resources, using bibliographies, or unspecific search mechanisms in the hope of finding a useful identification resource. It is hoped that, perhaps in the context of GBIF, more appropriate methods will be developed. The simple knowledge of sets of taxon names that can be differentiated with a given identification resource would allow an efficient search in many cases (if higher or closely related taxa can be identified). Even more desirable (but also more difficult) would be to define a set of essential criteria that can be searched across all identification systems known to an indexing service.

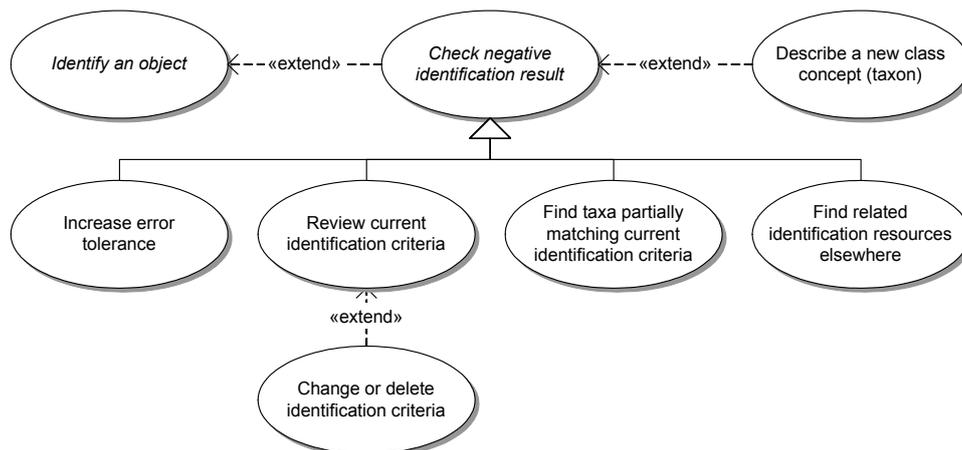


Figure 200. UML use case diagram showing the verification step after identification.

Identification of potential taxon concepts

A special identification use case is that both a class name and its description are already published, and the identification attempts to find the closest taxon concept published elsewhere. This requires that previously for each identifiable taxon concept an individual description has been recorded. The class name for the taxon concept that is to be identified is simply ignored and its description entered as identification data into a normal identification process.

This will normally return only a class name. The use case “Identify a taxon concept” (Fig. 201) would then continue with a less error-tolerant query algorithm and use the individual concept descriptions (rather than the generalized class descriptions normally used). No implementation of this use case is known so far.

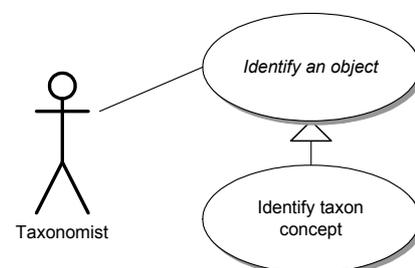


Figure 201. UML use case diagram showing the identification of potential taxon concepts as a specialization of general name-based identification.

Creation of branching keys

Branching keys (i. e. dichotomous/polytomous keys; compare p. 233) are a special form of hierarchically organized descriptions. Similar to diagnostic subsets, they attempt to capture the minimum of information necessary to differentiate the classes of interest (usually taxa, rarely races or populations). However, the diagnostic set is interpreted strictly locally (i. e., limited to the current branch of the key) so that branching keys usually contain even less information than diagnostic descriptions. This brevity is both the advantage and disadvantage of branching keys. Identification often fails if a characteristic cannot be observed (because, e. g., it is not present in the specimen, not expressed at the time of observation) or is interpreted incorrectly. Nevertheless, branching keys implicitly capture a huge amount of information about the availability, reliability, and selective power (information content) of the characters used. They are a valuable resource. Furthermore, taxonomists may intuitively create excellent branching keys expressing this knowledge, but may not be willing to collaborate on a more structured expression. The use of branching keys in descriptive information systems is therefore not restricted to legacy data.

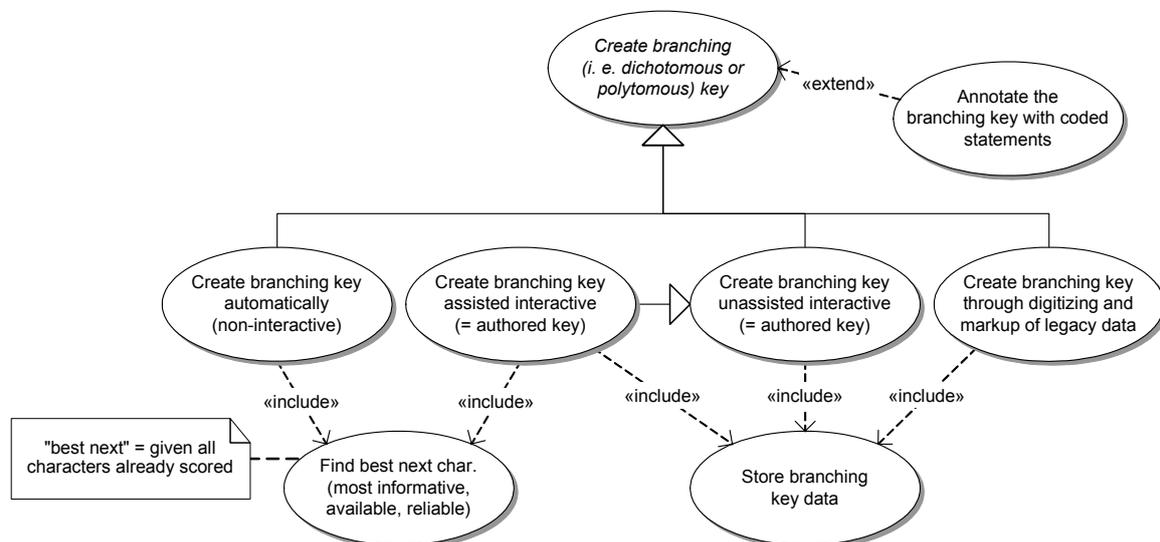


Figure 202. UML use case diagram showing that branching keys may be created automatically, assisted, fully manually, or through markup of legacy data.

The specific use cases of creating branching keys (Fig. 202) are *automatic, non-interactive creation*, *assisted interactive mode*, *unassisted interactive mode*, and the case that legacy data are *digitized and marked up*. The first two cases depend on some form of dynamic object categorization such as the find-best-next algorithm (see next section) that is either automatically accepted in the automatic creation, or used as suggestions to the author, who may decide otherwise. The assisted creation is essentially identical with the unassisted key editor, differing only by offering recommendations.

Automatically created branching keys will usually be generated on the fly and are not stored (they may be exported to exchange formats, however). All specific use cases may include an extension to add structured coded statements to the natural language lead statements in the key. In the case of automatic or assisted creation, this can even be accomplished with no or little user intervention.

Branching keys normally contain free-form text either as statements (lead propositions) or as question/answer pairs. Media resources may be used in addition or occasionally even instead of text.

To support the use case of switching between branching and multi-access keys while preserving identification progress (Fig. 197), the text in a branching key may be restated in coded termi-

nology (either as markup around the natural language text or as separate coded description data). One use case scenario would be a branching key editor where primarily free-form text is entered and coded statements may optionally be added. Another scenario might be a branching key editor where the free-form text is automatically generated on the basis of selected character statements and Boolean operators to guarantee that all information is available in coded form. The second scenario probably ideal for newly created keys, but usually not practical to reproduce existing legacy keys (such keys often contain statements that are difficult to interpret under a different terminology).

Dynamic character recommendations for identification purposes

To achieve fast and good identification results, characters may be ranked according to their suitability for identification. Such a character ranking may be used to automatically create branching keys, to recommend characters during interactive creation of branching keys, or to recommend characters to users of multi-access keys. In the latter, the ranking usually defines a display sequence and does not prevent the user from selecting a character with a poor rank.

A simple form of character ranking is based on authored information about character ratings (or “weights”) expressing their *Convenience*, *Availability*, and *Reliability*. More complex ratings also involve an analysis of character usage in descriptions. Given a set of descriptions, these algorithms (labeled “Find next best character” in Fig. 203) rank the characters based on how fast identification would advance. The set of descriptions may initially be all classes or reflect a subset based on the taxonomic class hierarchy (e. g., actors may have selected to identify only within the order Erysiphales because they recognized the characteristics of this taxon). Later, the subset reflects the identification progress, i. e., contains only those classes fulfilling the identification conditions. The topic of authored and algorithmic character guidance is discussed in detail on p. 267.

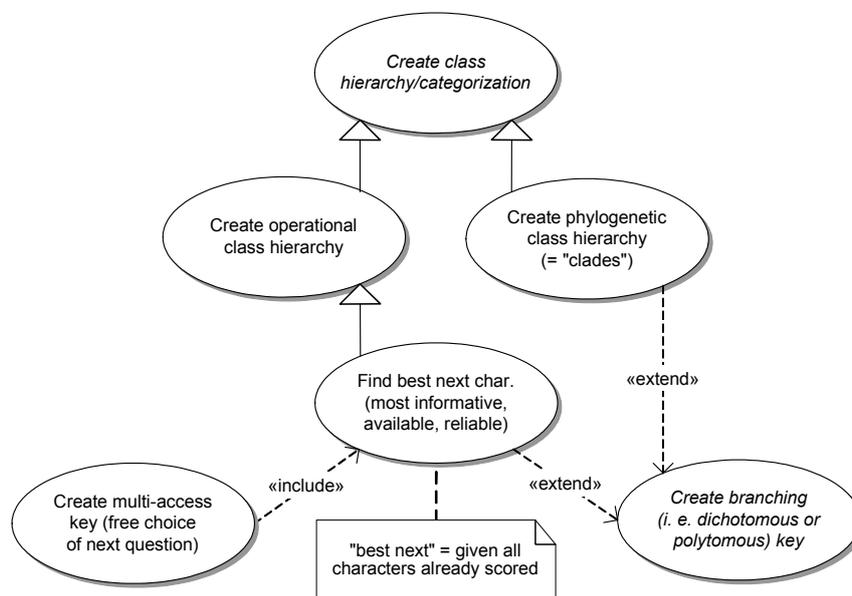


Figure 203. UML use case diagram showing the creation of interactive or branching keys as a special case of object categorization.

Character recommendations for identification purposes based on the phylogeny

In addition to character guidance based on an operational categorization (such as the “Find best next” algorithm), phylogenetic categorizations may be desirable in the assisted creation of branching keys. This is related to the concept of “phylogenetic keys” (= “natural keys” or “synoptic keys” in the traditional sense; compare p. 400). However, whereas such keys try to fully reflect the classification of phylogeny of organisms, the important aspect of phylogenetic categories in this context is that they correspond to concepts already known to experienced users. Using such concepts may therefore be desirable even if this does not occur throughout the key.

In most cases a major part of the identification path is memorized by human users. Very few identifications start without such knowledge and include the entire spectrum of organisms (viroids, viruses, bacteria... to animals). Instead, memorized concepts are used in the selection of an appropriate key to start with (see “Selection of branching keys”, p. 298). Exploiting this knowledge further inside a key makes usage of the key faster and safer.

Basing the primary motivation for characters that correspond with classification on the experience of users introduces the problem that some users will prefer traditional concepts, others the newest phylogenetic understanding. However, the history of biology has shown that even though non-phylogenetic taxonomic traditions may persist within one generation of biologists, the next generation will accept the new phylogenetic concepts and memorize them. Exceptions occur only where a phylogenetic classification can in practice not be recognized on the basis of descriptive features of organisms, or where a concept is already deeply embedded in non-scientific language (e. g., in the case of “algae”, or “fungi” including oomycetes and myxomycetes, etc.).

The use case for assisted creation of branching key (Fig. 202) may thus be more appropriately modeled to also include phylogenetic methods (Fig. 204). The designer of the branching key would then be able to select whether the next split should reflect a phylogenetic or an operational categorization. A useful feature in this context would be some metric that visualizes how much better an operational categorization is over a phylogenetic one. If the difference is relatively small, a phylogenetic split may be the better choice even if the resulting lead statements are more complex and require the use of ‘and’ or ‘or’. To achieve the desired effect, the correspondence with phylogenetic categories must be explicit, i. e., in addition to the descriptive statements the class names are cited.

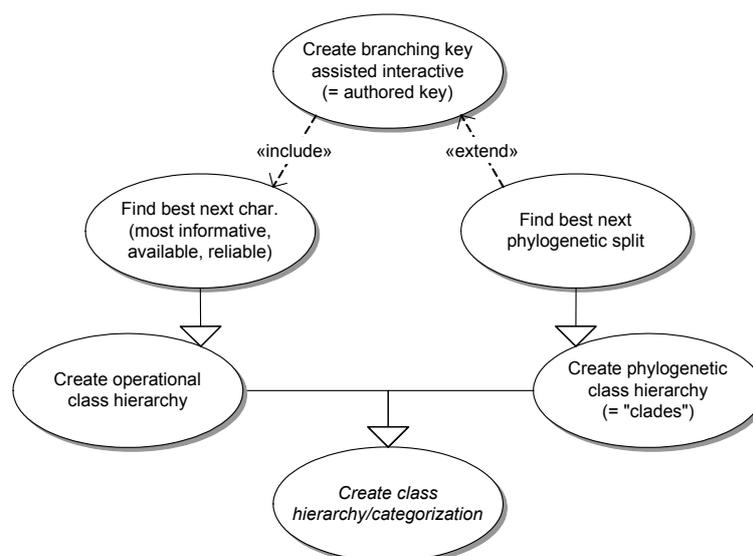


Figure 204. UML use case diagram showing that phylogenetic categorization may be relevant in the creation of branching keys.

6.7. Information application

Report generation

Similar to the analysis use cases, all report-generation use cases are modeled to implicitly start with a query (Fig. 205) that returns a set of objects (characters, descriptions, etc.). Although the plural form is used in use case names (“Report descriptions”), the use cases should always be considered applicable to sets containing only a single object. The special case that an entire project or data set is reported (or exported) may similarly be considered a query, containing the project as its only restriction.

As discussed above (p. 298), the selection of language and audience for the report is considered to be part of the query.

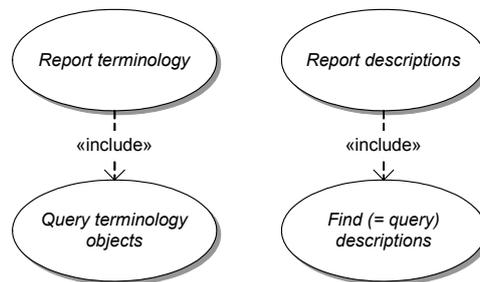


Figure 205. UML use case diagram showing that report generation includes finding (or selecting) the descriptions to be reported. See Fig. 181 for the specialized cases of “Find descriptions”.

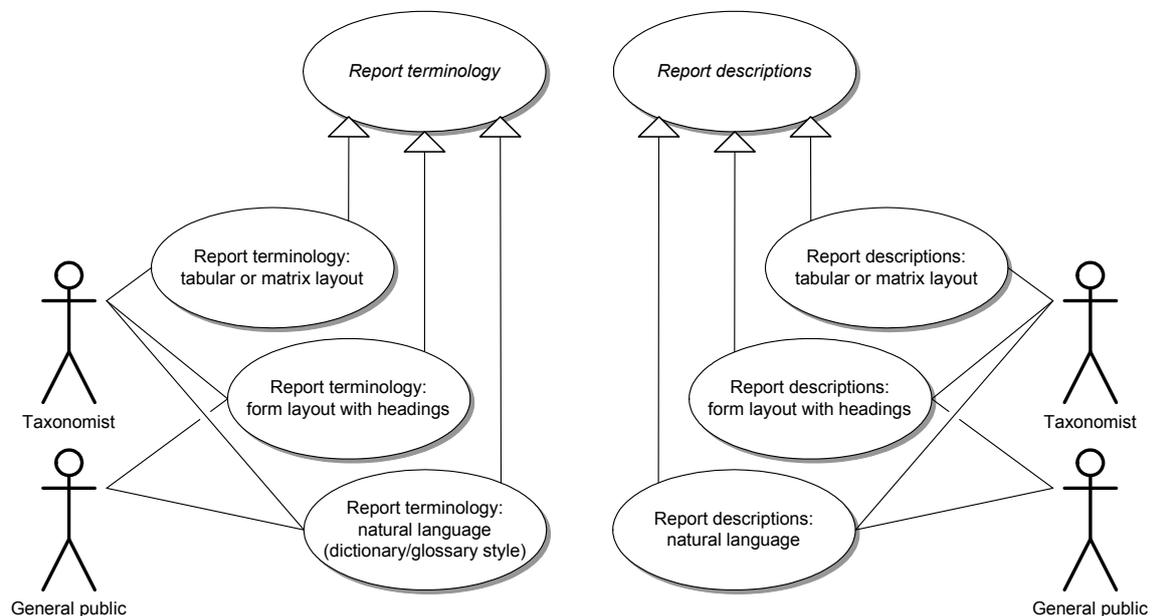


Figure 206. UML use case diagram showing the report generation of descriptive data and the associated terminology. Tabular or matrix layout is assumed to be of interest primarily to the taxonomist. However, no real reason exists not to offer it to the general public as well.

Fig. 206 displays a selection of possible reports both for terminology and the descriptions. The terminology reports can be produced based on either operational or ontological terminology; the latter is especially well suited for the dictionary/glossary-style terminology report. The description reports all require data supplied from terminology. The generation of natural language de-

scriptions based on coded description furthermore requires that appropriate wordings for character groups (i. e., nodes in the character hierarchy), characters, states, modifiers, etc. have been defined. Furthermore, different report-generation scenarios exist, depending on the kind of descriptions available:

Descriptive data available as ...

1) coded descriptions

2) natural language description

a) with complete markup

b) without or only with incomplete markup

If only 1) is available, all report variants shown can be delivered (assuming that a natural language generation process or service is available). If 1) and 2) are available, a choice between an original natural language description and a generated natural language description may have to be made. The two descriptions are based on independent data and will usually differ in content.

Most applications will be able to treat case 2 a) as identical with case 1). A natural language description with complete markup is relatively similar to a coded description. However, a number of data quality constraints present in coded descriptions cannot be applied to markup data, because the latter may contain coding errors or inaccuracies. It is therefore possible that the report-generation method will raise an exception if the structured reports are requested.

In the case of 2 b) usually only outputting the original natural language description as a report is meaningful. Although the other reports may be applicable, they would substantially distort the description content.

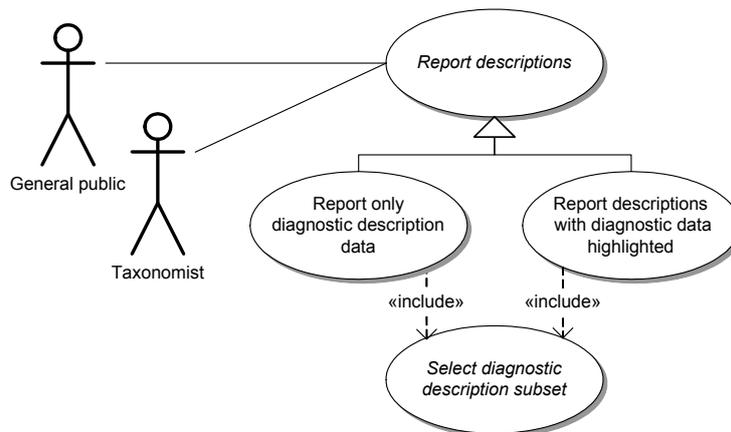


Figure 207. UML use case diagram showing the case of diagnostic descriptions that are abbreviated to contain only characters differentiating between members of a limited item/taxon group. See Fig. 194 (p. 307) for elaboration of “Select diagnostic description subset”.

Two special report use cases (Fig. 207) are associated with diagnostic descriptions (i. e., descriptions containing only information differentiating the taxa or objects in a set; compare “Creation of diagnostic subsets”, p. 307). Diagnostic descriptions may be rendered in any description format shown in Fig. 206. Tabular layout is especially attractive for small sets of diagnostic descriptions because it simplifies comparisons. A continuum exists between diagnostic tabular description reports and printable multi-access keys (see below). An occasionally used variant for very small taxon sets is a tabular arrangement of natural language fragments (broken down by some heading level), resulting in “synchronized” natural language descriptions rendered side by side.

A variant of descriptions containing only diagnostic data are full descriptions, where the diagnostic parts are highlighted (Fig. 207), e. g., by italicizing them (e. g., Watson & Dallwitz 1991).

Rendering printable multi-access keys

Computer-aided multi-access keys are considered under section 6.6 above. However, printable multi-access keys (p. 249) have a very distinct format and are closely tied to reporting of descriptions (Fig. 208).

Multi-access keys in *taxon-list style* (Fig. 130, p. 250; some or all characters are coded and listed for each taxon) closely corresponds to description reporting, except that some characters are expressed through “character formulas” instead of natural language text. This style often uses completely or partly tabular arrangements, so that the border to reporting descriptions in tabular or matrix layout (Fig. 206, p. 315) is fluent.

Multi-access keys in character-list style (Fig. 129, p. 250) organize the descriptions by character, with taxa or taxon codes listed under each character state. Morse (1974) implements a use case “preparation of inverted descriptions”, which is essentially identical to a multi-access key in character list style.

A printable multi-access key may be prepared to be used both computer-aided and in print, by providing hyperlinks in addition to links that can be followed in print. However, in contrast to branching keys (see below), printable hyperlinked multi-access keys are not as useful (identification is the result of “intersecting” multiple links, which is not available in standard hyperlinking).

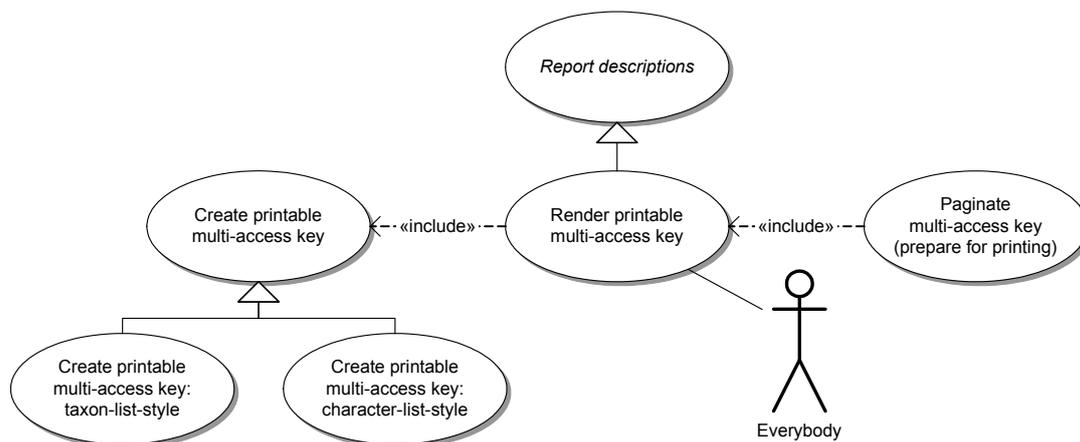


Figure 208. UML use case diagram showing the rendering (and optionally pagination to prepare for printing) of printable multi-access keys (in two strongly distinguished styles, each of which may have subvariants).

Rendering printable branching keys

Reporting branching keys (dichotomous or polytomous identification keys) is a highly specialized form of report generation not yet included in Fig. 206. Branching keys are essentially directed acyclic graphs and can be presented in various formats (nested, linked, or tabular; see “Printable branching keys”, p. 242, and especially Fig. 120ff, p. 244).

Most description reports generated for on-screen use are also easily printable. In the case of branching keys, however, this has been modeled as a separate use case, since certain special requirements exist. Keys intended for printing would contain printable codes or numbers to link to couplets, resulting taxa, or subkeys, statements, rather than the hyperlinks intended for interactive use in local applications or internet browsers. The complex formatting required also influences how to deal with available space; for most branching keys page width may be more relevant than page length.

Fig. 209 shows the use cases of formatting branching keys to report them. The special printing use case includes the general creation of branching keys (which is also the basis for computer-

aided keys), which in turn encompasses various creation procedure (automatic, assisted, manual or markup; see Fig. 202, p. 312).

Morris & al. (2003) and (2007) present an application that formats taxonomic keys based on XML data and prepares output in any format supported by the Cocoon publishing framework (XML, html, PDF, etc.).

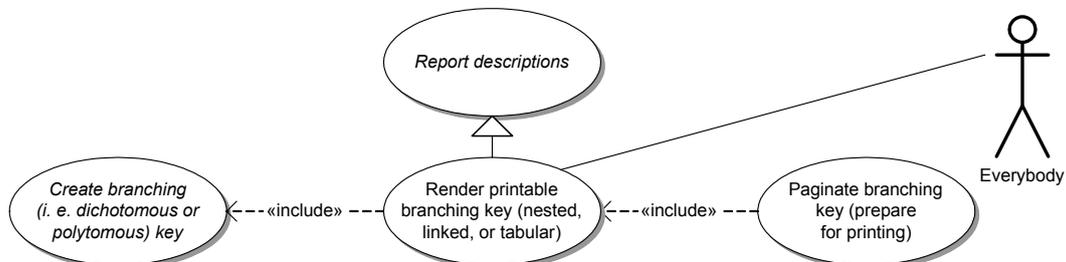


Figure 209. UML use case diagram showing the rendering (and optionally pagination to prepare for printing) of printable branching keys. See also Fig. 202 for the specialized use cases inheriting from the abstract “Create branching key”.

Class hierarchy reports

The class hierarchy, whether the result of analysis use cases (Fig. 191, p. 306) or obtained as data from another biodiversity framework component (compare p. 28), needs special provisions for report generation. Species descriptions are frequently reported using selected taxon ranks (e. g., order, family, subfamily, or genus) as formatted headings. In the case of phylogenetic analyses, more detailed graphical reports (with or without distance information, phylograms or cladograms, respectively) are desired. The tree-rendering use case was already included in Fig. 192, p. 307.

Creating index pages

Finally, reporting multiple terms from terminology or multiple natural language descriptions or keys may also require the generation of indices or tables of content (Fig. 210). The pointers of the index may be hyperlinks or page references for printed documents (dynamic page references are unfortunately not supported in html/xhtml!). Indexing may refer to multiple objects in a single document or to multiple documents, and it may be inside the object document or may form a separate document.

Indexes may be flat or contain structures, e. g., family, genus, species, infraspecific on different outline levels. In the case of indices to taxonomic names, a reverse index sorted by the epithet of lowest ranks (i. e., instead of “Genus species var. variety” the index entry would be “variety, Genus species var.”) is often very useful.

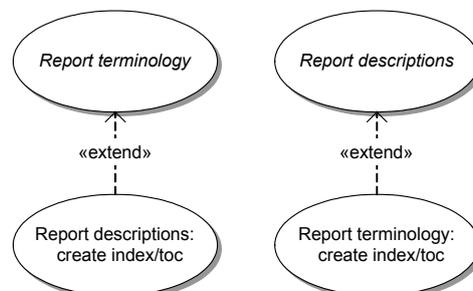


Figure 210. UML use case diagram showing that report generation for both terminology and descriptions may involve the creation of indices or tables of content (“toc”).

Taxon pages

An urgent need exists to document the properties of biodiversity on earth. One way to achieve this is to write a static html page as a “home page” for each taxon, combining all available information on:

- nomenclature,
- taxonomy,
- morphology and anatomy,
- chemical or molecular properties,
- geographical distribution,
- interactions with other organisms (e. g., pollinators, host-pathogen, predator-prey),
- interaction with the environment (e. g., growth parameters of plants in different soils, depletion or enrichment of nutrients),
- its known uses (e. g., in biotechnology, medicine including folk-medicine),
- economic importance (pest, biological control agent, pollinator, indicator, etc.),
- symbolic and mythological importance in culture and language, etc.

Taxon pages are often called “species pages” because species information is the most frequent and relevant form of taxon information. Such species pages have been advocated by H. Saarenmaa (Saarenmaa 1999 and 2002) and many others. A good example for species pages (called “fact sheets” there) is Schell & al. (not dated) (grasshoppers species). The “Tree of Life” (TOL, Tree of Life web project 2007) currently – and probably the upcoming “Encyclopedia of Life” (EOL.org 2007) in the future – provides truly *taxon pages*, i. e., for taxa at different ranks. Furthermore, it uses hyperlinks to navigate within these as a phylogenetic tree.

Manually generated static pages are like small essays on a taxon. They have serious disadvantages and many of the same problems that conventional printed taxonomic treatments have. It is difficult to keep them up-to-date, and it is unavoidable that individual pages contain only a selection of the total worldwide knowledge about a taxon. Furthermore, most information in these pages will be difficult to access other than by species name (the organizing principle of species pages); for example, host plant lists present in fungal species pages cannot be used as pathogen lists in the corresponding plant species pages. However, the fundamental problem of biodiversity informatics is the lack of information, not having too much information that is poorly organized. Therefore, the creation of static pages is without doubt a valuable resource. Also, some problems can be overcome, e. g., by improving collaboration through the use of a Wiki as a content management system. At the same time the foreseeable problems should caution against investing substantial resources into creating such poorly structured resources.

Another view on taxon pages is that they could be generated dynamically based on data that may be federated in multiple ways (Fig. 211). In Fig. 212 the use case “Report descriptions: natural language” is part of the descriptions component within a framework of biodiversity informatics components. The use cases “Report nomenclatural information” could point to a nomenclator, the reporting of the taxonomic hierarchy could be provided by the host application (which selects a particular taxonomic view for the arrangement of the species pages), etc. Within a framework component (p. 28), data could be federated as well. The natural language description could be based on information from many individual descriptions on multiple servers, and the use case “Report geographical distribution” would necessarily have to query a very large number of biodiversity collections (which may be indirect through the use of a portal like GBIF).

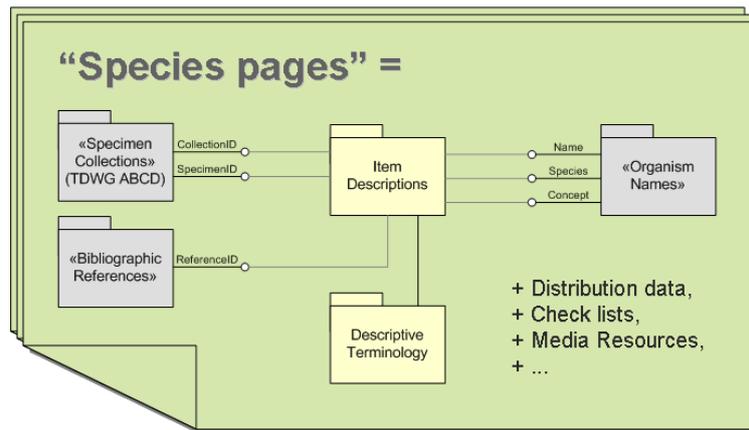


Figure 211. Relation between descriptive data and other biodiversity data areas shown as a package diagram; circles indicate component interfaces. “Species pages” are ideally a dynamic combination of descriptive data with data derived from other sources (Hagedorn 2002c).

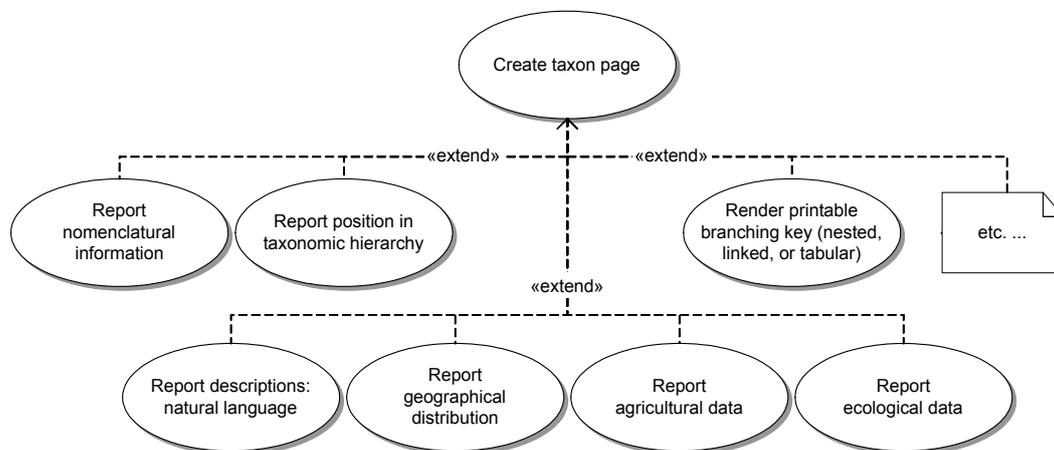


Figure 212. UML use case diagram showing how taxon pages are created by combining descriptive information with other information sources.

Digital monographic treatments like Flora or Fauna publications are closely related to the topic of species pages, but not analyzed in this work. This is a topic of active research (e. g., TaxonX: Catapano & al. 2006 or TaxMLit: Weitzman & Lyal 1999).

Data exchange and archival exports

A special case of report generation (and thus an extension to Fig. 206) is the export to data exchange formats. Such a format could be the XML-based SDD format, or the older DELTA or NEXUS formats. The NEXUS format currently has a special relevance since it is the preferred format for current phylogenetic (or cladistic) analysis applications.

A highly desirable feature of exchange formats is to inform about the completeness and quality of exported data relative to the original data. Data exchange formats differ strongly in their ability to express concepts used in an application, or to include even application-specific data (which is provided by SDD). Even if a format exports all descriptions, a loss of information may occur. For example, the format may support a coded description only by selecting a single audience and by combining character, state, and modifier information into a single unstructured text element. Also, assuming the original data are no longer available (e. g., when the export was used to archive information) it is relevant how many terminology or description objects out of the total

available are included in the export. A special case is that when exporting descriptions, the associated terminology may be exported only insofar as it is used in the descriptions, or in its entirety.

Fig. 213 displays the major component use cases for exporting descriptive data. The various use cases displayed have been defined separately because each has potential to be used singly rather than as part of a total export. For each export use case, also a corresponding import use case exists (Fig. 214).

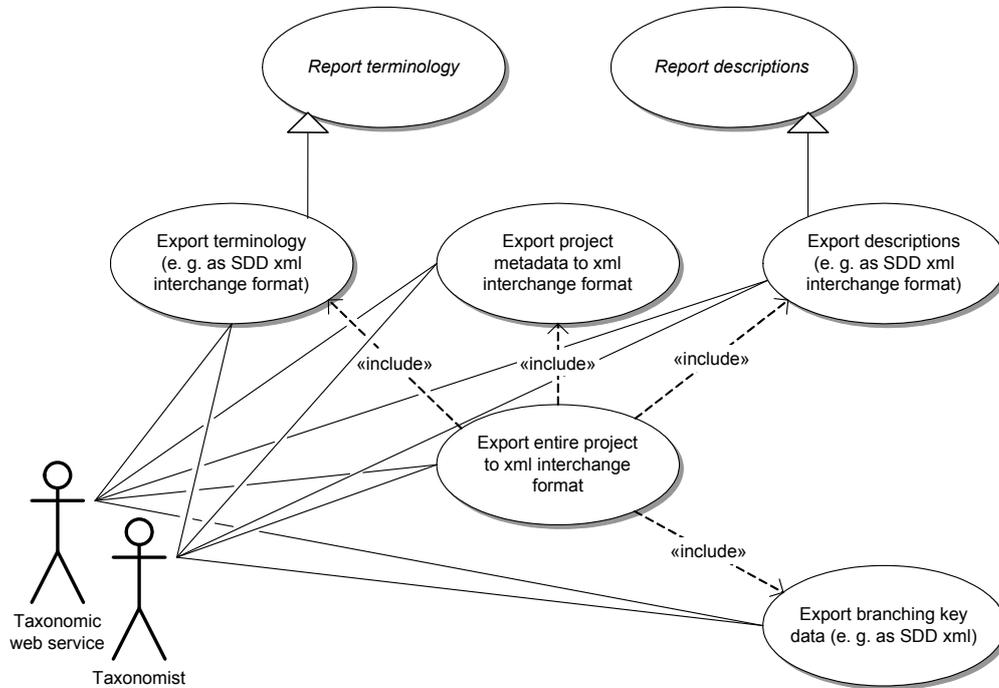


Figure 213. UML use case diagram showing export of descriptive data.

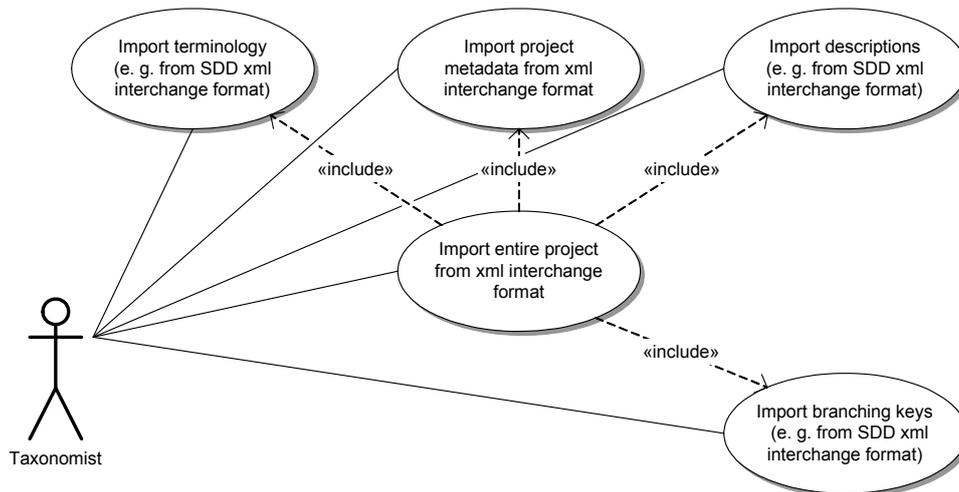


Figure 214. UML use case diagram showing partial or complete import of descriptive data, associated terminology, and project metadata.

The present analysis of data exchange is document-oriented. It fits identification packages, eLearning packages, taxonomic revisions, or monographs. However, not all information discovery and indexing use cases necessarily act on such documents, special discovery and searching protocols (e.g., to make data searchable through GBIF) providing data or metadata may be used.

It may be difficult to decide whether this needs to be modeled through low-level use cases, or whether this is more adequately modeled through software interfaces like those required for object exchange in a federated system.

6.8. Open aspects

Some aspects are already known to be incomplete. In the case of identification this is deliberate; the information presented in the use case analysis highlights the major use cases and is intended to be complemented by chapter 5, “Identification methods” (p. 229). No similar detailed analysis of phylogenetic analysis and of the creation of monographic treatments or revisions could be performed in this thesis. The use cases presented for these areas are fairly broad and require further studies.

Another area deliberately not analyzed is data security, access control, and perhaps encryption. These aspects potentially affect many use cases and would complicate the analysis. It is believed that the general patterns of making a system secure apply to descriptive data without any special cases. Prior to implementing a secure system, however, a detailed analysis of interactions would be required.

The author is highly interested if readers consider further use cases inadequately covered or missing.

7. Information model for Diversity-Descriptions 1.9

7.1. Introduction

DiversityDescriptions (originally named *DeltaAccess*, Hagedorn 1997, 2001a, 2005b) is the only information model documented in detail in this thesis (compare p. 16). As discussed in the introduction (p. 15), the model does not fulfill the requirements established in Ch. 4 to 6. Its presentation could have preceded those chapters, except that many concepts would have had to be introduced here.

It is believed that the model still has certain merits and is worth presenting. It is firmly based on the experience gathered with DELTA and has its own history of twelve years ongoing development and testing (see “history” further down). Furthermore, it allows a reassessment of the concepts of DELTA in the light of standard relational database techniques. Because it does not need advanced software support (like object-oriented or object-relational DBMS) it can be used as storage backend for a wide variety of software applications.

DELTA support and beyond: *DiversityDescriptions* is designed to support a large number of DELTA directives. To the author’s knowledge it is – next to the CSIRO DELTA programs themselves – the most complete implementation of the DELTA language. DELTA-coded text files are converted to relational database structures and may be exported back to DELTA.

In many ways, *DiversityDescriptions* goes beyond the CSIRO DELTA programs, trying to overcome some limitations of DELTA without giving up the compatibility with DELTA data files and programs supporting DELTA. Initially, *DiversityDescriptions* started to extend the DELTA language, but it soon became obvious that the DELTA format had some serious limitations (including the lack of an “end-of-block” marker). The DELTA extensions defined by *DiversityDescriptions* would prevent other DELTA-compatible programs from importing such data, seriously limiting the usefulness of this approach. By moving all extensions into a separate file (“Extras”) that is unknown to other applications, this could be circumvented, and a few such di-

rectives are still used. However, extending DELTA was not pursued as a priority, and priority given to developing the XML-based SDD standard (p. 20).

DeltaAccess can be a general data repository, in which the raw data can be entered during work in progress. It should be a working tool of the biologist, rather than an additional task after the completion of data collection. The data are edited or analyzed in the database, not in DELTA-coded text files.

History of DiversityDescriptions: DiversityDescriptions (here including the older name *DeltaAccess*) has been under development since 1995 with several beta versions distributed to colleagues and a public 0.99 beta release. Version 1.0 appeared 1997-07-31 (Hagedorn 1997), quickly followed by versions 1.1 (1997-10-24, adding subset management) and 1.2 (1997-11-16, adding multi-item edit, Fig. 174, p. 293) in the same year. These original versions already contained most of the DELTA support of the current version, and already added the new concepts of terminology-based modifiers and defined statistical measures.

With versions 1.3 (1998-03-26) the information model was considerably revised: the initial attempts to manage images and other resources had to be pruned back and the support for character headings and groups was added or greatly improved. In 1998, the program was presented on several congresses: TDWG (TDWG 1998), International Mycological Congress (Hagedorn 1998b), and OPTIMA (Organisation pour l'Etude Phyto-Taxonomique de la Région Méditerranéenne, Hagedorn 1998a) and a study using it for character analysis was published (Rambold & Hagedorn 1998).

The following versions (1.4 on 1998-11-11, 1.5 on 1998-12-01, and 1.51 on 1998-12-18) made only minor changes to the information model, primarily consolidating the basic functions of DeltaAccess, but also adding the first versions of HTML forms (Fig. 12, p. 41, left side). Version 1.6 (1999-07-31) finished adding and expanding the core functionality of DeltaAccess, including backup/restore, reorganization of projects, HTML form creation and HTML form re-import (Hagedorn & Rambold 2000).

Development continued at a slower pace in the following years: 1.7 (2000-06-30) introduced native generation of natural language descriptions, requiring some additions and changes to the information model, followed by maintenance/bug fix versions 1.71 (2001-11-28), 1.72 (2002-02-13, bug fixes), 1.8 (2002-12-20), 1.81 (2003-05-23), and 1.9 (2005-03-30, Hagedorn 2005b) introduced secondary, unchanging object identifiers (in addition to the sequence-dependent DELTA IDs), and updated the information model by providing multilingual support through translation tables, completely redesigning the resource table (images, etc.), and extending the character headings structure to become a fully hierarchical tree. As of 2007-06 a version 2.0 is in beta testing. The major change is adding a DescrScope class to improve linking descriptions with taxon name or concept databases, as well as specimen, publication, or geography databases using GUIDs like URIs (compare "Secondary classification resulting in description scopes", p. 215). Furthermore, the DELTA import/export was revised to bring it better in line with existing database features (improved support for inline formatting, long character state definitions, and extending the DELTA format for Unicode).

Presentation of the information model: The information model of DiversityDescriptions is presented both as an abstract *logical model*, highlighting the strategies for creating a relational model for descriptive data, and as a *physical model* (p. 332 ff), documenting in detail the tables, relations, and fields as implemented in DiversityDescriptions, version 1.9. *Logical* and *physical* model are used in the sense used in ER-modeling. In UML terminology, the present *logical model* may be understood to be approximately intermediate between UML *conceptual* and *specification model*, physical model is roughly equivalent to a UML *implementation model*.

7.2. Logical model for DiversityDescriptions 1.9

The logical model introduces the conceptual ideas and strategies that guided the design of DiversityDescriptions 1.9. The model uses supertype and subtype entities (i. e., generalized and specialized classes in UML), but otherwise the design principles of relational databases are observed. To simplify and focus the presentation, secondary entities are omitted and the list of attributes is often shortened (indicated by “[...]” in the attribute list).

The UML static class diagrams are shown with attributes but without operations. To simplify the comparison with the physical (implemented) model (see p. 335), key and foreign attributes are shown in addition to the relationship arrows. Also, n:m relationships have been modeled using association classes even where no attributes in addition to the foreign key attributes exist. Navigability for associations and association classes is not shown in the UML class diagrams. It is assumed to be bidirectional (following the relational model rather than directional pointers in hierarchical or network models). See “UML static class diagrams and ER models” (p. 24) for further information on the UML methodology and notations used.

Note that some attribute names differ between the logical and the physical model where this was considered advantageous for conceptual understanding. The complete list of attributes in the physical model may be found in the “Data dictionary”, p. 339 ff.

Packages and subsystems

The fundamental model used for descriptions is that all terms and concepts must first be defined to create a terminology that is subsequently applied to descriptions of objects or classes (taxa). Terminology, Descriptions, and Resources (mediating access to images, etc.) are the basic packages of the Descriptive Data subsystem (Fig. 215). The Descriptions package strongly depends on Terminology (it cannot exist without it). Both Terminology and Descriptions depend weakly on Resources (but some data sets may be completely without resource usage).

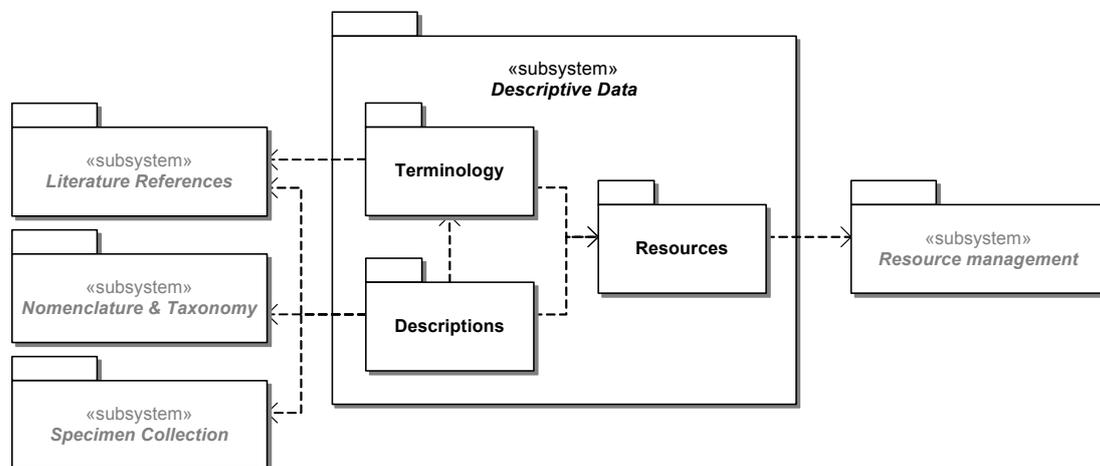


Figure 215. UML subsystem and package diagram for the descriptions model in DiversityDescriptions 1.9. The dashed arrows are to be read as “depends on”. The central subsystem, consisting of three packages is discussed here; “Resource management”, “Literature References”, “Nomenclature & Taxonomy”, and “Specimen Collection” subsystems are derived from a biodiversity information framework (e. g., DiversityWorkbench).

The DiversityDescriptions model is designed to collaborate with external components or subsystems in a biodiversity information framework (e. g., DiversityWorkbench, see Fig. 6, p. 29). However, the information model is designed in a way that permits to operate DiversityDescriptions in a “stand-alone” mode, using text labels to describe external objects rather than using

technical, ID-based linking mechanisms. Although much effort has been spent in recent years on developing matching applications as separate, interacting software component (e. g., Diversity-Collection, Hagedorn & Weiss 2002, DiversityResources, Hagedorn & Kohlbecker 2006), so far DiversityDescriptions is still normally used without online interactions with other components. This is largely due to the difficulty in integrating software components produced with different tools (.NET/Mono, Java, COM-Object model).

Package: Terminology

The main concepts defined in the Terminology package (Fig. 216, left side) are:

- Character headings: hierarchical classifications of character variables. These are roughly equivalent to, but less general than, “Concept hierarchies” (p. 125).
- Character variables, i. e., “character definitions” or “measurement concepts” (compare “The term ‘character’”, p. 31).
- Character states, i. e., value concepts for categorical data, coding status, and statistical measures (see below).
- Modifiers (meta-information on values; see “Modifiers”, p. 189).

Character states are strictly nested within characters (represented by an aggregation relation). The n:m relation between characters and modifiers or headings is defined using association entities (CHAR_Heading_Link and MOD_Link).

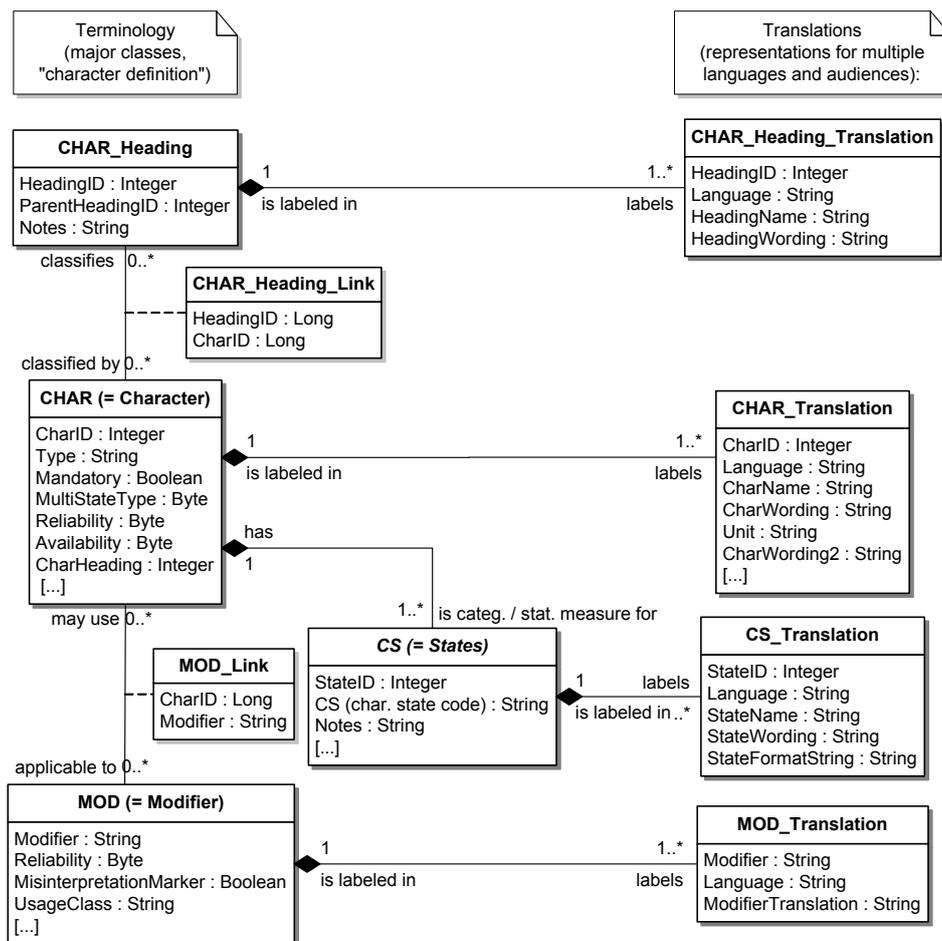


Figure 216. UML class diagram for major classes of the Terminology package (logical model DiversityDescriptions 1.9).

Translations

For each of these primary terminology concepts (characters, states, modifiers, and headings) the language-dependent attributes such as names (i. e. labels), wordings (for natural language generation), and value formatting patterns (“StateFormatString”) may be supplied in multiple languages (Fig. 216, right). The advantages of automatically translating many coded description by translating the centralized descriptive terminology has been discussed in “Multiple languages or audiences” (p. 282). Significant differences exist between the conceptual and the physical model with regard to translations, compare p. 338.

States and other terms used in descriptions

Character variables define the places where information can be stored in the descriptions (character data, see “Package: Descriptions”, p. 329). Characters have different character types (compare “Implemented data type systems”, p. 61, and Table 8, p. 61) and for each data type different categories of information are supported in the descriptions. These are:

- character states (i. e. categorical values),
- measurement values or statistical measures (compare “Quantitative data and statistical measures”, p. 110 for the concept and “Statistical measures in DiversityDescriptions”, p. 356, for a list of supported values),
- coding status values (p. 74), and
- free-form text (compare “Unconstrained text”, p. 56).

All character data in the description refer to instances of the class “CS” (originally the abbreviation of Character State). Conceptually this class is abstract and has separate non-abstract subclasses for each of the information categories mentioned above (Fig. 217). In the physical model, these subclasses are all merged into the single entity type CS and distinguished by different value ranges used for the CS.CS attribute.

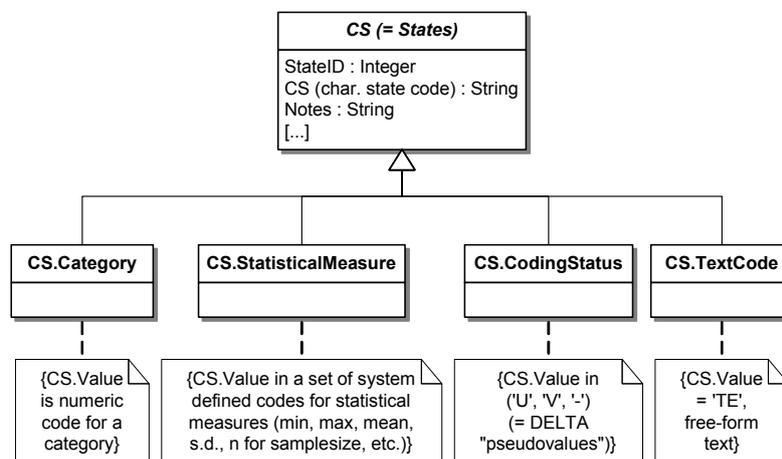


Figure 217. UML class diagram showing the conceptual subclasses for different kinds of character data. (This is purely conceptual, not part of any package in the logical model.)

To make a character value like a character state or coding status value usable in a description, it must be defined in the terminology. Some categorical values (e. g., shape or color values), but especially the coding status values and statistical measures, are typically used by many characters. However, to simplify the model, the relation has been modeled as a composition, i. e., each state belongs exactly to one character. This requires a certain amount of duplication of the labeling information, but has the advantage that the association of coding status or statistical measure

with a character (i. e., which coding status or measure is enabled for a given character) is defined in a simple, consistent way.

Semantics of categorical values are freely definable by the designer of the terminology. Several methods exist to support communication with humans, but no support for machine-reasoning other than what is defined through the character type (e. g., nominal versus ordinal) exists. In contrast, the remaining subclasses (statistical measures, coding status, and free-form text) each recognize a set of fixed values, for which system-defined semantics exist.

Character headings

Characters are organized into concept hierarchies (for parts, properties, field/laboratory characters, etc.) created through the class `CHAR_Heading` (Fig. 218). This class has a reflexive relationship on itself; the attribute `ParentHeadingID` is used to create a forest of trees (i. e., multiple unconnected heading hierarchies may be present). Cyclical relations are prevented through a separate constraint (“{acyclic}”). Each character may be a member of multiple character heading concepts.

The most general association between characters and the heading hierarchy occurs through the `CHAR_Heading_Link` association class. In addition two additional relations exist that are constrained to a 1 : n multiplicity (based on relations between “`CHAR_Heading.HeadingID`” and “`Char.CharHeading`” and “`CharWithNatLangDefinition.HeadingLink`”, respectively; labeled “legacy heading def.” and “legacy subheading def.” in Fig. 218). The topic is discussed in more detail on p. 336 in the physical model.

Descriptive concepts in DiversityDescriptions are not directly referred to by descriptions. They can therefore be improved and changed at any time, without affecting descriptions stored elsewhere.

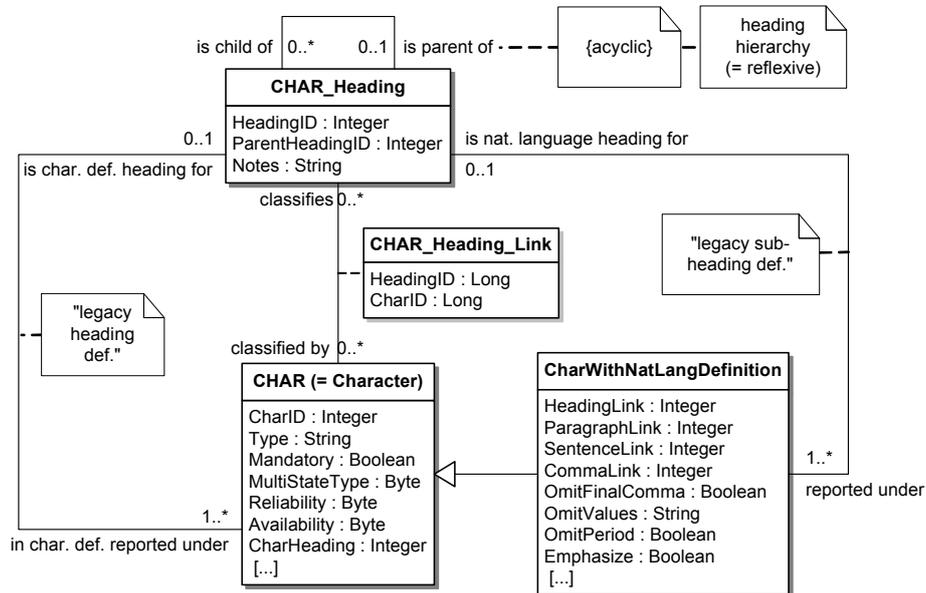


Figure 218. UML class diagram showing details of three different heading variants and attributes for the generation of natural language descriptions (logical model DiversityDescriptions 1.9).

Natural language generation

A special group of character attributes is used only for the conversion of coded descriptions into natural language descriptions (see p. 39). In Fig. 218 these are modeled as a subclass of the char-

acter definitions. Natural language generation is a complex process not discussed elsewhere in this thesis; it therefore needs a slightly more detailed discussion in the following.

Natural language descriptions typically arrange characters into multiple hierarchical levels: subheadings (with a heading text), paragraphs, sentences, semicolon- and comma-separated characters, and finally states connected by ‘or’, ‘and’, ‘to’, ‘with’, etc. In DiversityDescriptions the subheading level is represented using the special subheading relation mentioned above. The latter levels are created using an independent mechanism called “link groups”. These are attributes storing integer numbers, for which special semantics are defined such that consecutive identical numbers define a group. For example, if two consecutive characters have the same number in the attribute SentenceLink, they will be placed in the same sentence. A new sentence starts if the link group value changes (Fig. 219). The exact value of the numbers is irrelevant, and if always all characters were present, two different values would suffice (which is the method used in the simplified Figs. 219- 221). However, since any intermediate character may be missing in a given description, using a new value for each link group is the most reliable method.

Char. ID	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15
Sentence	█	█	█	█	█	█	█	█	█	█	█	█	█	█	█
Comma		█	█						█	█					
Output:	;	,	.	.	;	;	,	,	.	,	;	;	;	,	.

Figure 219. Diagram illustrating the results of value-based link group attributes (SentenceLink and CommaLink) on characters (top) in a natural language description. Alternating colors express different values, empty cells no value.

Link groups have a natural hierarchy: If two characters have the same sentence-link value, but different paragraph link value, the nature of the formatting implies that the characters will be placed in different paragraphs *and* different sentences (Fig. 220).

	Char. ID	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15
A)	Heading	█	█	█	█	█	█	█	█	█	█	█	█	█	█	█
	Paragraph	█	█	█	█	█	█	█	█	█	█	█	█	█	█	█
B)	Heading	█	█	█	█	█	█	█	█	█	█	█	█	█	█	█
	Paragraph	█	█	█	█	█	█	█	█	█	█	█	█	█	█	█
C)	Heading	█	█	█	█	█	█	█	█	█	█	█	█	█	█	█
	Paragraph	█	█	█	█	█	█	█	█	█	█	█	█	█	█	█
D)	Heading	█	█	█	█	█	█	█	█	█	█	█	█	█	█	█
	Paragraph	█	█	█	█	█	█	█	█	█	█	█	█	█	█	█

Figure 220. Diagram illustrating the hierarchical nature of link groups. Links groups on a higher level (here heading) effectively break lower level link groups (here paragraph). All four examples result in four sentences, albeit with a different number of headings.

It may be noted that a – perhaps more conventional – model of “new paragraph/sentence/etc. here” would not work. In an actual description any character may be missing and the mechanism must be totally independent of the presence of other characters.

The present model is not optimal. A better mechanism would be either to use actual character headings (used in SDD, which supports unlabeled nodes and delimiters before, after, and between members of a node in a character tree, see Fig. 221), or a modified link group mechanism with just a single link number attribute, and a second attribute expressing the kind of link group (paragraph, sentence, etc.).

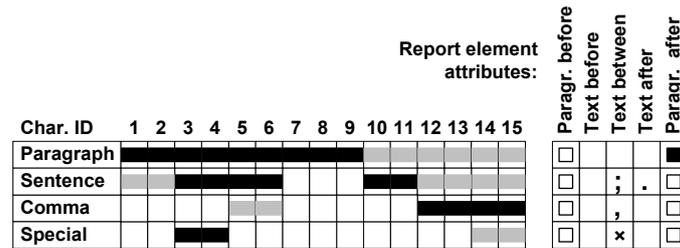


Figure 221. Diagram illustrating how link groups may be translated into “report elements” defining delimiters, compare Fig. 219 for diagram conventions. The Special link shown at the bottom is extensible, used here to define a multiplication sign (e. g., between length and width; see physical model for further details.)

Character applicability

Following DELTA, character dependencies are limited to character applicability rules where the controlling character is of a categorical type (compare “Character applicability rules”, p. 76). Counts of object parts are not supported as controlling characters, although this is in principle desirable (see p. 77). This limitation closely follows the DELTA model.

Applicability rules are expressed in an association table (“DEP”) between a controlling state and the controlled character (Fig. 222). The rules are defined as part of descriptive terminology, but evaluated by applying them within individual descriptions. If in a description the set of controlling states for a controlling character results in others characters becoming inapplicable, only the characters in the same description are affected.

The model supports only *inapplicable-if* rules. It is similar to the BAOBAB model (White & al. 1993), except that DiversityDescriptions defines inapplicable dependent characters, while the BAOBAB model defines characters explicitly as applicable. The potential problems that arise from the incomplete reversibility of the two forms of dependency rules have been discussed on p. 79.

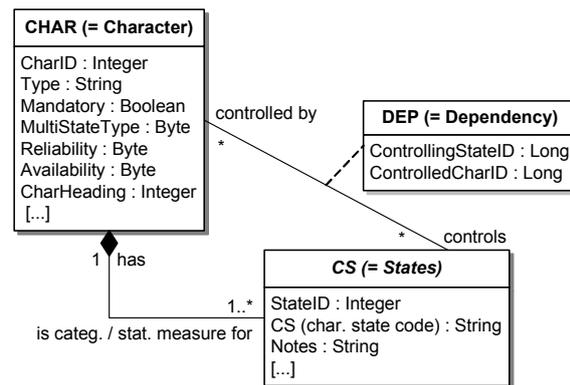


Figure 222. UML class diagram illustrating character applicability (logical model DiversityDescriptions 1.9).

Package: Descriptions

The Descriptions package (Fig. 223) is structured into two classes: a definition of the item (i. e., the specimen, taxon, disease, behavior, etc.) that is being described (class “ITEM”) and the individual description records (class “DESCR”). The model follows the “list model” described in “Categorical data: Character matrix vs. character state matrix” (p. 104).

The item definition primarily defines an immutable ID for reference and an informal label (“ItemName”) which may contain the taxon or disease name, specimen accession number, geographic or other secondary classification the scope definition. (“ITEM” as shown in Fig. 223 also contains several specific scope attributes (specimen from collection, literature publication citation). This will change in the upcoming version 2.0 of DiversityDescriptions, where a separate and more general DescrScope class along the lines discussed in “Secondary classification resulting in description scopes” (p. 215) will be added. In the interest of a consistent documentation this is not further shown or discussed in this thesis.)

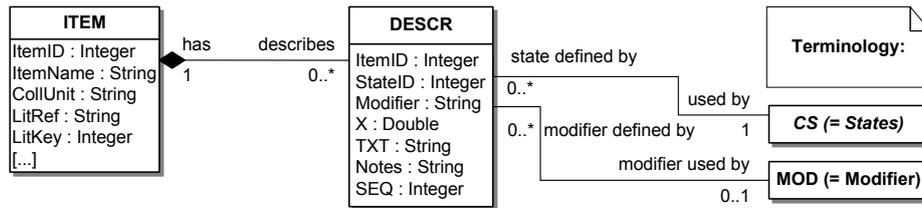


Figure 223. UML class diagram showing the Descriptions package (left) and its relations to terminology (logical model DiversityDescriptions 1.9).

Description records (class DESCR in Fig. 223) belong to items by composition. The description records aggregated under an item form “the description” of an item. The “list model” storage solution scales very well from small to very large data sets. The number of items, characters, and states is unlimited and no undue performance penalties arise if any of these is unusually large.

Each DESCR record must refer to exactly one instance of the state terms (CS) defined in the terminology, covering categorical states, coding status values, statistical measures or a free-form text status value (‘TE’). The character is not stored explicitly in DESCR, but can be inferred by following the link from CS to CHAR.

Conceptually, for each of the superclass/subclasses for different character types and coding status values discussed in Fig. 217 for Terminology, corresponding classes exist in the Descriptions package. Thus, the simplified relation between DESCR and CS shown in Fig. 223 (which is close to the physical model) can conceptually also be presented as shown in Fig. 224 below.

This diagram illustrates that the applicability of some attributes in DESCR depends on the subclass of CS referenced. Whereas the Notes and SEQ attributes are applicable to all subtypes, modifier values may only be added if the record refers to a categorical state or a statistical measure. If a modifier value is added, it must come from the set of modifiers defined in the class “MOD”. The attribute X may only be used when statistical measures are referred to, and TXT when free-form text is selected.

Again, both in the logical (Fig. 223) and the physical model, this concept is simplified by merging all subclasses of CD (CharacterData) into a single class “DESCR” that relates to a single class CS. The value space of CS.CS is partitioned such that the different subclasses of CD can be identified.

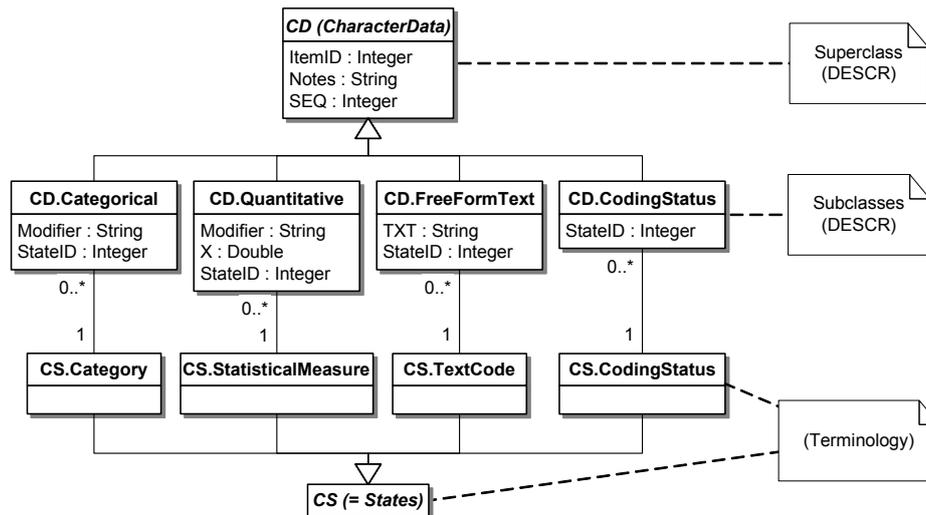


Figure 224. UML class diagram showing the conceptual superclass/subclass design of the DESCR (Descriptions package) and CS (Terminology package). Compare Fig. 217 (p. 326) for the Terminology part. (This is purely conceptual, not part of any package in the logical model.)

The attribute `DESCR.SEQ` fulfills the special requirement that the sequence of categorical states in character data may or may not be semantically significant. In many cases the author of a description will expect the states to be ordered in the sequence defined in the terminology. Occasionally, however, a special meaning is given to a sequence of states that deviates from the one defined in the terminology (e. g., “red, orange, yellow” versus “flower orange, rarely red”, see “Value order in character data” on p. 113). In support of this, the `SEQ` attribute is by default set to ‘0’, but may be used to manually order individual state sequences. When generating character data reports, a join of `DESCR`, `CS`, and `CHAR` will be ordered by character order (`CID`), manual order (`DESCR.SEQ`) and default state order (`CS.CS`).

Package: Resources

“Resources” as used here is an abstraction for still images, video or audio files, documents with complex formatted text and embedded media (such as taxon pages, PDF, or XPS files), and executable tools. The latter is especially relevant for identification keys (which, however, may also be simply formatted text). As used here, resources should be available in digital format and presentable by a computer process to a human user. Resources may be referred to by URLs (universal resource locators) or web service parameter sets, but also by URNs (universal resource names), provided the Descriptive Data subsystem can detect a resolution service for them (i. e., a method how, knowing a resource identifier, to obtain the resource such as a digital image itself; an example of a URN-based resolution service is the Life Science Identifier (LSID) system planned by GBIF).

Ideally, a resource record would refer to an abstract entity (e. g., a literature reference, a specimen, or an image from an image bank available in multiple resolutions). This abstract resource could then be subject to content negotiations, so that a client could obtain it in a format, quality, and compression level that is appropriate for the display device used. Similar negotiations may allow returning the most appropriate caption for a given language and expertise level of the user. The differentiation between abstract resources and concrete instances is potentially appropriate for all resource types mentioned above. In the case of computer-aided identification keys, different versions may exist for different target machines (e. g., for web-browser-based JavaScript, Java, or .NET/Mono).

Content negotiation may simply be based on http mechanisms, or may be implemented through custom web services. To stress this design, the Resources package in Fig. 225 (right) is drawn as referring to a separate resource management subsystem, which could handle these services as well as other management services for resources (versioning, persistence of URLs, IPR and access right management, discovery services, etc.). In practice, however, the `ResourceLink.ResourceString` will often be a simple URL, which may be a simple path or parameter-based, and not offer any persistence, content negotiation, etc.

Despite the fact that “resource” is a strongly generalized concept and is outside the Descriptive Data subsystem, several specialized and context-dependent metadata are necessary. These are handled by the Resource package inside Descriptive Data (Fig. 225). The full conceptual model is that each character, state, or item may be associated with a list of resources (Fig. 225). For each association a preferred display order, a *role*, and a *usage class* may be defined. The *role* informs whether the resource is intended to be used as an icon (i. e., supporting a text label in lists or trees of characters, states, or items), as a selector (information-rich so that it can fully represent the associated object, with or without text), definitional (informing about the object), or associational (other relation considered of interest or use to the user). The *usage class* supports the categories “Always”, “Desirable”, or “Secondary” (compare also detailed value descriptions in the physical model; see `RSC.Role` and `RSC.ItemUsage`, p. 351). Based on these metadata, a client application is enabled to decide whether a resource, if possible for the program, is to be embed-

ded or linked to. Links may be directly available, or only after an intermediate step (a list of links is displayed after the user selects an option like “further information...”).

The association classes point to a further association class, which is a kind of local proxy for the external resource and provides the option to add captions and notes in multiple languages (and perhaps even default Roles and Usages, not shown). Note that the same resource may have multiple, context-dependent captions. For example, the butterfly images in Fig. 118 (p. 239) labeled “Tiger” or “Clearwing” would be differently labeled if they were used in the context of a species page. In the logical model presented here, this requires creating multiple instances of ResourceLink containing the same ResourceLink.Resource URI or web service access definition.

A problem not discussed here is that, although following a link to a resolvable resource in the web by opening a new browser window is highly general and can be expected to work for most resources, embedding a resource (images, audio files, Java applets) inside a page is much less general and requires additional metadata on the resource. Currently it is assumed that this information can be obtained by resolving the resource string to obtain the MIME-type information. In a physical model it will probably be beneficial to cache this information.

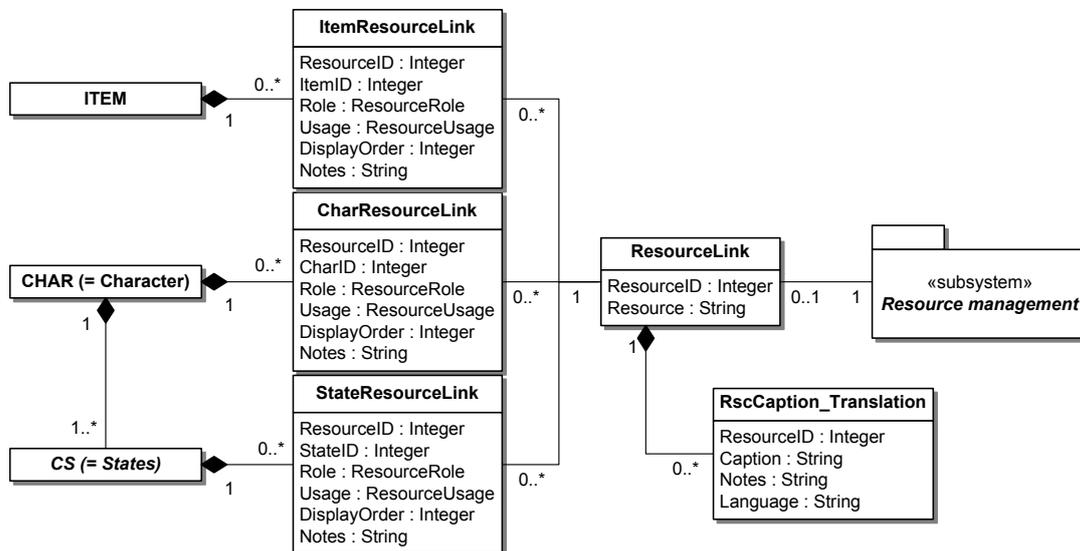


Figure 225. UML class diagram illustrating the Resource entity and its relations to entities from other packages (logical model DiversityDescriptions 1.9).

Note that the complexity of this model has been strongly reduced in the physical model (compare Fig. 230, p. 338).

7.3. Physical model for DiversityDescriptions 1.9

The *physical model* differs from the logical model in a few selected simplifications plus several cases where refactoring limitations have prevented updating the physical model with the improved model used in later versions of DiversityDescriptions. A major reason for such legacy problems is the original design goal to stay as close as possible to DELTA, supporting lossless import and export operations with the DELTA format. These differences are discussed in the first section, “Implementation constraints”.

In the next section actual relationships are documented (p. 335 ff), followed by a “Data dictionary” (p. 339 ff) containing a complete list of attributes in the physical model, including information about implemented data types, optionality, field and table constraints, associated pick lists, and the various index types defined. The extensible property-value model for project metadata

and the support for statistical measures (p. 356) are documented. In addition to documentation through a data dictionary, the SQL-source code is given in the appendix (p. 400 ff).

Earlier versions of DiversityDescriptions (up to version 1.6) were documented in the DeltaAccess/DiversityDescriptions user guide (Hagedorn 1999a).

Implementation constraints

The database engine (Microsoft JET = Joint Engine Technology) used for the implementation of the various versions of DeltaAccess/DiversityDescriptions provides a fairly complete implementation of a relational DBMS, including declarative referential integrity and cascading updates and deletes. Constraints can be placed on tables and individual attributes. The major limitations of the JET DBMS are the lack of triggers (other than those implicit in declarative referential integrity) and the lack of constraints that span several tables. See also Connolly & Begg (2002) for a general comparison of the architecture of JET with other DBMS.

The JET DBMS is optimized for small networks of about 40 concurrent users (maximal 255). Although this is a serious limitation for many types of applications, DeltaAccess/DiversityDescriptions was designed from the start as a project-oriented data management system. Each project was intended to be used only by a limited number of collaborators. The DELTA language supports no IPR, attribution, or change history, and the original implementations were all restricted to single-user operations. DiversityDescriptions inherited this DELTA legacy, and while designed for small concurrent editing, lacks advanced features for large-scale collaboration support.

In general, the physical model remains largely similar to the logical model discussed above. No denormalizations were introduced, but the following design decisions and deviations from the logical model may be noted:

- Superclass/subclass relations are not directly supported in the JET DBMS. As a consequence, implementations must choose whether to model them using 1 : 1 relations or by combining all attributes in a single entity, and leaving it to database constraints and applications code to sort out the subclass situations. Testing showed that models with a high number of 1 : 1 relations (and the ensuing multiple outer joins) perform extremely poorly in large data sets. The chosen solution is to merge the attributes of the superclass and all subclasses into a single table. The consequence of this is that subclass attributes will often be Null where not used. Testing showed that this has no detectable performance penalty; this can probably be generalized for any DBMS providing variable length data storage (i. e., the storage space required for empty fields is minimal).
- The information model is designed only for a single project, but the application supports an unlimited number of projects. To achieve this, the full model is dynamically created for each project, using a template model. From each project name a token is generated and used to prefix the table names. In the following model, all tables are prefixed with DD (“DiversityDescriptions”). If a database contains two projects, one called “BUS”, the other “CUS”, the tables BUS_CHAR and CUS_CHAR would be present (but no “DD_CHAR” table!). The use of name prefixes is necessary because objects in JET have no namespace or schema component. For example, in SQL Server the design might have been implemented as bus.CHAR and cus.CHAR. The advantage of separating projects into separate tables is:
 - Simplicity: In the DELTA model projects are totally independent entities. Managing multiple projects in a database seemed desirable, but otherwise little interaction between projects was planned for.
 - Performance of the database. The alternative design of a single set of tables for all projects would require almost all queries to be restricted to a project. Such queries would be more difficult to optimize for the internal query optimizer, more constraints and indices being involved. Furthermore, the likelihood that data pages are placed physically close (and included in consecutive hard-disk reads) is decreased. Finally, the size of indices becomes proportional to the sum of records in all projects, not just a single one, and performance

critically depends on index size. This was particularly obvious since the first versions of DeltaAccess/DiversityDescriptions were designed for computers with 16 MB total RAM, and performance testing showed the importance of index size in large projects.

- A simple design of character and item subset restrictions. DiversityDescriptions was planned from the start to provide for subsets functioning as user-definable “windows” on the underlying project. The JET database treats views (which may include joins) and tables identical. This allows creating an unlimited number of subset projects consisting of SQL-views which contain the necessary subset restrictions, but otherwise behave identically to base project tables (Fig. 226). For a subset project “Family1”, all based tables would be mirrored in views named as “Family1_CHAR”, etc. To all data-processing programming code, these view-based projects act identically to table-based projects. Compare also Fig. 152 (p. 261) for a different subset illustration.
- The first versions of DeltaAccess/DiversityDescriptions followed the example of the DELTA format and combined the functions of providing a unique ID and defining a display order sequence in a single attribute. For example, the attribute “CID” defines both a character ID and the sequence in which the characters are to be displayed to the user. The character display sequence is frequently changed, but due to the cascading updates provided by referential integrity, this proved to be both simple and sufficient for a considerable time. However, over time an increasing need for external references to character or items appeared and the initial design proved to be problematic. Due to the central position of the key attributes for existing programming code, it was impractical to fully refactor the design. Instead, in addition to the original keys CID, HID, IID, and CID/CS, new system-generated candidate keys: CharID, HeadingID, ItemID, StateID were introduced (not for resources, where a complete redesign was still possible). Some newer relations (esp. with translation tables) already use the new keys, whereas most relations are still based on the older keys. Note that the logical model in the previous section only mentioned the new keys and thus differs from the physical model.

It has to be emphasized that the information model of DiversityDescriptions 1.9 does not fulfill all the requirements elaborated in the previous sections. It is presented here because it represents a substantial advancement over previous models and because it has been fully implemented and tested in a number of software applications. The experience with these applications is the basis for the evolutionary redesign of the next information model and for the currently developed SDD XML data exchange format. This is also the reason why no fully conceptual model is presented: Either the conceptual model would have to be reduced in a way that it no longer reflects the improved understanding of descriptive data, or it would be too difficult to bridge the gap between the conceptual and the logical model.

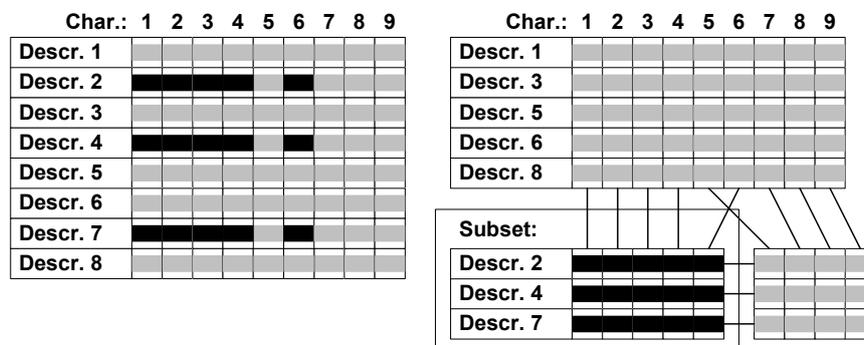


Figure 226. Illustration of a full project (left) from which only those characters and descriptions colored black shall also be accessible in a subset project (right; in DiversityDescriptions subsets are based on views, allowing both analysis and editing).

Entity relationship diagrams

The ER diagrams for the physical model are presented as Microsoft-Access diagrams, showing table and field names and relations. Field names in bold belong to the primary key of a table (i. e., the combination of values in these fields must be unique). The multiplicity is expressed through ‘1’ (primary key, representing the UML equivalents ‘1’ or ‘0..1’), and the infinity sign (‘∞’, foreign key, representing the UML equivalents ‘0..*’, ‘1..*’, or ‘*’). Consequently, optionality of relations is not shown (in the case of primary key attributes, printed in boldface, it can be inferred). An advantage of these diagrams over most ER or UML diagrams is that the relation lines are always drawn next to the field to which they refer, indicating the primary and foreign key fields involved in a relation.

All relations are protected by relational integrity rules with cascaded updates. In cascaded updates, if the attribute on the 1-side of a relation (primary key) is changed, the DBMS automatically updates the attribute on the n-side (foreign key) as well. Furthermore, several relationships are defined with cascading updates. If CHAR records are deleted, all related CS, DEP, RSC, MOD_Link, and CHAR_Heading_Link records are deleted as well. Similarly, if CS or ITEM records are deleted, DESCR and RSC records are automatically deleted; CS further cascades to DEP. All ... Translation records are automatically deleted if the corresponding entity is deleted.

The first ER diagram (Fig. 227) presents an overview of the principal entities. The field list of some tables is abbreviated, indicated by “[...]”. By convention, all fields of numeric identifiers end in “ID”.

On the left side the tables related to descriptive terminology are listed. The primary concepts used in descriptions are Character definitions (CHAR), one or several character state definitions (CS) for each character, and modifier definitions (MOD). These provide the semantics of the variables, enumerated values, and modifications of values such as frequency or certainty with which a value is correct.

Dependencies between characters are expressed through the table DEP. The physical model of character dependency fully follows the logical model discussed above (p. 329), except that the ControlledCharID is named “InapplicableCID”, and ControllingStateID is represented by the combination of the CID and CS fields, the primary keys of the CS table.

Further, in MOD_Link the relation between modifiers and characters can be controlled such that only specific modifiers apply to a character. Note that in contrast to states, which are always local to a given character, a modifier may be defined for the entire terminology.

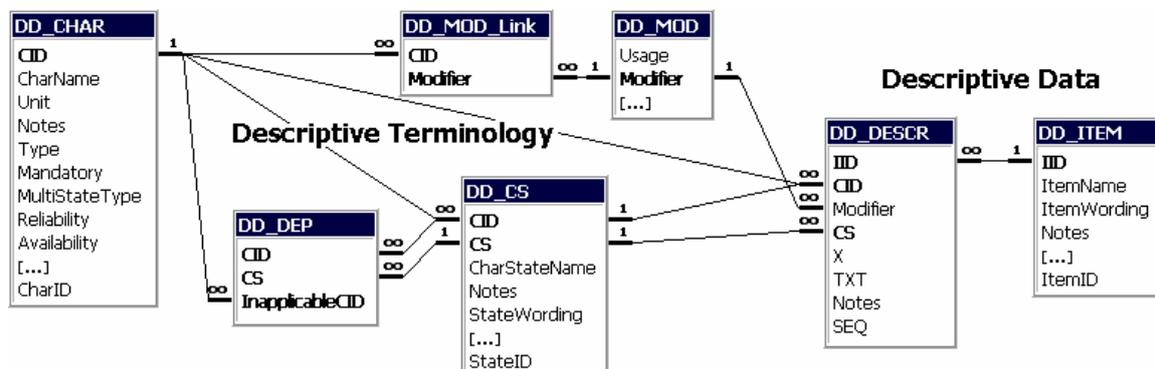


Figure 227. Reduced entity relationship diagram of DiversityDescriptions 1.9 (physical model), containing only the principal tables.

On the right side of the ER diagram (Fig. 227), the two tables containing the descriptive data are shown. ITEM contains the definition of the items being described and DESCR the pieces of the description.

An ITEM may be an object/specimen or a class/taxon). It is largely defined through an unconstrained text label “ItemName” which in the case of a specimen may contain taxon name plus accession location or number.

In the DESCR table one record is created for each character state or statistical measure used in a character \times item combination. The information model is essentially based on the “list model” (compare Table 27, p. 106) as discussed in “Categorical data: Character matrix vs. character state matrix”, p. 104. Most fields of DESCR are applicable to all types of descriptive data, but TXT (text of unlimited length) is applicable only to free-form text and the real numeric field X to quantitative values or measures. In the first case CS must contain the special state ‘TE’ indicating a text value, in the latter case CS must contain a code for each statistical measure (mean, min, max, s. d., etc.). Which measures are applicable to which character is defined by the author of the terminology, by adding a definition for a measure in the table CS; see the section “Statistical measures in DiversityDescriptions” (p. 356) for further information.

Coding status values (also called “pseudo-values” or “special symbols” in DELTA; compare p. 74) are also handled using the CS field in DESCR. Only the DELTA-defined values ‘U’, ‘V’, and ‘-’ are supported natively in DiversityDescriptions, but any number of special codes may be defined in principle. DiversityDescriptions thus follows a fixed vocabulary approach both for coding status and statistical measures (compare p. 287 in use cases). When using Diversity-Descriptions, all three values are added for newly created characters. The author of the terminology may, however, decide which coding status values are available by removing undesired codes from the CS table.

By default, the order in which categorical data (records in DESCR) are displayed is the order in which states and measures are defined in CS. This may, however, be overruled by the data editor using the SEQ field in DESCR. This may optionally contain a number defining a sequence (order) of character data in the item description.

If the relation between a modifier and a character is removed, existing character data in DESCR using modifier are not removed. They indeed remain valid because the modifier semantics (in MOD) are still available. Thus the MOD_Link may be interpreted as a usage recommendation, rather than as a strictly enforced constraint.

As mentioned above, many tables have two ID fields (e. g., “CID” and “CharID”). The short form CID, HID, IID, and CID/CS is for legacy reasons still the primary key. These follow the example of DELTA and express both identifier and ordering semantics. Because of the latter they are liable to change over time. Within DiversityDescriptions this is unproblematic due to the actions of referential integrity with cascading updates. However, to simplify the interaction with external programs, additional ID fields have been added over time (CharID, HeadingID, ItemID, and StateID) which are now candidate keys, i. e., they could be used as primary keys.

The complete ER diagram is shown below (Fig. 229). The main concepts not yet discussed are:

1. Character Headings: These allow defining hierarchical groupings of characters. In DiversityDescriptions, headings are defined in the table Char_Heading, primarily providing a display label “HeadingName” and a wording for natural language reporting (“Heading-Wording”). The AutoGroup field is an internal mechanism to provide for headings based on search rules rather than on relations with the character table. ParentHeadingID creates a reflexive relation from Char_Heading to itself, providing the option to build nested hierarchies (trees) of headings.

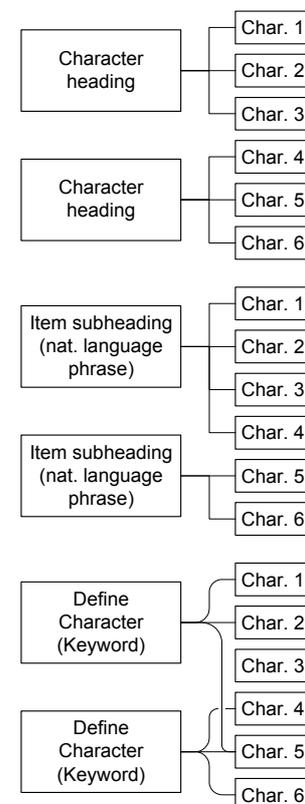


Figure 228. The different character-grouping mechanisms in DELTA.

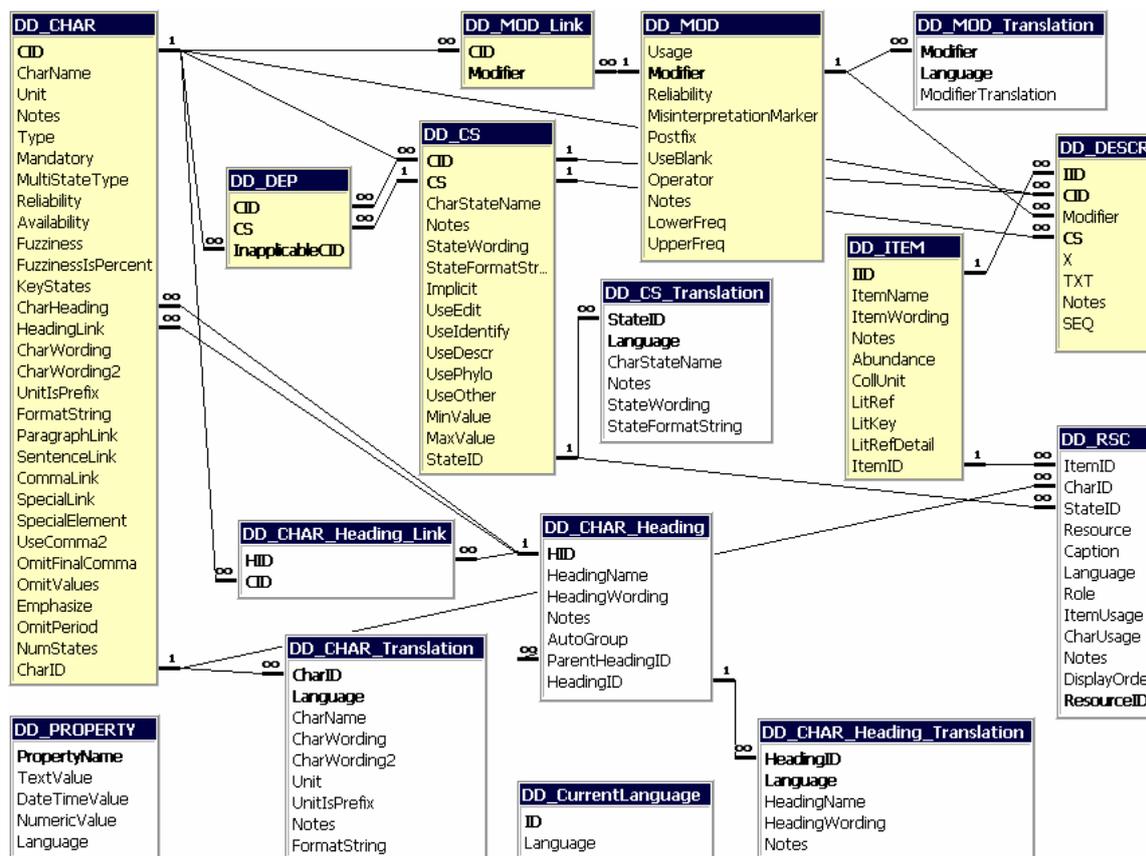


Figure 229. Complete entity relationship diagram of DiversityDescriptions 1.9 (physical model). Tables already presented in the previous overview diagram are lightly colored.

DELTA, as implemented in the CSIRO Confor and Intkey programs, uses three separate heading directives with special properties (Fig. 228). All three define only a single level of headings on top of the characters. Earlier versions of DiversityDescriptions accepted this limitation as well, but recent versions support hierarchical character headings. Despite this addition to DELTA, DiversityDescriptions aims at maintaining backwards compatibility with DELTA. Therefore, three different relations with the CHAR table exist, corresponding with the three DELTA directives. These are:

- DELTA “*Character Headings*”: This provides headings in front of characters and is intended for a list of character definitions. They are usually generally useful, e. g., to structure the character list in an editor or during identification. Each character may be member only of a single character heading and is represented by the 1 : n relation from CharHeading.HID to CHAR.CharHeading.
- DELTA “*Item Subheadings*”: The name is slightly deceptive; these are not headings to item names (which are not provided in DELTA), but character headings to be used when item descriptions are generated (especially natural language descriptions). Similar to character headings, each character may be member only of a single character heading; this is reflected in the 1 : n relation from CharHeading.HID to CHAR.HeadingLink. During natural language generation, the HeadingWording will be preferred over HeadingName. If no HeadingWording is defined, HeadingName is used here as well.
- DELTA “*Define Characters (Keyword)*”: This is an Intkey directive in the Intkey.ini file. Character Definitions are called “named character groups” in the DeltaAccess/Diversity-Descriptions documentation (Hagedorn 1999a). In contrast to the previous two heading directives, a single character may be a member of multiple such character groups. Thus headings may group characters under different aspects such as by organ of the organism,

by observation or analysis method, or by developmental stage. The n:m relation is implemented in the CHAR_Heading_Link table between CHAR and CHAR_Heading.

2. Resources: In the physical model the relatively complex logical model (Fig. 225, p. 332) of the Resources package has been denormalized and simplified to a single entity, “RSC”, by merging the various association classes into a single class (Fig. 230). RSC may contain one or several of item, character, or state references. If captions in multiple languages are desired, they may use multiple RSC records each using otherwise the identical information.

If multiple ID-references are present, the Role field applies to all, and Usage may be separately defined for characters and items (but not yet for states). Perhaps a better solution for usage may have been to use only a single field, and require multiple RSC records if usage differs for item, character and state reference.

Since any of ItemID, CharID, and StateID may be Null, the combination of these fields does not provide a composite primary key in RSC (like it did in DESCR for the combination of IID, CID, and CS).

Although the table is currently not referenced from other tables in the information model, a system-generated identifier ResourceID (“autonumber” or “autoincrement”) is defined for internal database uses.

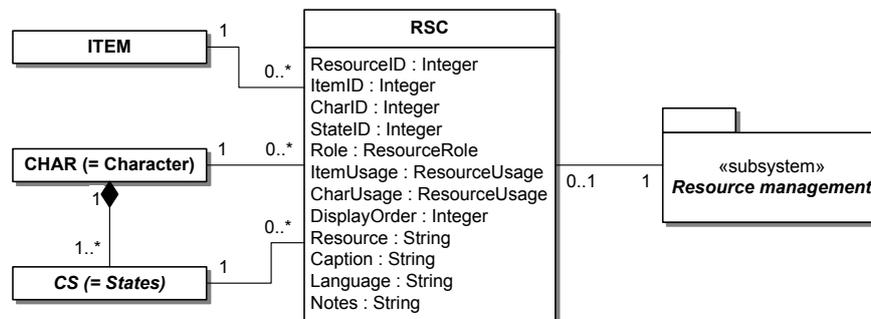


Figure 230. UML class diagram illustrating the simplified Resource table and its relations with other packages (physical model DiversityDescriptions 1.9).

3. Translations. For each character, heading, character state, and modifier an additional “... Translation” table is present. This table supports translations of language-dependent information into multiple languages. Note that the model currently still lacks a translation table for the free-form DESCR.Notes field. No project currently operating DiversityDescriptions had the resources to also translate these description-specific notes.

When updating the information model to support multiple languages in parallel it turned out to be impractical to remove the language-dependent fields from the original tables (e.g., CharName and StateName in CHAR and CS, respectively). Doing so would have made it impossible to maintain existing code inside DiversityDescriptions and other tools using the same information model. As an alternative solution, a “current language” was defined in the CurrentLanguage table. By means of a table constraint this table can contain only a single record and stores the currently preferred language. This language in CurrentLanguage is the language for which the text and labels are stored in the primary tables (such as CHAR, CS, MOD, and CHAR_Heading). Information for all other languages is stored in the “... Translation” tables.

It is possible to change the CurrentLanguage, provided methods are developed that first copy the current language information for CHAR, CS, etc. to the translation tables (with the old language value from CurrentLanguage), then copy the translation attributes to the current language attributes, and finally delete the copied information from the translations tables. Unquestionably this solution is not optimal, but it provides a migration path to continue using the original DiversityDescriptions code until a new solution based on the (still unfinished) SDD model can be provided.

Note that in the ITEM table some additional fields (CollUnit, LitRef) are shown, supporting precise relations with other components. As mentioned in the logic model, in the next version of DiversityDescriptions these will be replaced by a separate SCOPE table, enabling a greater range of links and more flexibility in the way links are expressed.

Two tables have no relations with other tables. The CurrentLanguage table (single value in single record) was already mentioned above. The PROPERTY table stores metadata, settings, and properties for an entire project. The field PropertyName contains values from an agreed vocabulary (which, however, is not controlled by the physical model) and the remaining fields provide a storage mechanism for text, date/time, or a number. Further information can be found in the section “Project Properties” (p. 352).

The diagrams above and the following data dictionary for the physical model for DiversityDescriptions are also available in a hyperlinked form (Hagedorn 2005b). ANSI SQL 92 code to create the model is provided in the appendix (p. 400); the online documentation further provides MS SQL Server T-SQL, PostgreSQL, and a simple w3c XML-schema for the model.

Data dictionary

The following tables provide the details about attributes and their properties. The column **Type** denotes data type of field content. '**Text (255)**' indicates a text of varying length for which no specific design restrictions have been formulated. 255 characters should be read as a proposed technical maximum limit; this may be changed if required by the database management system. In contrast, '**Memo**' is explicitly defined as text of unlimited length. All text is Unicode text. Numeric types: **Long** = 4 byte integer, **Integer** = 2 byte integer, **Byte+** = 1 byte positive integer (0 to 255).

The checkbox (or) in the column **Rqrd.** indicates that the attribute is required. The last column indicates presence and type of indices. **I** = The field is indexed to enable faster searching. Different types of indices are denoted by additional letters in parentheses: **I (P)** = attribute is part of the primary key (values in the index must be unique), **I (U)** = values in the index must be unique, **I (N)** = Null values are ignored in the index, **I (M)** = the index contains more than one attribute. + denotes that the attribute is involved in more than one multiple-field index.

Attributes and indices of the entity 'CHAR'

Character table = the central entity to define operational terminology.

Name	Type	Description / Default value & validation	Rqrd./Index
CID	Integer	Character ID number. Also currently defines the order of characters. This ID may change over time, please compare the separate CharID.	<input checked="" type="checkbox"/> I (P)
CharName	Text	Short name of character. <i>Validation rule: Not (Like ' %' Or Like '% ' Or Like '% %'). Validation message: Do not start or end with a blank. Do not include multiple blanks.</i>	<input checked="" type="checkbox"/> I (U)
Unit	Text	For numeric characters: an optional measurement unit like 'mm'. Only true units here, text like 'wide' belongs to CharWording2! <i>Validation rule: Is Null Or Not (Like ' %' Or Like '% ' Or Like '% %'). Validation message: Do not start or end with a blank. Do not include multiple blanks.</i>	<input type="checkbox"/> -
	<i>Example values (any other values may be added):</i>	m; dm; cm; mm; µm; kg; g; mg; µg; L; ml; µl; nl; °C; M; mM; µM; nM; pM; Mol; mMol; µMol; g/ml; Pa; hPa; bar; mbar; mm Hg; mS; µS; °; ';	
Notes	Memo	Character notes. <i>Validation rule: Is Null Or Not (Like ' %' Or Like '% ' Or Like '% %'). Validation message: Do not</i>	-

Attributes and indices of the entity 'CHAR'

Character table = the central entity to define operational terminology.

Name	Type	Description / Default value & validation	Rqrd./Index
		<i>start or end with a blank. Do not include multiple blanks.</i>	
Type	Text	Type of character: Text, Ordered/unordered multistate, Integer/Real numeric. <i>Default value: "UM"; Validation rule: "UM" Or "OM" Or "IN" Or "RN" Or "TE"</i>	I
	<i>Values restricted to:</i>		
	Code	Description	
	UM	Unordered multistate (categorical data/nominal scale)	
	OM	Ordered multistate (categorical data/ordinal scale)	
	TE	Textual data	
	IN	Integer numeric (cardinal scale)	
	RN	Real numeric (floating-point, interval scale)	
Mandatory	Boolean	Is the scoring of this character mandatory (required) in each item? <i>Default value: 0</i>	<input checked="" type="checkbox"/> -
MultiStateType	Byte+	Are multiple states allowed and how are they interpreted? <i>Default value: 1</i>	<input checked="" type="checkbox"/> I
	<i>Values restricted to:</i>		
	Label	Code	Description
	Excl.	0	states are exclusive = only a single state may be present in each item
	Or	1	combine states with OR operator (DELTA: '!')
	And	2	combine states with AND operator (DELTA: '&')
	To	3	states intergrade, e. g., '1 to 2 or 5 to 6' (DELTA: '1-2/5-6')
	To/And	4	states intergrade, e. g., '1 to 2 and 5 to 6' (DELTA: '1-2&5-6')
	With	5	states are a combination, e. g., 'green with some yellow'
Reliability	Single	Reliability (or weight) of character for identification, 1-10. <i>Default value: 5. Validation rule: (≥ 0 And ≤ 32) Or Is Null.</i>	<input checked="" type="checkbox"/> -
	<i>Values restricted to:</i>		
	Code	Description	
	0	ignore	
	1	very low	
	2	low	
	3	below average	
	4	slightly below average	
	5	standard (default value)	
	6	slightly above average	
	7	above average	
	8	high	
	9	very high	
Availability	Single	Availability (or accessibility) of character for identification, 1-10. This is an extension to the DELTA standard. <i>Default value: 5. Validation rule: (≥ 0 And ≤ 32) Or Is Null.</i>	<input checked="" type="checkbox"/> -
	<i>Values restricted to: (Identical with Reliability values above)</i>		
Fuzziness	Single	For identification: Unless a range is explicitly present, used to form a range around the mean (RN/IN) or state (OM). <i>Default value: 0</i>	<input checked="" type="checkbox"/> -
FuzzinessIsPercent	Boolean	Interpret 'Fuzziness' as 'percent' rather than 'absolute value' (e. g., Fuzziness=10 → range=mean ± 10%, instead of ± 10 absolute). <i>Default value: 0</i>	<input checked="" type="checkbox"/> -
KeyStates	Text	For use in a key: combine multistate char, into new combinations or define ranges for numeric char.	<input type="checkbox"/> I
CharHeading	Integer	A heading defined in the headings definition, inserted in char. def. output in front of the current character.	<input type="checkbox"/> I (N)
HeadingLink	Integer	A heading defined in the headings definition, inserted in natural language descriptions in front of descriptions using this character.	<input type="checkbox"/> I (N)
CharWording	Text	Natural language descript.: Wording to be used instead	-

Attributes and indices of the entity 'CHAR'

Character table = the central entity to define operational terminology.

Name	Type	Description / Default value & validation	Rqrd./Index								
		of CharName. <i>Validation rule: Is Null Or Not (Like ' %' Or Like ' % ' Or Like ' % %'). Validation message: Do not start or end with a blank. Do not include multiple blanks.</i>									
CharWording2	Text	Natural language descript.: Wording to be used AFTER states or values + unit, e. g., 'wide' for 'leaves 3-5 mm wide'. <i>Validation rule: Is Null Or Not (Like ' %' Or Like ' % ' Or Like ' % %'). Validation message: Do not start or end with a blank. Do not include multiple blanks.</i>	–								
UnitIsPrefix	Boolean	True if unit is to be placed in front of value, e. g., to render “pH 7.2”.	<input type="checkbox"/> –								
FormatString	Text	Default formatting for all states (compare StateFormatString). Esp. for numeric values (number of decimal places etc.). Standard Basic formatting string like “#.0”.	<input type="checkbox"/> –								
	<i>Example values (any other values may be added):</i>	<table border="1"> <thead> <tr> <th>Code</th> <th>Description</th> </tr> </thead> <tbody> <tr> <td>## #0</td> <td>numerical, no decimal places, 1000 separator = blank</td> </tr> <tr> <td>## #0.0</td> <td>numerical, 1 decimal place</td> </tr> <tr> <td>## #0.00</td> <td>numerical, 2 decimal places</td> </tr> </tbody> </table>	Code	Description	## #0	numerical, no decimal places, 1000 separator = blank	## #0.0	numerical, 1 decimal place	## #0.00	numerical, 2 decimal places	
Code	Description										
## #0	numerical, no decimal places, 1000 separator = blank										
## #0.0	numerical, 1 decimal place										
## #0.00	numerical, 2 decimal places										
ParagraphLink	Long	Natural language descript.: Define char. linked into a single paragraph. A new paragraph starts if group ID changes. <i>Default value: 1. Validation rule: > 0 Or Is Null.</i>	<input type="checkbox"/> –								
SentenceLink	Long	Natural language descript.: Define char. linked into a single sentence. A new sentence starts if group ID changes or whenever Null. <i>Validation rule: > 0 Or Is Null.</i>	<input type="checkbox"/> –								
CommaLink	Long	Natural language descript.: Define characters linked into a comma-enumeration ('sub-sentence'). A new group starts if group ID changes or whenever Null. <i>Validation rule: > 0 Or Is Null.</i>	<input type="checkbox"/> –								
UseComma2	Boolean	Natural language descript.: Use alternative comma separator between states of this character (e. g., for Chinese).	<input type="checkbox"/> –								
OmitFinalComma	Boolean	Natural language descript.: The final comma between character states is omitted ('1, 2, and 3' instead of '1, 2, and 3'). <i>Default value: 0</i>	<input checked="" type="checkbox"/> –								
OmitValues	Text	Natural language descript.: Omit lower ('-') or upper ('+') part of numeric character ranges. <i>Validation rule: Is Null Or '+' Or '-'</i>	<input type="checkbox"/> –								
	<i>Values restricted to:</i>	<table border="1"> <thead> <tr> <th>Code</th> <th>Description</th> </tr> </thead> <tbody> <tr> <td></td> <td>output full range in natural language description</td> </tr> <tr> <td>-</td> <td>omit lower range and min in natural language description</td> </tr> <tr> <td>+</td> <td>omit upper range and max in natural language description</td> </tr> </tbody> </table>	Code	Description		output full range in natural language description	-	omit lower range and min in natural language description	+	omit upper range and max in natural language description	
Code	Description										
	output full range in natural language description										
-	omit lower range and min in natural language description										
+	omit upper range and max in natural language description										
Emphasize	Boolean	Natural language descript.: Emphasize (italic or bold print) this character in all items. <i>Default value: 0</i>	<input type="checkbox"/> –								
OmitPeriod	Boolean	Natural language descript.: Omit the end delimiter (usually the period for a sentence, DELTA: 'OMIT PERIOD FOR CHAR'). <i>Default value: 0</i>	<input type="checkbox"/> –								
NumStates	Integer	Calculated field: Number of states other than special states U,V,- present in this character. Calculated automatically, do not edit! <i>Default value: 2</i>	<input checked="" type="checkbox"/>								
CharID	Autonom	Immutable unique number identifying a character (candidate key). Not exported to DELTA! Semantics are similar to CID, which, however, is exported to DELTA	<input type="checkbox"/> (U)								

Attributes and indices of the entity 'CHAR'

Character table = the central entity to define operational terminology.

Name	Type	Description / Default value & validation	Rqrd./Index
		and requires renumbering after character deletions.	
Index name:		Attributes & index properties	
CharHeading:		CharHeading (<i>Duplicates OK; Ignore Nulls</i>)	
CharID:		CharID (<i>Unique values</i>)	
CharName:		CharName (<i>Unique values</i>)	
CID:		CID (<i>Primary key; Unique values</i>)	
HeadingLink:		HeadingLink (<i>Duplicates OK; Ignore Nulls</i>)	
KeyStates:		KeyStates (<i>Duplicates OK</i>)	
MultiStateType:		MultiStateType (<i>Duplicates OK</i>)	
NumStates:		NumStates (<i>Duplicates OK</i>)	
Type:		Type (<i>Duplicates OK</i>)	
Relation type:		Attributes involved	
Cascading updates		CHAR_Heading.HID ↔ CHAR.CharHeading	
Cascading updates		CHAR_Heading.HID ↔ CHAR.HeadingLink	
Cascading updates & delet.		CHAR.CID ↔ CHAR_Heading_Link.CID	
Cascading updates & delet.		CHAR.CharID ↔ CHAR_Translation.CharID	
Cascading updates & delet.		CHAR.CID ↔ CS.CID	
Cascading updates & delet.		CHAR.CID ↔ DEP.InapplicableCID	
Simple relation (no integrity)		CHAR.CID ↔ DESCR.CID	
Cascading updates & delet.		CHAR.CID ↔ MOD_Link.CID	
Cascading updates & delet.		CHAR.CharID ↔ RSC.CharID	

Attributes and indices of the entity 'CHAR_Translation'

Character table, translations into multiple languages.

Name	Type	Description / Default value & validation	Rqrd./Index
CharID	Long	Character ID (unchanging version).	I (PM)
Language	Text	Language of the translation.	I (PM)
CharName	Text	Short label for character; in the current Language.	I
CharWording	Text	Natural language descript.: Wording to be used instead of CharName; in the current Language.	–
CharWording2	Text	Natural language descript.: Wording to be used AFTER states or values + unit, e. g., 'wide' for 'leaves 3-5 mm wide'; in the current Language.	–
Unit	Text	For numeric characters: an optional measurement unit like 'mm'. Only true units here, text like 'wide' belongs to CharWording2!	–
UnitIsPrefix	Boolean	True if unit is to be placed in front of value, e. g., to output "pH 7.2".	–
Notes	Memo	Character notes.	–
FormatString	Text	Default formatting for all states, but esp. for numeric values (number of decimal places etc.). Compare Char.FormatString for example values.	–

Attributes and indices of the entity 'CHAR_Translation'

Character table, translations into multiple languages.

Name	Type	Description / Default value & validation	Rqrd./Index
Index name:		Attributes & index properties	
CharNameTranslation:		CharName (<i>Duplicates OK</i>)	
CID:		CharID; Language (<i>Primary key; Unique values</i>)	
Relation type:		Attributes involved	
Cascading updates & delet.		CHAR.CharID ↔ CHAR_Translation.CharID	

Attributes and indices of the entity 'CHAR_Heading'

Character heading/Identification/HeadingLink table.

Name	Type	Description / Default value & validation	Rqrd./Index
HID	Integer	Character heading ID; determines sequence of headings when used in identification; change number to change that sequence!	I (P)
HeadingName	Text	Heading name, used for CharHeadings and Named character groups (identification). <i>Validation rule: Not (Like ' %' Or Like '% ' Or Like '% %')</i> . <i>Validation message: Do not start or end with a blank. Do not include multiple blanks.</i>	I (U)
HeadingWording	Text	Optional wording; preferred over HeadingName if headings for natural language descriptions are defined through the HeadingLink mechanism. <i>Validation rule: Is Null Or Not (Like ' %' Or Like '% ' Or Like '% %')</i> . <i>Validation message: Do not start or end with a blank. Do not include multiple blanks.</i>	–
Notes	Memo	Internal notes (not exported to DELTA format). <i>Validation rule: Is Null Or Not (Like ' %' Or Like '% ' Or Like '% %')</i> . <i>Validation message: Do not start or end with a blank. Do not include multiple blanks.</i>	–
AutoGroup	Text	Empty for user defined headings; else special code which is recognized during identification or a SQL query returning a set of character IDs. (HeadingName of predefined AutoGroups may be changed!).	–
ParentHeadingID	Long	A hierarchy of headings can be defined in Diversity-Descriptions by adding the higher hierarchy here. However, this is not supported by DELTA and cannot be exported.	I
HeadingID	Autonum	Immutable unique number identifying a heading (candidate key).	I (U)
Index name:		Attributes & index properties	
HeadingID:		HeadingID (<i>Unique values</i>)	
HeadingName:		HeadingName (<i>Unique values</i>)	
HID:		HID (<i>Primary key; Unique values</i>)	
ParentHID:		ParentHeadingID (<i>Duplicates OK</i>)	
Relation type:		Attributes involved	
Cascading updates & delet.		CHAR_Heading.HID ↔ CHAR_Heading_Link.HID	
Cascading updates & delet.		CHAR_Heading.HeadingID ↔ CHAR_Heading_Translation.HeadingID	
Cascading updates		CHAR_Heading.HeadingID ↔ CHAR_Heading.ParentHeadingID	
Cascading updates		CHAR_Heading.HID ↔ CHAR.CharHeading	

Attributes and indices of the entity 'CHAR_Heading'

Character heading/Identification/HeadingLink table.

Name	Type	Description / Default value & validation	Rqrd./Index
Cascading updates		CHAR_Heading.HID ↔ CHAR.HeadingLink	

Attributes and indices of 'CHAR_Heading_Translation'

Character headings, translations into multiple languages.

Name	Type	Description / Default value & validation	Rqrd./Index
HeadingID	Long	Character heading ID.	I (PM)
Language	Text	Language of the translation.	I (PM)
HeadingName	Text	Heading name, used for CharHeadings and Named character groups (identification). <i>Validation rule: Not (Like ' %' Or Like '% ' Or Like '% %'). Validation message: Do not start or end with a blank. Do not include multiple blanks.</i>	I (U)
HeadingWording	Text	Optional wording; preferred over HeadingName if headings for natural language descriptions are defined through the HeadingLink mechanism. <i>Validation rule: Is Null Or Not (Like ' %' Or Like '% ' Or Like '% %'). Validation message: Do not start or end with a blank. Do not include multiple blanks.</i>	–
Notes	Memo	Note on Translation. <i>Validation rule: Not (Like ' %' Or Like '% ' Or Like '% %'). Validation message: Do not start or end with a blank. Do not include multiple blanks.</i>	–
Index name:	Attributes & index properties		
CID:	HeadingID; Language (<i>Primary key; Unique values</i>)		
HeadingName:	HeadingName (<i>Unique values</i>)		
Relation type:	Attributes involved		
Cascading updates & delet.	CHAR_Heading.HeadingID ↔ CHAR_Heading_Translation.HeadingID		

Attributes and indices of the entity 'CHAR_Heading_Link'

Character groups for identification and linking.

Name	Type	Description / Default value & validation	Rqrd./Index
HID	Integer	Character heading ID to be linked.	<input checked="" type="checkbox"/> I (PM)
CID	Integer	Character ID to which the identification heading is applicable.	<input checked="" type="checkbox"/> I/I (PM)
Index name:	Attributes & index properties		
CID:	CID (<i>Duplicates OK</i>)		
HID:	HID; CID (<i>Primary key; Unique values</i>)		
Relation type:	Attributes involved		
Cascading updates & delet.	CHAR_Heading.HID ↔ CHAR_Heading_Link.HID		
Cascading updates & delet.	CHAR.CID ↔ CHAR_Heading_Link.CID		

Attributes and indices of the entity 'CS'

Character states for each character.

Name	Type	Description / Default value & validation	Rqrd./Index
CID	Integer	Character ID.	I (UM)+
CS	Text	Character state code. Usually pos. integer number or special codes for variable/unknown and for statistics (mean etc.). <i>Validation rule: Not (Like '%' Or Like '% ' Or Like '% %')</i> . <i>Validation message: Do not start or end with a blank. Do not include multiple blanks.</i>	I/I (PM)
CharStateName	Text	Name or description of character state. <i>Validation rule: Not (Like '%' Or Like '% ' Or Like '% %')</i> . <i>Validation message: Do not start or end with a blank. Do not include multiple blanks.</i>	I (UM)
Notes	Memo	Character state notes. <i>Validation rule: Is Null Or Not (Like '%' Or Like '% ' Or Like '% %')</i> . <i>Validation message: Do not start or end with a blank. Do not include multiple blanks.</i>	–
StateWording	Text	Wording to be used instead of CharStateName for natural language descriptions output. <i>Validation rule: Is Null Or Not (Like '%' Or Like '% ' Or Like '% %')</i> . <i>Validation message: Do not start or end with a blank. Do not include multiple blanks.</i>	–
StateFormatString	Text	State-specific formatting, overriding FormatString defined in the character definition. Compare Char.FormatString for example values.	–
Implicit	Boolean	Mark this state as a default which is automatically set. <i>Default value: 0</i>	–
<i>(Important note on the following five Use... attributes: These are not yet used in the DiversityDescriptions application and it remains questionable, whether using such attributes to identify state-subsets is a desirable solution!)</i>			
UseEdit	Boolean	Use this state during entering or updating of item descriptions. <i>Default value: True</i>	–
UseIdentify	Boolean	Use this state during identification. <i>Default value: True</i>	–
UseDescr	Boolean	Use this state for natural language item descriptions. <i>Default value: True</i>	–
UsePhylo	Boolean	Use this state for phylogenetic analysis. <i>Default value: True</i>	–
UseOther	Boolean	Use this to define a character state set for user-defined purposes. <i>Default value: True</i>	–
MinValue	Double	Numeric characters only: The lowest value of X in item description to be mapped to this state (inclusive). <i>Default value: -1E+308</i>	–
MaxValue	Double	Numeric characters only: The highest value of X in item description to be mapped to this state (inclusive). <i>Default value: 1E+308</i>	–
StateID	Autonum	Immutable unique number identifying a character state independently of the character. Not exported to DELTA! Preferred key for any external references to states not protected by cascaded referential updates. (Candidate key).	I (U)
Index name:	Attributes & index properties		
CID:	CID; CS (Primary key; Unique values)		
CS:	CS (Duplicates OK)		

Attributes and indices of the entity 'CS'

Character states for each character.

Name	Type	Description / Default value & validation	Rqrd./Index
CSNameUniqueInCID:		CharStateName; CID (<i>Unique values</i>)	
StateID:		StateID (<i>Unique values</i>)	
Relation type:		Attributes involved	
Cascading updates & delet.		CHAR.CID ↔ CS.CID	
Cascading updates & delet.		CS.StateID ↔ CS_Translation.StateID	
Cascading updates & delet.		CS.CID ↔ DEP.CID; CS.CS ↔ DEP.CS	
Cascading updates & delet.		CS.CID ↔ DESCR.CID; CS.CS ↔ DESCR.CS	
Cascading updates & delet.		CS.StateID ↔ RSC.StateID	

Attributes and indices of the entity 'CS_Translation'

Character states, translations into multiple languages.

Name	Type	Description / Default value & validation	Rqrd./Index
StateID	Long	State ID (foreign key). <i>Default value: 0</i>	I/I (PM)
Language	Text	Language of the translation.	I (PM)
CharStateName	Text	Name or description of character state. <i>Validation rule: Not (Like ' % ' Or Like ' % ' Or Like ' % % '). Validation message: Do not start or end with a blank. Do not include multiple blanks.</i>	–
Notes	Memo	Notes on translation. <i>Validation rule: Is Null Or Not (Like ' % ' Or Like ' % ' Or Like ' % % '). Validation message: Do not start or end with a blank. Do not include multiple blanks.</i>	–
StateWording	Text	Wording to be used instead of CharStateName for natural language descriptions output. <i>Validation rule: Is Null Or Not (Like ' % ' Or Like ' % ' Or Like ' % % '). Validation message: Do not start or end with a blank. Do not include multiple blanks.</i>	–
<i>Dynamically filled list: SELECT StateWording FROM X_CS ORDER BY StateWording</i>			
StateFormatString	Text	State-specific formatting, overriding FormatString defined in the character definition. Compare Char.FormatString for example values.	–

Index name: Attributes & index properties**PrimaryKey:** StateID; Language (*Primary key; Unique values*)**StateID:** StateID (*Duplicates OK*)**Relation type: Attributes involved**

Cascading updates & delet. CS.StateID ↔ CS_Translation.StateID

Attributes and indices of the entity 'MOD'

Modifier wordings for categorical or quantitative data.

Name	Type	Description / Default value & validation	Rqrd./Index
Usage	Text	Type of characters for which the modifier is applicable, e. g., General colors, Frequency of occurrence. <i>Dynamically filled list:</i> SELECT Usage FROM MOD GROUP BY Usage ORDER BY Usage	
Modifier	Text	Modifier wording for categorical or quantitative data, e. g., "mostly", "usually", "rarely".	(P)
Reliability	Byte+	Influence of modifier on data coded through categorical/quantitative char. states. More (>5) or less reliable (<5). <i>Default value: 5.</i> <i>Values restricted to:</i> (see list shown for CHAR.Reliability)	–
MisinterpretationMarker	Boolean	If set to true, the state to which this modifier is added is marked as being present only by misinterpretation of structure (phylloclade as leaf) or state (rough spore surface as smooth). <i>Default value: 0</i>	<input checked="" type="checkbox"/> –
Postfix	Boolean	Checked = output after the character state wording, unchecked = in front of it.	<input checked="" type="checkbox"/> –
UseBlank	Boolean	Checked = blank is added between the modifier wording and the item data text, else modifier added compress. <i>Default value: True</i>	<input checked="" type="checkbox"/> –
Operator	Byte+	>0 = Override operator, the normal operator between states is omitted when this modifier is encountered, e. g., to insert AND where normal operator would be OR. <i>Default value: 0. Validation rule: (>-1 And <4) Or 5</i> <i>Values restricted to:</i> 0 NONE 1 OR 2 AND 3 TO 5 WITH	<input checked="" type="checkbox"/> –
Notes	Text	Internal notes.	<input type="checkbox"/> –
LowerFreq	Single	The lower border of the frequency range, for freq. modifiers only. <i>Validation rule: (≥ 0 And ≤ 1) Or Is Null.</i>	<input type="checkbox"/> –
UpperFreq	Single	The upper border of the frequency range, for freq. modifiers only. <i>Validation rule: (≥ 0 And ≤ 1) Or Is Null.</i>	<input type="checkbox"/> –
Index name:	Attributes & index properties		
Modifier:	Modifier (<i>Primary key; Unique values</i>)		
Usage:	Usage (<i>Duplicates OK</i>)		
Relation type:	Attributes involved		
Cascading updates & delet.	MOD.Modifier ↔ DESCR.Modifier		
Cascading updates & delet.	MOD.Modifier ↔ MOD_Link.Modifier		
Cascading updates & delet.	MOD.Modifier ↔ MOD_Translation.Modifier		

Attributes and indices of the entity 'MOD_Translation'

Modifier wordings, translations into multiple languages.

Name	Type	Description / Default value & validation	Rqrd./Index
Modifier	Text	Modifier wording. Foreign key linking to _MOD. In current language!	<input checked="" type="checkbox"/> I (PM)
Language	Text	Language of the translation.	<input checked="" type="checkbox"/> I (PM)
ModifierTranslation	Text	Translation of modifier wording.	<input checked="" type="checkbox"/> -
Index name:		Attributes & index properties	
Modifier:		Modifier; Language (<i>Primary key; Unique values</i>)	
Relation type:		Attributes involved	
Cascading updates & delet.		MOD.Modifier ↔ MOD_Translation.Modifier	

Attributes and indices of the entity 'MOD_Link'

Links between characters and modifiers (n : m table).

Name	Type	Description / Default value & validation	Rqrd./Index
CID	Integer	Character ID to which the modifier is applicable.	<input checked="" type="checkbox"/> I/I (PM)
Modifier	Text	Modifier wording for categorical or quantitative data, e. g., "mostly", "usually", "rarely".	<input checked="" type="checkbox"/> I (PM)
Index name:		Attributes & index properties	
CID:		CID (<i>Duplicates OK</i>)	
Modifier:		Modifier; CID (<i>Primary key; Unique values</i>)	
Relation type:		Attributes involved	
Cascading updates & delet.		CHAR.CID ↔ MOD_Link.CID	
Cascading updates & delet.		MOD.Modifier ↔ MOD_Link.Modifier	

Attributes and indices of the entity 'DEP'

Dependent (= inapplicable) characters for each character state.

Name	Type	Description / Default value & validation	Rqrd./Index
CID	Integer	Controlling character ID.	<input checked="" type="checkbox"/> I (PM)
CS	Text	Controlling character state.	<input checked="" type="checkbox"/> I (PM)
InapplicableCID	Integer	CID of dependent character, i. e. inapplicable for any item where current CID/CS combination is used.	<input checked="" type="checkbox"/> I/I (PM)
Index name:		Attributes & index properties	
CID:		CID; CS; InapplicableCID (<i>Primary key; Unique values</i>)	
InapplicableCID:		InapplicableCID (<i>Duplicates OK</i>)	
Relation type:		Attributes involved	
Cascading updates & delet.		CHAR.CID ↔ DEP.InapplicableCID	
Cascading updates & delet.		CS.CID ↔ DEP.CID; CS.CS ↔ DEP.CS	

Attributes and indices of the entity 'ITEM'

Defines header to individual descriptions, i. e. item, object, specimen, class, taxon, etc.

Name	Type	Description / Default value & validation	Rqrd./Index
IID	Long	Item ID.	<input checked="" type="checkbox"/> I (P)
ItemName	Text	Name or description of item (incl. taxon authors if necessary), link to taxonomic subsystem. <i>Validation rule: Not (Like ' %' Or Like '% ' Or Like '% %'). Validation message: Do not start or end with a blank. Do not include multiple blanks.</i>	<input checked="" type="checkbox"/> I
ItemWording	Text	Natural language descript.: Wording to be used instead of ItemName. <i>Validation rule: Is Null Or Not (Like ' %' Or Like '% ' Or Like '% %'). Validation message: Do not start or end with a blank. Do not include multiple blanks.</i>	<input type="checkbox"/> -
Notes	Memo	Item notes. <i>Validation rule: Is Null Or Not (Like ' %' Or Like '% ' Or Like '% %'). Validation message: Do not start or end with a blank. Do not include multiple blanks.</i>	<input type="checkbox"/> -
Abundance	Single	Abundance, relative importance (or weight) of item, 1-10. <i>Default value: 5. Validation rule: (≥ 0 And ≤ 10) Or Is Null.</i>	<input checked="" type="checkbox"/> I
<i>Values restricted to:</i>			
	Code	Description	
	0	ignore	
	1	very low	
	2	low	
	3	below average	
	4	slightly below average	
	5	standard (default value)	
	6	slightly above average	
	7	above average	
	8	high	
	9	very high	
CollUnit	Text	Unit code in specimen collection, link into collection subsystem. (Not Defined In DELTA!)	<input type="checkbox"/> I (N)
LitRef	Text	Literature reference (user-readable text form). (Not Defined In DELTA!)	<input type="checkbox"/> I (N)
LitKey	Long	Literature reference (numeric link into literature reference subsystem). (Not Defined In DELTA!)	<input type="checkbox"/> I (N)
LitRefDetail	Text	Reference detail, like page(s) of interest, specific figures, etc.	<input type="checkbox"/> -
ItemID	Autonum	Immutable unique number identifying an item (candidate key). Not exported to DELTA! Semantics are similar to IID, which, however, is exported to DELTA and requires renumbering after deletions.	<input type="checkbox"/> I (U)

Index name: Attributes & index properties**Abundance:** Abundance (*Duplicates OK*)**CollUnit:** CollUnit (*Duplicates OK; Ignore Nulls*)**IID:** IID (*Primary key; Unique values*)**ItemID:** ItemID (*Unique values*)**ItemName:** ItemName (*Duplicates OK*)**LitKey:** LitKey (*Duplicates OK; Ignore Nulls*)**LitRef:** LitRef (*Duplicates OK; Ignore Nulls*)**Relation type: Attributes involved**

Cascading updates & delet. ITEM.IID ↔ DESCR.IID

Cascading updates & delet. ITEM.ItemID ↔ RSC.ItemID

Attributes and indices of the entity 'DESCR'

Description data for each item.

Name	Type	Description / Default value & validation	Rqrd./Index
IID	Long	Item ID.	I/I (PM)
CID	Integer	Character ID.	I (PM)
Modifier	Text	Modifier wording for categorical or quantitative data, e. g., "mostly", "usually", "rarely". <i>Validation rule: Is Null Or Not (Like ' %' Or Like '% ' Or Like '% %'). Validation message: Do not start or end with a blank. Do not include multiple blanks.</i>	–
CS	Text	Character state code. Usually pos. integer number or special codes for variable/unknown and for statistics (mean etc.).	<input checked="" type="checkbox"/> I/I (PM)
X	Double	Numeric value, defined by CS.	<input type="checkbox"/> I (N)
TXT	Memo	Text data. <i>Validation rule: Is Null Or Not (Like ' %' Or Like '% ' Or Like '% %'). Validation message: Do not start or end with a blank. Do not include multiple blanks.</i>	<input type="checkbox"/> –
Notes	Memo	Notes and additional information, included in natural language item descriptions. <i>Validation rule: Is Null Or Not (Like ' %' Or Like '% ' Or Like '% %'). Validation message: Do not start or end with a blank. Do not include multiple blanks.</i>	<input type="checkbox"/> –
SEQ	Long	Sequence of character states for the item.	<input type="checkbox"/> I
Table validation rule:		(Not IsNull(TXT) And IsNull(X) And CS='TE') Or (IsNull(TXT) And (IsNull(X) Or Not IsNull(X) And Not IsNumeric(CS)))	
Table validation text:		You may not enter data in both TXT and X, use only one of them! To enter TXT data, use CS='TE', to enter numeric data use the appropriate CS attributes.	
Index name:		Attributes & index properties	
CID:		CID; CS; IID (<i>Primary key; Unique values</i>)	
CS:		CS (<i>Duplicates OK</i>)	
IID:		IID (<i>Duplicates OK</i>)	
SEQ:		SEQ (<i>Duplicates OK</i>)	
X:		X (<i>Duplicates OK; Ignore Nulls</i>)	
Relation type:		Attributes involved	
Simple relation (no integrity)		CHAR.CID ↔ DESCR.CID	
Cascading updates & delet.		CS.CID ↔ DESCR.CID; CS.CS ↔ DESCR.CS	
Cascading updates & delet.		ITEM.IID ↔ DESCR.IID	
Cascading updates & delet.		MOD.Modifier ↔ DESCR.Modifier	

Attributes and indices of the entity 'RSC'

External resources stored as files or URL, e. g., illustrations for characters/items.

Name	Type	Description / Default value & validation	Rqrd./Index
ItemID	Long	ID of associated item (optional). This refers to ItemID, not IID!	<input type="checkbox"/> I (N)
CharID	Long	ID of associated character (optional). This refers to CharID, not CID!	<input type="checkbox"/> I (N)
StateID	Long	ID of associated character state (optional). This refers to	<input type="checkbox"/> I (N)

Attributes and indices of the entity 'RSC'

External resources stored as files or URL, e. g., illustrations for characters/items.

Name	Type	Description / Default value & validation	Rqrd./Index															
		StateID, not CS!																
Role	Text	Roles the resource is intended for: I = Icon, S = Selector (displayed directly, e. g., to select a state), D = Definition (usually displayed only as thumbnail image or link for further information). <i>Default value: "S". Validation rule: In ("I", "S", "D")</i>	–															
<i>Values restricted to:</i>		<table border="1"> <thead> <tr> <th>Code</th> <th>Label</th> <th>Description</th> </tr> </thead> <tbody> <tr> <td>I</td> <td>Icon</td> <td>Small image; usually not informative enough to make a choice without a text label. Only a single resource per item/char./state should be defined as icon.</td> </tr> <tr> <td>S</td> <td>Selector</td> <td>Medium-sized and sufficient to make a selection without text. Multiple resources per item/char./state may have this role.</td> </tr> <tr> <td>D</td> <td>Definition</td> <td>Detailed supplementary information used to define an item, char., or state.</td> </tr> <tr> <td>A</td> <td>Association</td> <td>Associated information that is considered useful to relate.</td> </tr> </tbody> </table>	Code	Label	Description	I	Icon	Small image; usually not informative enough to make a choice without a text label. Only a single resource per item/char./state should be defined as icon.	S	Selector	Medium-sized and sufficient to make a selection without text. Multiple resources per item/char./state may have this role.	D	Definition	Detailed supplementary information used to define an item, char., or state.	A	Association	Associated information that is considered useful to relate.	
Code	Label	Description																
I	Icon	Small image; usually not informative enough to make a choice without a text label. Only a single resource per item/char./state should be defined as icon.																
S	Selector	Medium-sized and sufficient to make a selection without text. Multiple resources per item/char./state may have this role.																
D	Definition	Detailed supplementary information used to define an item, char., or state.																
A	Association	Associated information that is considered useful to relate.																
ItemUsage	Text	Usage of resource in the context of items (including natural language descriptions). Usage is especially relevant if both Item and CharIDs are defined, but resource is relevant for entire item (e. g., habit). <i>Validation rule: Is Null Or In ("1", "2", "3")</i>	–															
<i>Values restricted to:</i>		<table border="1"> <thead> <tr> <th>Code</th> <th>Label</th> <th>Description</th> </tr> </thead> <tbody> <tr> <td>1</td> <td>Always</td> <td>Always include resource (especially for character in char. definition, for item in natural language report)</td> </tr> <tr> <td>2</td> <td>Desirable</td> <td>It is desirable to include resource if space permits (e. g., in web forms, in addition to usage "always")</td> </tr> <tr> <td>3</td> <td>Secondary</td> <td>Resource of secondary relevance (e. g., in web forms on request; in case of characters: e. g., state resources not illustrating character as a whole)</td> </tr> </tbody> </table>	Code	Label	Description	1	Always	Always include resource (especially for character in char. definition, for item in natural language report)	2	Desirable	It is desirable to include resource if space permits (e. g., in web forms, in addition to usage "always")	3	Secondary	Resource of secondary relevance (e. g., in web forms on request; in case of characters: e. g., state resources not illustrating character as a whole)				
Code	Label	Description																
1	Always	Always include resource (especially for character in char. definition, for item in natural language report)																
2	Desirable	It is desirable to include resource if space permits (e. g., in web forms, in addition to usage "always")																
3	Secondary	Resource of secondary relevance (e. g., in web forms on request; in case of characters: e. g., state resources not illustrating character as a whole)																
CharUsage	Text	Usage of resource in the context of items (especially character definition in print or web form). Usage is especially relevant if also Item or StateIDs are defined, but resource is relevant for entire character as well. <i>Validation rule: Is Null Or In ("1", "2", "3")</i>	–															
<i>Values restricted to:</i>		(see values for ItemUsage above)																
Caption	Memo	Caption for the resource, e. g., a text to display while showing an illustration or a video.	–															
Language	Text	Language of the translation.	–															
Notes	Memo	Internal notes (perhaps also formatting commands for INTKEY).	–															
Resource	Text	Filename of illustration (photo/drawing/graph) or other media resources (see ResourceDefaultPath / DefaultURL in table _PROPERTY for global paths).	I															
ResourceID	Autonom	Media resource ID (any of item, char, or state IDs may be missing)	I (P)															
DisplayOrder	Long	A positive number that can be used to define the sequence in which multiple resources are displayed. <i>Default value: 0</i>	–															
Index name:		Attributes & index properties																
CID:		CharID (<i>Duplicates OK; Ignore Nulls</i>)																
CS:		StateID (<i>Duplicates OK; Ignore Nulls</i>)																
IID:		ItemID (<i>Duplicates OK; Ignore Nulls</i>)																
Resource:		Resource (<i>Duplicates OK</i>)																
RID:		ResourceID (<i>Primary key; Unique values</i>)																

Attributes and indices of the entity 'RSC'

External resources stored as files or URL, e. g., illustrations for characters/items.

Name	Type	Description / Default value & validation	Rqrd./Index
Relation type:		Attributes involved	
Cascading updates & delet.	CHAR.CharID	↔ RSC.CharID	
Cascading updates & delet.	CS.StateID	↔ RSC.StateID	
Cascading updates & delet.	ITEM.ItemID	↔ RSC.ItemID	

Attributes and indices of the entity 'PROPERTY'

General header information about the project.

Name	Type	Description / Default value & validation	Rqrd./Index
PropertyName	Text	The name of the project property (do not change!).	I (P)
TextValue	Memo	Text information.	–
DateTimeValue	Date/Time	Date or time information, e. g., of last update.	–
NumericValue	Double	Numerical information, stored as real number.	I (N)
Language	Text	Language of a property translation, e. g., for project title. <i>Default value: 'en'</i>	–
Index name:		Attributes & index properties	
NumericValue:		NumericValue (<i>Duplicates OK; Ignore Nulls</i>)	
PropertyName:		PropertyName (<i>Primary key; Unique values</i>)	

Attributes and indices of the entity 'CurrentLanguage'

Definition of a single language as the current working language, determines which language is displayed in editing forms, reports, etc. The table supports only a single record!

Name	Type	Description / Default value & validation	Rqrd./Index
ID	Long	ID (restricted to a single record, ID must always be 1!). <i>Default value: 1</i>	I (P)
Language	Text	2 character ISO language code.	–
Index name:		Attributes & index properties	
PropertyName:		ID (<i>Primary key; Unique values</i>)	

Project Properties

The project properties (project metadata) are stored as name-value pairs in the table PROPERTY. Although they are therefore not listed in the information model above, they form an important part of the overall model of DiversityDescriptions. Implementing them as name-value pairs was chosen to improve the extensibility of the model. Except for a rarely used title/heading directive, the DELTA data exchange standard itself provided no meta-information about a project, not even a version number. As a consequence, all project properties initially were application-specific and evolved strongly with successive versions of DeltaAccess/DiversityDescriptions. The name-value model supported this evolution with minimal refactoring.

The following tables list the project properties supported by DiversityDescriptions 1.9. The first column informs about the intended status of the property:

R = required property considered generally useful (should be included in future data exchange formats)

O = optional property considered generally useful (should be included in future formats)

A = optional application-specific properties (may be present in exchange formats through application-specific extensions)

I = internal use in a database only, not used in file-based exchange format.

Two important properties that do not apply to all projects (*ProjectEditors* and *ProjectVersion*) are considered *required*, but can be left empty. If no editor or version exists, these properties must be present without a value, to identify the situation that positively no editor or version number exists.

Project objective and scope

A potential user needs means to learn more about a project and answer the questions: What is this project about? Is it relevant for me? Possible properties are: Project title, subtitle, description, abstract, cross reference to a project homepage on the internet, perhaps keywords and project version (i. e. version of the descriptive data, relevant here to determine which the most recent version available is). The following have so far been used in DiversityDescriptions:

PropertyName	Description
R ProjectName	Short title < 20 characters, without blanks, period, colon, etc.
R ProjectTitle	Title/long name of project (imported from & exported to DELTA Heading directive)
O ProjectDescription	Description or short introduction to project
O ProjectComments	Comments concerning the project (imported from and exported to DELTA Comment directive)
R ProjectVersion	Version number or code for the project data set. DiversityDescriptions does not support separate versions for character definition and item description yet.
O ProjectHomePageURL	Main descriptive document about the project, project home page.
O ProjectIcon	Path to a bitmap containing an icon or symbol for the project. The size should be between 16 x 16 and 64 x 64. Use the GIF-format, if the icon shall be used in a web interface.

Documentation of intellectual property rights

These properties provide a simple means to document intellectual property rights. In the case of large projects additional, item-specific documentation is necessary to document which part of the data set was prepared or edited by whom. In DiversityDescriptions this is supported only as free-form notes.

PropertyName	Description
R ProjectAuthors	Authors of the project character definition and item description data
R ProjectCopyright	Copyright notice for the project character definition and item description data
R ProjectEditors	Can be used together with <i>ProjectAuthors</i> or as an alternative. Appropriate if the data are collected by a larger group of scientists, some of which function as editors. Compare quality control below.
O Acknowledgments	Acknowledgement of significant contributions to the work by persons or organizations who are neither authors nor editors.

Documentation of technical standards followed in data exchange format

The exchange format that is being used (name and version), possible with extra information if not already defined by format, e. g., character set used, or text format coding (None / RTF / HTML) used. In addition to the documentation of the exchange format, it is desirable to identify the application name and version of the exporting application. This can be especially helpful, if different applications interpret a standard differently or erroneously (which is likely to occur in the case of a complex exchange format).

PropertyName	Description
R FormatName	The name of the information model used internally. Several applications may have agreed on a common information model to simplify data exchange.
R FormatVersion	The version number of DiversityDescriptions, e. g., "1.4". DateTimeValue contains the date and time at which the project was created.
R ApplicationName	<i>In a database:</i> The name of the application which created the project and which has the right to modify the definition (other, independent applications can have direct access to the data base and may have the right to modify item descriptions, but not the character definition). <i>In a file based exchange format:</i> The name of the application from which the data were exported.
R ApplicationVersion	Version of application identified by ApplicationName
R CharSet	The character set used to store/export the data, e. g., Unicode, ANSI (= Windows 3.1/95/98/NT), IBM-OEM (= DOS-'ASCII'), ISO-Latin 1, HTML character entities ('Å' = 'Ä,' etc.). This property is only necessary, if the exchange format is not specific about the character set to be used.
O CharFormatCoding	The coding scheme used to encode character formatting (bold, italic, etc.) in various wording fields. Possible values are: NONE, BASIC, HTML, XML, CSIRO, RTF. BASIC supports a limited number of basic HTML/XHTML tags. HTML refers to full HTML, i. e. no conversions are applied any more. The old proprietary typesetting coding scheme of the CSIRO CONFOR program is being phased out and replaced by an RTF variant (termed "CSIRO" here). DiversityDescriptions currently uses a 6-part code separately for ItemWording, HeadingWording, CharWording (incl. StateWording), CharNotes, ItemDescription Text and ItemDescription Notes. Example: "BASIC; BASIC; HTML; BASIC; BASIC; NONE;".

Internal application settings (exported)

These properties are written into export files, but can be ignored if the data exchange occurs between different applications. However, if the same application (of the same or a later version) imports a data set, many specialized settings are desirably to be maintained. Examples are preferences for analysis, natural language reporting, or HTML form creation, or the date of export and internal information about the source project that was exported.

PropertyName	Description
Saved preferences:	
A ExportExchangeFormatFile	File name of the most recent export to an exchange format.
A ExportExchangeFormatName	The preferred or most recently used export format. Examples: see ImportExchangeFormatName below
A ExportExchangeFormatVersion	The version associated with ExportExchangeFormatName
A ExportExchangeFormatDate	The date of the most recent export to an exchange format
A Options_ToForm_Default	Saved options/parameters of the HTML form generation process.
A Options_ToNat_Default	Saved options/parameters of the natural language item description process.
Information about origin of project:	
A ImportExchangeFormatFile	Name of the file from which the data were originally imported. For many applications the file name will be significant (NEXUS, DELTA as used by Pankey/Pandora, etc.). In the case of DELTA/CONFOR compatible files, the file names are always the same (chars, items, specs, etc.). Storing the full path, may convey the necessary information in these cases.
A ImportExchangeFormatName	The format used when the data were originally imported using an exchange format. Examples: Nexus or DELTA. Not used when created directly inside the application.
A ImportExchangeFormatVersion	Examples: 2 for Nexus 2, 1 for Nexus versions before Nexus 2 (even though no number was defined at that time)
A ImportExchangeFormatDate	Date/time on which the import occurred. (Compare also the internal properties <i>ExportExchangeFormatName / Version / Date</i>)
A ImportExchangeSkippedDirectives	Text of all unrecognized information in the import file (e. g., DELTA directives that could not be processed.
A ImportExchangeWarnings	The text of the warnings displayed by the import procedure after import
A CopyOfCharProject	Source of character definition if project was created by copying (with or without subset restriction) from another project
A CopyOfItemProject	Source of item definition/data if project was created by copying (with or without subset restriction) from another project

PropertyName	Description
If project is a subset of/link to another project:	
A SubsetCharBaseProject	Applicable to character subsets: The name of the base project of which the current project is a subset.
A SubsetItemBaseProject	Applicable to item subsets: The name of the base project of which the current project is a subset.
A SubsetCharRestriction	Applicable to character subsets: The definition of the subset, either a dynamic where-clause or a list of ID numbers to be included. (Only used in the exchange format, not as a project property in the database.)
A SubsetItemRestriction	Applicable to item subsets: The definition of the subset, either a dynamic where-clause or a list of ID numbers to be included. (Only used in the exchange format, not as a project property in the database.) [NOTE: The distinction between dynamic and static subsets could either be evident from the form of the subset restrictions above, or could alternatively be explicitly stated in a separate project property.]
A SubsetIncludeNewChars	Applicable to character subsets: characters added to the base project are automatically included in the subset (values: Yes, No = default).
A SubsetIncludeNewItem	Applicable to item subsets: characters added to the base project are automatically included in the subset (values: Yes, No = default).

Internal application settings (not exported)

The property mechanism is also used to store internal information that is not exported, like information about recent exports, the status of compilation for identification, local settings about the path of the preferred browser, the default publishing or image resources path/URLs.

PropertyName	Description
I IdentifyLastCompilation	Date, perhaps state of project when the project was last compiled to be used for identification purposes.
I LastBackup	Date and file name of last backup to a backup file
I LastExport	Date and file name of last export, e. g., to a DELTA or NEXUS file
I LastRestore	Date and file name of last restore from a backup file
External resources and paths:	
I ResourceDefaultPath	Default path if no specific pathname is given; refers to Resource attribute of _RSC
I ResourceDefaultURL	As above; alternative path used for internet access; should point to identical resources
I ExternalBrowser	Path and application name of external browser to display URLs. Example: "C:\programs\Netscape\netscape.exe"
I DefaultPublishingPath	Default folder to write reports and exports into. If missing, the current directory in which the project database resides is assumed.
I DefaultPublishingURL	Default folder to write reports and exports into. If missing, the current directory in which the project database resides is assumed.

Note: Several additional aspects were planned for DiversityDescriptions, but have not been implemented in the versions published so far. These are:

- **Documentation of quality control methods and standards.** The existence of an *editor* under *intellectual property rights* (see above) is only a secondary indication that a quality control process has taken place. Ideally, this process should be documented in further detail. As long as no standards are generally accepted in the systematic biology community (where GLP = good laboratory practice standards are rarely used), a free text documentation probably serves this purpose best.
- **Documentation of standards followed in the character definition.** In the future it will be increasingly important to define standard character definitions. A standard character definition can be used (a) in its entirety, (b) only in part but unchanged, or (c) with some characters modified. It is planned to provide mechanisms to document the adherence to standards on a character-by-character basis as part of the character definition. However, some information will be global and it may thus be desirable to document it here.

Statistical measures in DiversityDescriptions

The DiversityDescriptions model differs substantially from DELTA in respect to the handling of quantitative measurements and their aggregation methods (statistical measures). The DELTA standard defines the following attributes for numerical characters: Central value, a lower limit of a range, an upper limit of a range, a minimum, and a maximum value. These are expressed as a single string in the following format: “(” Minimum “-)” Lower range limit “-” Central value “-” Upper range limit “(-” Maximum “)”, where Minimum, Maximum, and either Central value or both range limits may be missing. The semantics of the central value (e. g., single measurement, mean, mode, or median) and the two range limits (e. g., mean \pm standard deviation, a 95% confidence interval, a human estimate of what is “typical”, etc.) cannot be defined in DELTA.

DiversityDescriptions defines statistical measures (originally called “numerical or statistical attributes” in the DeltaAccess/DiversityDescriptions documentation, Hagedorn 1999a) similar to character states of categorical characters. The set of measures used in DELTA is called the “Standard or Minimal set” in DiversityDescriptions. In principle, however, any desired statistical measure can be used in DiversityDescriptions. For certain analytical or report-generation purposes a longer list of statistical measures receives special treatment. These are predefined as an “Extended set of numerical attributes” (Table 60, selectable during import of a DELTA file or from a pick list in the character state subform).

Upon export from DiversityDescriptions to DELTA, all ranges, percentiles, or confidence intervals must be converted to unspecified ranges, and the distinction between mean, median, etc., will be lost. The sample size N and other extra attributes can only be exported as comments.

Table 60. Statistical measures recognized by algorithms in DiversityDescriptions.

Symbol	Description	Symbol	Description
Min	Minimum value	+SD	Range: mean plus 1 standard deviation
-Low	Lower value of unspecified range	-SD	Range: mean minus 1 standard deviation
Mean	Mean (average)	-CI 95	Range: Lower limit of 95% confidence interval for mean
+High	Upper value of unspecified range	+CI 95	Range: Upper limit of 95% confidence interval for mean
Max	Maximum value	-CI 90	Range: Lower limit of 90% confidence interval for mean
Median	Median (2nd quartile)	+CI 90	Range: Upper limit of 90% confidence interval for mean
Mode	Mode	+Q90	Range: Upper limit of 90% interval (= 95% percentile)
SD	Standard deviation of sample (df = n-1)	-Q90	Range: Lower limit of 90% interval (= 5% percentile)
SE	Standard error of mean	+Q80	Range: Upper limit of 80% interval (= 90% percentile)
Val	Single value (i. e. sample size N=1)	-Q80	Range: Lower limit of 80% interval (= 10% percentile)
N	Sample size	+Q50	Range: Upper limit of interquartile range (= 3rd quartile)
		-Q50	Range: Lower limit of interquartile range (= 1st quartile)

8. Final Discussions

The present thesis aims to reassess the position of descriptive data in the context of biodiversity information frameworks and to reanalyze the requirements for a general information model for descriptive data. The goal is a model that is not focused on a specific organismic group or purpose, but truly suitable for the biodiversity of organisms and the corresponding diversity of methods to study them. Towards this goal several contributions are presented, which approach the topic from different angles. Much of the discussion is presented throughout the work, and some notes on the “Scope, motivation, and constraints of the current work” have already been given on p. 15, but an additional discussion is in this section.

8.1. A progression of information models

It was found unavoidable to approach the subject both from a fundamental angle and from a practical angle of implemented or proposed models. The diversity of implemented and proposed models, and often the insufficient information on existing models made it difficult to achieve a fully consistent treatment of individual topics, but restricting the view to the study of systems, one may distinguish the following layers in the thesis:

- A background of DELTA and NEXUS, the classical exchange formats for descriptive data.
- Implemented and proposed systems trying to improve on these (e. g., New DELTA, Lucid, DiversityDescriptions). Of these, DiversityDescriptions, an original contribution of the author, is documented in detail in this thesis.
- Alternative proposals investigating a strongly different approach than DELTA, especially Nemisys/Genisys (p. 21) and the Prometheus description model (p. 21).
- The development of a new XML-based data exchange format for descriptive data (SDD). Although SDD is not documented in detail in this thesis, the goals and perspective of the SDD development may help to understand the choice of requirement analysis topics presented in this thesis. SDD attempts to reconcile various purposes of descriptive data (e. g., diagnosis and identification, phylogenetic and other analyses), and serve various existing applications.

The goal of broadening the perspective and analyzing the requirements for a general information model is a dominant topic in this thesis; it is discussed further below in the section “Results of requirement analyses” (p. 361).

Background of DELTA and NEXUS

DELTA has been the prevalent format for descriptive data for natural language descriptions and identification purposes for several decades. None of what DELTA successfully and often pragmatically achieves was to be repeated here. However, several issues with DELTA lead to a need to redefine future concepts and exchange formats:

- The format of DELTA has some technical limitations making it difficult to parse and extend, which could be easily addressed by creating a close analogue of DELTA in xml as proposed in the first XDELTA drafts (Dodds 1999). XDELTA never was widely used, because most members in the community desired a more substantial revision of DELTA.
- DELTA is designed more as a programming language, providing processing commands, than as a data exchange format. The DELTA standard in the narrow sense (Dallwitz & Paine 1999, 2005) is limited to five data directives (Character list, Character types, Item descriptions, Dependent characters, Implicit values) plus four data set metadata directives (Number of characters/states, Maximum number of items/states). However, The CSIRO DELTA program suite, includes over 170 Confor directives (Dallwitz & al. 2000a) and over 70 Intkey directives (Dallwitz & al. 2000b) supporting the actual functionality. Some of these are clearly program-specific processing instructions, while others contain general data or metadata that could be used by a wide variety of processors. The exact border is often difficult to establish, in the author’s estimate 65 Confor and three Intkey directives should be considered part of a data exchange standard.
- Despite this wide array of functionality, several deficits had been analyzed over time, requiring some fundamental changes in DELTA which were proposed as “New DELTA” (p. 20).

Thus, a simple and successful core system existed, but it had somewhat outgrown its original design goals. A redesign and reconsideration of the assumptions was considered necessary.

The NEXUS format (p. 18) is at least as widespread as DELTA, although largely limited to phylogenetic analysis purposes. It has a slightly better syntax than DELTA, enabling extensibility. The fundamental features of NEXUS for storing coded data are largely a subset of what

DELTA supports. However, many advanced features for special phylogenetic analysis methods are provided; these have not been studied in the present analysis.

DiversityDescriptions

The relational entity-relationship model for DiversityDescriptions is an evolutionary step based on DELTA. This model is documented here, forming a major result of this thesis (“Information model for DiversityDescriptions 1.9”, p. 322).

DiversityDescriptions attests that it is possible to translate the “processing-instruction-model” of DELTA into a relational information model. The relational model is tested and successfully solves the problems that the CSIRO DELTA programs address. In the context of a requirement analysis it serves as a baseline, translating the information items identified in over 25 years of DELTA development into a form that can be – perhaps – easier studied.

Among all the models considered in this thesis, the DiversityDescriptions information model is unique with respect to depth of supporting the DELTA language in the broad sense (i. e., including directives used by the CSIRO suite of DELTA programs). Clearly this is difficult to prove, since many important databases are yet undocumented (e. g., Biolink, p. 22; ActKey, p. 20 and Fig. 140, p. 254) or – due to commercial or patent issues – even unlikely to become documented (ALICE, p. 22, FRIDA, p. 22). One major DELTA-related database model (DELIA, p. 19) may support even more DELTA directives than DiversityDescriptions. The precise extent of DELTA support is difficult to assess, because no documentation of the DELIA data structures has been published (M. Choo, pers. comm.) and the software uses a proprietary or encrypted database format. However, it is clear from available documentation that DELIA has a different approach than that of DiversityDescriptions. DELIA manages DELTA data only on the level of projects, relying on external DELTA programs like CSIRO DELTA or CBIT Lucid for all detailed editing or analytical tasks. Unlike DiversityDescriptions, DELIA does not expose descriptive data as relational tables, e. g., to perform external statistical analysis.

Only few relational databases including support for descriptions could be studied in detail. Of these only Pandora (p. 19) provides DELTA import and export. Unlike DiversityDescriptions, Pandora is a complete integrated database program, addressing issues of nomenclature, literature, and synonymy, as well as descriptions. With regard to the latter, however, the difference between the handling of DELTA by DiversityDescriptions and Pandora is that DiversityDescriptions has a detailed information model corresponding and exceeding DELTA features, whereas Pandora saves character data in a character \times item table as a custom data type using DELTA formatted “text fragments” (thus avoiding the handling of multiple states, statistical measures, free-form text annotations, etc.). As a result, Pandora data can be easily processed by DELTA software, but many questions like statistical analysis require custom-created code for the DELTA data type. Among the non-DELTA enabled systems, TAXIS (p. 21) and AditKey (p. 21) supports comparatively simple character + state model with illustrations (AditKey supports in addition to multi-access keys also authored branching keys, which are not available in DiversityDescriptions).

The development of DeltaAccess/DiversityDescriptions was an ongoing process while this dissertation was prepared and helped to gain a new understanding of the problems involved in structuring descriptive data. Several solutions that improve on DELTA are present in the DiversityDescriptions model:

- **Modifiers** are introduced as a new terminology class. The need for some form of modification was already recognized in the original DELTA publication (Dallwitz 1980), but free-form text comments were considered sufficient. More structured forms have since been proposed (compare p. 192; especially in “New DELTA” as “coded comments” and in CBIT Lucid as flags for rare and misinterpretation). All these approaches are limited to very few, system-defined modifiers. The new approach introduced in Hagedorn (1997) and elaborated in this thesis in the

chapter “Modifiers” (p. 189), is to treat modifiers as a general form of descriptive vocabulary. Similar to categorical states Modifiers form a flexible, user-extensible concept.

- A more flexible system of handling **statistical measures** is introduced. Using a similar approach to modifiers, the small system-defined set of supported measures, widely varying from application to application (see Table 31, p. 112), was replaced by a vocabulary-based, flexible, and extensible system (see p. 356). DiversityDescriptions was the first descriptive tool to provide semantic definition for aggregated statistical value, including expressiveness about the lack of semantic knowledge in legacy data. This fills one of the requirements for a semantic framework for descriptions (Lebbe & Vignes 1998).
- The handling of **multilingual data sets** is supported. In DELTA “shadow copies” of the entire terminology-files had to be maintained (requiring manual synchronization if order was changed or new characters or states were created). As presented in Fig. 216 (p. 112), translations are now restricted to language-sensitive attributes and an unlimited number is supported. Changes in terminology no longer pose a problem.
- DELTA supported **character and item subsets** for report-generation purposes alone. By basing subset and filter features on views that may be used instead of the base tables, these can be used for interactive and collaborative editing (see p. 334).
- Character attributes to create **natural language descriptions** from structured descriptions are improved (see p. 327).
- Metadata for a data set (i. e., “project”) are provided, improving the ability of data exchange without external documentation (see p. 352).

However, many shortcomings of the DiversityDescriptions 1.9 model may also be noted:

- The physical model is already burdened with a number of legacy/backward compatibility artifacts (like duplicate key structures or an overly complex heading hierarchy; compare “Physical model for DiversityDescriptions 1.9”, p. 332)
- No support for sample data (compare “Data recording levels (sample data)”, p. 89).
- Very limited support for ontological concept hierarchies (through “character headings”).
- The support for multiple languages is asymmetric, with a “current language” treated different than the translations.
- The modifier model is limited to a single modifier per statement, frequency and certainty modifiers have no quantitative equivalents, and the linking occurs through natural language strings rather than IDs.
- No true support for interactions between multiple, independent projects (e. g., using a mixture of terminology from project A and B in the descriptions in project C and D).
- The metadata for projects are still relatively poorly structured.
- In version 1.9 the support for machine-readable specimen or publication scope (p. 329) and secondary classifiers (p. 215) is still poor. However, this problem will be addressed in the next release (version 2.0, not documented here).

At some point, a redesigned information model will be required. As long as no substantial funding for this can be achieved, the DiversityDescriptions will be changed evolutionary. The current tie to the Windows operating system has recently been reduced with the release of descriptive data support in DiversityNavigator (Fig. 231).

	50 ascus wall kind	51 ascus wall amyloidity presence	52 ascus tholus amyloidity pattern	53 ascospore number	54 ascospore shape	55 ascospore length	56 asc wid
0487 <i>Placopyrenium heppicoides</i> (Zahlbr.) Breuss	2	1		8.0	11/22	16.0-23.0	
0488 <i>Placynthium asperellum</i> (Ach.) Trevis.	2	1		4.0-8.0	4/26	11.5-21.0	
0489 <i>Placynthium pannariellum</i> (Nyl.) H. Magn.	2	1				15.0-20.0	
0490 <i>Platismatia herrei</i> (Imshaug) W. Culb. & C. Cul...	2	2	5	8.0	4		
0491 <i>Pleurosticta acetabulum</i> (Neck.) Elix & Lumbsch	2	2	5	8.0	4	14.0-17.0	
0492 <i>Polyblastia cruenta</i> (Körb.) P. James & Swinsc...	1	1		8.0	4	40.0-90.0	20
0493 <i>Polyblastia gothica</i> Th. Fr.	1	1		8.0	4	(15.0-)20.... (8.0	
0494 <i>Polyblastia plicata</i> (A. Massal.) Lönnr.	1	1		8.0	4	10.0-12.0	6
0495 <i>Polyblastia verrucosa</i> (Ach.) Lönnr.	1	1		8.0	4/26	30.0-45.0(...	15

Charac...	Modifier	X	Comments
11		0	oblong-ellipsoid
22	narrowly 0		

Sel.	CS	ascospore shape
<input checked="" type="checkbox"/>	11	oblong-obtuse
<input type="checkbox"/>	12	oblong-truncate
<input type="checkbox"/>	13	discoid (in surface view)
<input type="checkbox"/>	14	discoid (in side view)
<input type="checkbox"/>	15	lenticular (in surface view)
<input type="checkbox"/>	16	lenticular (in side view)
<input type="checkbox"/>	17	sigmoid
<input type="checkbox"/>	18	reniform (fabiform)
<input type="checkbox"/>	19	allantoid
<input type="checkbox"/>	20	lunate
<input type="checkbox"/>	21	falcate
<input checked="" type="checkbox"/>	22	ovoid

Figure 231. Java-based DiversityNavigator grid view editor for descriptive data (Neubacher & Rambold 2007b). The underlying information model is DiversityDescriptions 1.9.

SDD

SDD could not be documented in detail here because of limitations of available time and space. Despite that the present thesis evolved in parallel with the development of SDD and both have profited from each other, SDD is not necessarily the solution to the requirements developed here. The design process of SDD was complex, involving many practical considerations and issues of compatibility with existing software and data sets. As a result, while many of the considerations given in the current thesis are informed by SDD discussions and may help to understand decisions taken in the SDD design, many of the more difficult, specialized or little understood issues have not yet been addressed in SDD, or have been addressed in a form that cannot yet be considered satisfactory. For example, a major task in a future redesign will be the seamless integration of further complex data types: molecular sequence and pattern data or quantitative color or shape measurements (especially where it may aid automated identifications, compare p. 231). The present work tries to show that no fundamental distinction exists between conventional and molecular data: both molecular and new morphological methods may have specific data type or management requirements. In SDD, the traditional morphometrically oriented framework of the character was transformed into an extensible character type (being extensible also in the sense of an XML-schema). However, a completely integrated information model is still a task for the future.

SDD is the first model that allows descriptions to be associated with more than one defining classification. It is applicable to both descriptions of taxa or specimen, supports secondary classifier systems (like stage and sex), geographically specialized descriptions, and system-internal (data editor) and external (publication, data source) attribution. It achieves this by rejecting the traditional approach that a description must be nested in a single defining entity. Instead, it treats descriptions as a first-class object that can be associated with all kinds of other objects, defining identity or scope. By the same mechanism, SDD supports part-specific descriptions, i. e., the scope of a description may be only a part of an organism, supporting the specific expressivity of the Prometheus description model in this regard.

The flexibility of this approach can be seen when considering the relation between descriptions and media objects like images, videos, or audio. Traditionally media objects have a voucher

role, supporting descriptive information presented in separate observation or specimen data. With the increasing information density of digital media, media objects are now more frequently the object that is described or identified (see Media data, p. 60). SDD can handle this ambiguity without problems.

SDD provides support for applications that inherit information along the taxonomic hierarchy (p. 99), including character \times taxon specific character guidance information. The latter information is not structured as a modifier (“Reliability modifiers”, p. 213), but directly relates to character variables rather than character data. This solves a long-standing problem that character guidance information can be defined as character metadata (as in DELTA) only for small projects with limited taxonomic diversity (see, e. g., Diederich & Milton 1993a).

Among the requirements for SDD was lossless, fully interoperable data exchange. An example scenario for this is that data editing starts in one application specialized for handling natural language descriptions. All data are then transferred to another application, testing their suitability for interactive identification, and editing erroneous or missing data. When returning to the first application, no information shall have been lost. Some support for this has been built into SDD; the extent to which this can be exploited by applications remains to be answered in the future.

8.2. Results of requirement analyses

The requirement analyses performed in the present work aim to be a structured inventory of fundamental problems encountered when collecting and summarizing scientific descriptions of organisms. Although these analyses try to cover the conceptual space of descriptive data, a moderately even coverage could only be achieved in the use case analysis. This analysis tries to prepare a framework for future specialized and software-engineering-oriented use case analysis that includes detailed analysis. The remaining analyses are strongly influenced by current discussion in the literature and specifically discussions brought up during the development of SDD.

The highly specialized analysis of identification methods was performed because identification is one of the central applications of descriptive data. As a consequence many publications deal with identification methods, and it was not trivial to distinguish between presentational or algorithmic aspects and aspects that need to be reflected in information models. A corresponding analysis for the similarly relevant as well as complex topic of phylogenetic analyses is a task for the future.

The majority of questions have been addressed in the fundamentals chapter. This analysis is guided by a number of “lead questions” resulting from the development of SDD (see p. 17). Some final thoughts or summaries are provided in the following to reflect the major results.

How abstract should be model be?

The required abstraction level is a function of the complexity of the problem domain. In section 4.4 (p. 42) the arguments for developing abstract systems based on data types and high-level semantics are discussed. Given the diversity of biological organisms and methods to study them, and given that software development resources are scarce, a fairly abstract model is generally preferred (as in, e. g., DELTA, NEXUS, or SDD).

Are different models for individual and class descriptions needed?

In principle, descriptive data on individual objects differ from data on a set of individuals (i. e. class). Each individual object must have an unambiguous state or attribute value (which may be unknown, but not variable). No individual can have a component that is “present or absent”, whereas a class can. Unfortunately, this philosophical distinction cannot be directly transferred to individual *organisms*, because:

- Large parts of the description of an individual are truly descriptions of parts of the object, which in turn can be repeated (leaves, flowers, spores, etc.). Thus a tree may have “elliptic or lanceolate leaves” (see “Aggregation within individuals”, p. 93).
- Even a single part can have a property pattern (see section “Pattern versus composition”, p. 165) or gradient (see section “Spatial gradients”, p. 150).
- Temporal (ontogenetic) developments may change the value of properties (e. g., “flowers red, later blue”, see section “Change of object concepts through temporal development”, p. 162). Thus only measurements of individuals at a single point in time, referring to parts that are neither repeated nor involved in patterns or gradient structures, have the unambiguous property values expected.

The Prometheus description model (p. 21) models the first part (multiple object parts) in detail, expressing knowledge on composition as well as forming the basis for all data storage. This could be a model to be extended to include the other sources of variation and polymorphism within individuals. However, the resulting physical model is not fully documented in the publications so far, so that its complexity is difficult to assess. The complexity of a model including all other possible causes for data aggregations within individuals as well need further studies. Furthermore, the Prometheus description model does not support ambiguities and lack of information on data quality, i. e., where only a range of confidence interval is given, but where the basis for this aggregation is not known. Expressing such data is not in the scope of the model, Prometheus being explicitly designed as a model only for new data.

The analysis presented in this thesis shows that the level of the *individual organism* is not identical with the concept of an individual observation. Although models like Prometheus have some unique advantages in data analysis (for example they may constrain that certain parts can only occur a single time), the majority of requirements seem to point towards a model *supporting* a distinction between various levels of data sampling and data aggregation, rather than *requiring* it as an *a-priori* condition before data can be stored. From the stand point of data management and analysis, supporting individual observations (including “Linked observations”, p. 90) is more relevant than a special data structure for individual organisms. The data aggregation methods used within individual organisms and taxa are the same.

From this perspective, the distinction between a description of a class and an individual becomes metadata on the description. For example, the SDD model supports – through description scopes – to inform that a given description is the exact description of one individual specimen. However, the same scope mechanism may also be used to specify that the scope is a particular part (e. g., a single leaf) of that specimen, or that the scope is the set of all “specimens studied”, listing each specimen.

Which data types are required?

The analyses suggest that the categorical and quantitative data types are more fundamental than the distinction between data on the nominal or ordinal, and interval or ratio, respectively. Furthermore, except for data relating to counts, all so-called “qualitative data” seem to be not “essentially different”, but are categorizations of phenomena that may also be measured quantitatively. For this reason, the frequently used term “qualitative data” is rejected, and “categorical data” – implying a perspective of categorization – is preferred. With this perspective, the analysis of the underlying “value space” that results in a categorization is studied, and an attempt for new concepts to express the “Singularity, extension and connectedness of categories” (p. 53) is made.

The importance of free-form text (p. 56) is discussed. Unconstrained narrative text, if the only form of expression (as opposed to an annotation or extension of other data types), is modeled as a separate data type.

In the future, additional “Complex quantitative data types” (p. 59) need to be introduced for digital recorded data. In the light of the growing importance of molecular data in recent years, perhaps the most urgent question of extending data types is the question how to fully integrate

descriptive molecular data with biochemical, anatomical, morphological data. Although some thoughts for a subset of these data (sequence data) are collected (“Molecular sequence data”, p. 57 and “Special aggregation cases”, p. 92), a deeper analysis of this problem was deliberately postponed and is a major task for future studies.

The present work primarily establishes that no principal distinctions between conventional and molecular data exist, and that similar differences between data types, aggregation methods, and analysis and comparison methods exist for new forms of morphological data, such as digitally recorded color or shape measurements.

It may be interesting to compare the kind of data types discussed here with those available in standard object-oriented programming languages (C++, Java, .NET languages, Eiffel, etc.). These are based on two primary concepts:

- A generalization hierarchy (kind-of relations) provided through class inheritance (associated with concepts like polymorphism). In principle, this hierarchy may be a tree (single inheritance languages like Java or .NET languages) or a directed acyclic graph (multiple inheritance languages like C++ or Eiffel). The picture is slightly complicated by the fact that most single-inheritance languages support *interface classes*. Although these do not enable inheritance of methods or private data, they do support the polymorphic use of classes. One may say that interface classes provide a secondary generalization hierarchy.
- An object composition model (part-of relations) provided in the model through variables typed to another class. No limitations are imposed on these relations (they may be cyclic).

In modern languages, the type system itself is part of the generalization hierarchy and may make use of compositions (in collections, structures, etc.). The distinction between “data types” (p. 49) and “properties” (as in character decomposition models) that was made in the discussion so far has an analog in programming languages in the distinction between built-in data types (often with special properties) and user-defined types (any classes). Although recently modern OOP languages try to reduce the visibility of these difference (by using similar language constructs to address either built-in or user-defined data types and classes), the reason for this distinction remains valid and is the same for descriptive information models: More fundamental data type allow applications to rely on their behavior, thus opening opportunities to optimize the handling of these. The basic types defined for descriptive data embed the essential information a processor needs during statistical or phylogenetic analyses or identification. The semantic interpretation of the domain-specific terminology can be built on top of this.

Interestingly, there is no equivalent in programming languages for the “develops-from” relation (ontogenetic or phylogenetic) commonly found in biology (see “Change of object concepts through temporal development”, p. 162).

Use a character matrix or character state matrix model?

In “Categorical data: Character matrix vs. character state matrix” (p. 104) the fundamental convertibility of the character matrix, character state matrix, and the list model is shown. The true difference is what kind of information is recorded at which level. The existing character state matrix model in Lucid needs some special rules to deal with unknown character information, and as a result a general coding status model (p. 74) is more difficult to introduce in this model. Conversely, negative information of the form “the state has been considered, but did not occur in the entity” is recorded currently only in the character state matrix, but not in existing character matrix and character list models. With a static terminology, this is indeed redundant, but with evolving terminologies, and perhaps initially imperfectly defined characters, this redundant information can be valuable.

How to handle quantitative values and statistical measures?

The examples given in Table 31 (p. 112) clearly show that the selection of a small set of statistical measures is somewhat arbitrary or at best strongly depends on the specific purpose of the model. Using a rich vocabulary for univariate statistical measures (see “Quantitative data and statistical measures”, p. 110, and “Standard aggregation methods”, p. 85), introduced by Diversity-Descriptions and elaborated in SDD, seems to be a powerful and flexible method that can easily be implemented in existing database software.

How to handle original measurements and sample data?

A major result of the requirement analysis is that although the distinction between intra-individual and inter-individual aggregation methods is weak, the handling of sample data (p. 89) does require special attention. In the case of quantitative data a specific form of the character data type is required (storing only single measurements without support for statistical measures). Furthermore, sampling events (combined multiple measurements) have metadata and it is desirable to preserve original recording order. Within a sampling event, linked observations on multiple characters (p. 90) may occur, requiring another structural level of relating descriptive data.

How to handle the relation between broad and narrow concepts of character states?

The relation between broad and narrow concepts may be interpreted as a tree-like hierarchy or as a directed acyclic graph. The latter enables narrow categorical values to be generalizable to more than one broader concept. In the chapter “Mappings within categorical data” (p. 68) it is argued that the latter is desirable. It supports varying analysis perspectives, error tolerance during identification, and does not cause an undue burden to the user interface (in current descriptive data applications it suffices to display the DAG as a simple tree with duplicated branches). Furthermore, it can be shown that the concept of relations between categorical states can be viewed as a special application of the more general “mapping” between data representations of the same or different data types.

How to handle character dependency (and whether two complementary mechanisms (applicable-if, inapplicable-if) are necessary or desirable)?

The analysis of character dependency and current character applicability models (p. 76) could resolve the seeming discrepancy between the desire to provide two alternative forms of applicability (applicable-if and inapplicable-if) and the fact that these are indeed convertible (as re-analyzed in “Analysis of convertibility”, p. 80). The central points are the question of communication with human users for revisions and error checking, and the aspect of evolving terminologies to which new character states may be added after the original character applicability rules were created.

Although highly general character dependency models may be elaborated, capturing quantitative character-value correlations as well as total inapplicability, such deviations from the assumptions of character independence are usually analyzable from description and can only rarely be used to improve data integrity. The most immediate need for a future improvement would be, within the framework of character applicability rules, to extend the current models to provide for quantitative characters as controlling characters. This would often allow using a single character for the multiplicity of a structure (“number of part”) to also control presence of part. Current data sets often require redundancy if they intend to use inapplicability rules. SDD does not support this yet.

Which DELTA features can be omitted or generalized?

A number of DELTA and New DELTA features can be generalized if data inheritance along the lines of the taxonomic hierarchy is generally supported. The present analysis revealed that DELTA supports two mechanisms (default or “implicit states”, p. 102 and “variant item”, p. 101) that can be interpreted as a top-level and bottom-level inheritance mechanism and become redundant with the introduction of general data inheritance.

A related example is the replacement of the top-level-only character-ranking metadata (DELTA “reliability” and “weight” directives) with a mechanism supporting inheritance (independently of character values, see above).

Completely independently, a specific SDD decision was made to initially support no “item abundance”, i. e., information how abundant an item is in a given region. Such information can easily be stored in a normal character. The reason why item abundance (p. 276) is specially modeled in DELTA is that it used as a weighting parameter for creating identification keys similar to character-ranking metadata (see Authored character guidance”, p. 267). However, this weighting is not necessarily related to abundance and further studies are necessary whether it is desirable at all.

Should the traditional character concept (employed, e. g., in DELTA and NEXUS) be followed or should a “character decomposition model” be embraced?

One of the most difficult analyses is perhaps the study of “character decomposition” (e. g., the Nemisys/Genisys, p. 21, or Prometheus description model, p. 21, requiring separate object composition and property/property states generalization hierarchies). These models promise to embed ontological knowledge into the data storage itself. However, as analyzed in the section “Descriptive ontologies” (p. 131), a number of problems exist with applying ontologies to descriptive data. The problems with distributing information on part multiplicity between ontological and actual knowledge can probably be solved. In the Prometheus model the problem was studied extensively, but the solution is difficult to fully understand from the information published so far.

A major problem with the earlier Nemisys/Genisys was the limitation to the concept of “basic properties” which were understood as fundamental and not extensible data types. Even for morphological data these types show some degree of inconsistency (p. 62) and adopting this strategy would have led to information models incapable of handling taxonomic groups where non-morphological data are essential. The Prometheus model has already generalized this concept and is more flexible in this regard. The original “basic properties” can still be seen as useful and pragmatic categorizations, helping to organize or structure morpho-anatomical descriptive characters. They should not be used for system building, however.

One of the major problems with all decomposition models is perhaps the “oscillation” of terminology between property and object composition concepts. Many observable features can alternatively be interpreted as belonging to the structural or property domain (patterns, surface roughness, and pigment colors). Another problem that seems to be insufficiently understood are the taxon-specific specializations of ontological concepts and how to deal with them.

Furthermore, the problems of recognizing parts in composition through properties and then describing them through properties lead to difficulties. The philosophical problem behind this can perhaps be studied in (a) “scales”: “color: green or white”, “thickness: 1 mm” *versus* (b) “white scales”: “thickness: 1 mm”.

Despite the complexity of models such as Prometheus, these models capture only a portion of the full dimensionality of descriptive data in biology. Prometheus records the spatial composition of an individual at a single point in time with great detail, but can neither deal with the temporal composition of individuals over their life span, nor with the variety of observation and measurement methods required for many taxonomic groups. Interestingly, Germeier & Frese (2001), coming from plant genetic resources have a completely different view on character decomposition than Nemisys/Genisys or Prometheus. Whereas the latter combine property and methodolo-

gy into a single entity, “decomposing” only the part information, Germeier & Frese (2001) keep property and part information combined, but desire to separate out the dimension of methodology.

Even within a single ontology, many aspects of it are strongly guided by its design purpose. As mentioned in the main text, the recently released plantontology.org-ontologies (Pujar & al. 2006, Ilic & al. 2006) are optimized for genetic studies and, as the authors acknowledge themselves, difficult to use or even unusable for descriptive data. Similarly, conflicting ontologies may arise for the purpose of phylogenetic analysis (where all concepts shall fully satisfy homology assumptions) and identification (where “operational homology” definitions that are recognizable during the identification process are required).

Character models which combining object part, property, and observation methods (including conditions and instrumentation) may be more adaptable to different interpretation as to whether a measurable concept shall be considered a structural part or a property, and under which purpose-guided ontology the data are to be interpreted. The section “Concept hierarchies” (p. 125) introduces the alternative “concept hierarchy” proposal and compares it with character decomposition models. The models primarily differ with respect to whether ontologies directly form the basis of data storage, or whether an intermediate, operational concept called “character” (or perhaps “feature”) should be present, to decouple knowledge representation and storage from analysis, interpretation, and evolving concept hierarchies (semantic ontologies).

Using concept hierarchies in combination with characters, multiple competing hierarchical classifications of atomic characters are possible. This avoids arbitrary decisions of placing characters in a required hierarchy. In the “feature path” in Prometheus a feature can only be found if the path defined by the classification system used is known. It may be desirable to use a classification system directly during identification, e. g., to ask for the size and shape of *any* spore if the user cannot be expected to be able to differentiate spore generations.

The currently best solution proposed in this analysis is the character model, combined with mechanisms to optionally analyze (and thus annotate) it as compositional and property hierarchies. This model has been pursued in the SDD process. It allows alternative hierarchical views on the list of characters defined, but does not force decisions in cases where alternative concepts apply equally well.

If the ongoing development of ontological techniques in information science allows for generalized rather than purpose-built ontologies, and if agreed and stable terminologies for at least spatial and temporal compositions, properties, and methodology can be created in the coming years, a more general form of “character decomposition model” based on object part, property, method, and point in time may be desirable.

How to federate and modularize terminology as well as descriptions?

The discussion of federation and modularization aspects of descriptive terminology and data (p. 180) can only be considered a first start. With increasing of use GUIDs and semantic web technologies, new approaches are likely to emerge. When evaluating such technologies, however, care should be taken to support the flexibility of local knowledge production of single research scientists. By making complex and time consuming management procedures obligatory, a danger exists that knowledge can only be adequately expressed through large and powerful institutions. This may be detrimental to science in general, and the typically underfunded biodiversity research in particular.

How can the terminology be kept concise, while supporting structured extensions to the expressibility? Can “modifiers” contribute to this?

Describing bio-*diversity* in a structured way requires a correspondingly diverse vocabulary of recordable concepts, observation, instrumentation, experimentation, and measurement methods. Managing this descriptive diversity is not trivial. A major point is the need to keep the tool-speci-

fic learning curve relatively shallow for the users of the system. Existing systems are a major source for previous user experiences. DELTA and DELTA-like systems show that for moderately complex groups a large number of characters are required and that it is difficult to work with systems with several hundred characters.

Both “character decomposition models” (Nemisys/Genisys, Prometheus) and the “concept hierarchies” (proposed here and in SDD) are an attempt to improve the manageability of characters, by supporting knowledge on generalization and composition to be usable to edit and query descriptive information.

On the other hand, many existing systems already recognize the need to extend the expressivity offered by a fundamental variable + value structure with unconstrained (free-form, natural language) text. Some formats designed for special analytical purposes (NEXUS for phylogenetic analysis, Lucid for identification data) omit this extensibility, but these do not aim to be general and primary knowledge repositories.

As shown in the chapter “Modifiers” (p. 189), much information presented in these free-form text extensions actually comes from a constrained vocabulary and is analytically useful. A general modifier mechanism providing a flexible and user-extensible vocabulary was first introduced in DiversityDescriptions in 1997. It followed earlier proposals of related mechanisms in Lucid and New Delta, which were limited to small sets of system-defined values. The topic of modifiers is studied in detail here.

Modifiers can be seen as metadata on values and are related to the concept of modal logic in philosophy.

How to handle secondary classifiers like sex or life cycle stages?

The analysis of “Secondary classification resulting in description scopes” (p. 215) remains somewhat inconclusive. It establishes that the problem is not adequately solved by the creation of “pseudo-taxa” and that the number of potential secondary classifiers is more diverse than may initially be expected. As a result, it is clear that a general solution is desirable rather than creating special concepts for sex or stage. However, the exact form requires further analysis. The SDD data exchange standard does make the decision to consider primary classifiers, indication of information source, geographical scope, and secondary classifiers on a single level, all being generalized to be the scope of a description. Whether this is ultimately satisfactory will be shown as experience with the application of SDD grows.

Topics that require further analysis

In the fundamental analysis some topics are not addressed with the necessary detail. In addition to phylogenetic analyses already mentioned above, this involves in particular

- the creation of descriptive data in the context of “online monographs”,
- the creation of natural language descriptions from structured coded descriptions,
- the markup of legacy natural language descriptions to improve comparability and searchability, and
- aspects of multilingual data sets.

Some information on these topics are discussed on a more general level in the use case analysis, see “Taxon pages” (p. 319), “Conversion of digitized data to coded descriptions” (p. 294; plus “Natural language generation” on p. 327 in the “Logical model for DiversityDescriptions 1.9”), “Digitization and markup of descriptions” (p. 293), and “Multiple languages or audiences” (p. 282). See also the “Open aspects” (p. 322) of the use case analysis.

8.3. Description logic and unified systems

The present work discusses approaches to description rooted in biological tradition, computer and information science, statistical analysis, and phylogenetic data analysis. Although some aspects

of philosophy, logic, and mathematics are indirectly used insofar as they are embedded in current software modeling and implementation approaches, a more fundamentally philosophical approach (based in logic, analytical philosophy, linguistic philosophy, etc.) is not presented. The focus of this work is on solutions that can be implemented and managed using current software and available resources.

However, a more rigorous philosophical framework (e. g., description logic) may offer additional insight into some of the problems discussed. Fundamental statements like:

- The window of the house is green,
- The round window of the house is green,
- The dog is barking,
- The black dog is barking,
- The dog is probably frequently loudly barking,

correspond closely to some of the problems discussed (descriptions as properties of object parts, which themselves are defined through properties and parts, Figs. 8-9, p. 37, problem of secondary classifiers defining description scope, “Secondary classification resulting in description scopes”, p. 215, “Modifiers”, p. 189). In this view a close relationship exists between properties, decomposition into parts, description scope, and taxonomic hierarchy. From a biological viewpoint the taxonomic hierarchy, because of evolution and phylogenetic history, is an essential and independent concept in descriptions. For the methodology of description and identification, however, it may lead to more consistent models to view it as accidental. In such a model, identifiable objects may be parts of organisms, entire organisms, symbiotic associations, or even loose communities. Descriptive knowledge of the world would be build incrementally by defining identifiable parts that are then reused as properties and parts of compositions.

Two major problems are immediately recognizable. One is that rather than building analytically from parts, the human brain usually recognizes patterns on a high level. For example, one may define a “table” as “three or more legs, flat surface”. A table with a single central leg, mounted to a wall lacking legs, or a compact cube would be accepted as well. Perhaps a table is a stable, level flat surface raised above ground level. The example shows that normal recognition does not proceed analytically, but rather as an instant recognition of a “table” pattern, from which special cases are derived. A pattern-based approach has been formulated by Fortuner (1989b) for the purpose of biological identification in the “promorph” proposal (see p. 238). Promorphs can be understood as “corner stones” in the identification process, perhaps as patterns that are recommendable to learn by heart. Without calling them such, they are often embedded in conventional keys as the points where an initial key refers to separate subkeys. Often these points will coincide with major taxonomic ranks, which conventionally are placed at such intuitively recognizable entities. It is, however, unclear how to either combine or replace the approaches (DELTA-like character/state models, p. 104, character decomposition models, p. 116) discussed in this these with a “promorph”-system; this may be a fruitful area for future studies.

The second major problem is lack and uncertainty of knowledge in biological descriptive data. This makes a consistently incremental knowledge system difficult. The fact that the concepts used as building blocks in conventional descriptions are often only loosely defined, and interpreted in the knowledge of uncertainty has its roots in the incomplete knowledge of biological species and character diversity. This makes it difficult to agree on an “axiomatic” system of building blocks, that could be considered fixed and the “atomic” basis for comparison. However, changing these building blocks may then invalidate the system. This problem is comparable to the problems of schema evolution (e. g., “Static versus dynamic terminology models”, p. 45; translation issues, p. 71. If a model no longer makes a distinction between terminological conventions and descriptive data based on these conventions, controlling these issues may become considerably more difficult. An essential requirement of a new system would be its ability to deal with change, error, and insufficient knowledge. The inability of managing these problems is probably the major reason that attempts to create rule-based “expert systems” after initially promising results failed to serve through continuous improvements and corrections.

8.4. Future relevance:

A proposal to record identification data

As a final outlook to the relevance of descriptive data a proposal may be made to document organism identifications in biodiversity research. A first step might be that where specimens in natural history collections are identified using computer-aided methods, the identification steps (using any combination of multi-access keys and branching keys) could be converted to basic descriptions and permanently stored as descriptive data. Not only would this automatically build a wealth of descriptive data for general use without requiring additional investments from the taxonomists doing the revisions. When the specimen is next used in a taxon revision (and perhaps re-identified), these data could be used to confirm the identification or they could help to understand what taxonomic concept was used in the original naming process. This process is not used so far. With the progress made in recent years in designing collection and description information models that are intended specifically to interact with each other, the scenario may, however, soon become reality.

The same principle could furthermore be applied to all data recording or publishing activities that involve identifications of organisms, in taxonomy as well as applied fields like ecology, pathology, medicine, biotechnology, etc. The deposition of voucher specimens for the organism identified should clearly be called for (Agerer & al. 2000) wherever possible, but due to lack of time and resources not doing so is the dominant practice in most areas of applied science dealing with biodiversity. As a result, published property data (molecular, enzymatic, morphological, etc.), geographical distribution data, or organism interactions (predator-prey, host-pathogen, pollinator, etc.) are often not comparable because different taxonomic concepts have been used.

Assessing these concepts is extremely difficult. The Prometheus taxonomic model (“Prometheus I”, Pullan & al. 2000, Raguenaud & al. 2002) defines a taxon concept (i. e. taxon circumscription) as an enumeration of preserved specimens. This makes it applicable to taxonomic revisions, but not to applied biodiversity research, which contributes a large amount of biodiversity information.

Another approach would be to cite the primary identification literature used to reach an identification result (e. g., a flora, fauna, or a specialized monograph) together with each taxon name. The currently dominant practice of citing the taxonomic authority (i. e., the team of authors responsible for the nomenclatural priority of a name) essentially is an indirect citation of the type specimen. However, usually only taxonomic revisions directly study type specimens. Most information on organisms has to rely on published descriptions to compare and identify organisms with taxon concepts. In these cases citing a concept reference would be far more relevant, especially since the taxonomic authority can usually be retrieved from the concept reference. Thus, for an ecological or geobotanical study in Germany, it is considerably more informative to cite “*Thymus serpyllum* sec. Rothmaler & al. 1985”, “*Thymus serpyllum* sec. Schmeil & Fitschen 1988”, or “*Thymus serpyllum* sec. Schmeil & al. 2006” – highlighting the strongly different circumscriptions – rather than citing “*Thymus serpyllum* L.”

Furthermore, although homonyms are a principal problem in taxonomic research, in the practice of applied biodiversity research they occur much more rarely. For example, in Germany three major identification works for higher plants are commonly used (i. e., various editions of Oberdorfer 1983, Rothmaler & al. 1985, Schmeil & Fitschen 1988). Albeit occasionally differing in their choice of accepted (or “correct”) name and frequently in their circumscription of taxa (taxon concepts), the present author is not aware of a single case where they accept different homonyms. In taxonomic groups where homonymy is more frequent, this is usually a result of a lack of taxonomic revisions and, consequently, it is rare that competing current identification literature exists at all. Consequently, citing a concept reference instead of a taxonomic reference requires the consumer of the information only very rarely to remember taxonomic differences between names with different authors.

A few software applications at least support recording the identification literature used. Examples are Recorder 2000, a commercial software used in British nature surveys (<http://www.jncc.gov.uk/species/Recorder2000/default.htm>), or DiversityCollection (Hagedorn & Weiss 2002). However, no real tradition of citing identification concept references could so far be established in biology. As a consequence, most taxon names in the literature have a taxon concept which can be recorded (using the reference of the publication in which it appears), but cannot be operationalized since the taxon circumscription and relations to other taxon concepts are completely unknown. The assumption that the citation of a taxonomic name together with the concept reference is sufficient to manage taxon concepts (see, e. g., Yoon & Rose 2001 and Ytow & al. 2001) is erroneous, if no comparison operations can be established. Comparison operations have been analyzed, e. g., in the MoreTax project (Berendsohn & al. 2003, Geoffroy 2003, Geoffroy & Berendsohn 2003), but the practical problem of retrospective interpretation of taxon concepts (Kennedy 2003) and mapping these concepts to each other is unsolved and most likely unsolvable.

However, even if a practice of citing identification references as concept references could be established, this would provide only the necessary precondition to establish concept comparisons. The problem of finding the resources to actually define the comparisons would remain. It is unlikely that the diagnostic taxon descriptions of more than the most important and frequently used diagnostic publications will ever be analyzed. This process first requires to interpret the natural language descriptions (which are usually incomplete because mainly diagnostic characters are mentioned) in terms of a coded terminology. Since it is unlikely that two descriptions match totally (they may consider different characters diagnostic), the process then requires an assessment how close a match between taxon concepts must be, to consider them still identical for the purpose of analysis.

This process of identification, publication, and later “reverse engineering” the taxon concept and circumstances of identification could be greatly simplified. If identification would already be performed using a computerized process, the identification data could be stored in some form of “IdentificationBank” (or “ID-Base”) and a citable accession code provided (see also the use case diagram Fig. 177, p. 296).

IdentifyLife (IdentifyLife 2005) could, in addition to being a repository of identification key data, become such an IdentificationBank.

Good scientific practice (and journal editors) might then require the publication of an identification concept IDs, similar to the current requirement for accession codes for the (descriptive!) nucleotide sequences submitted to GenBank, EMBL, or DDBJ. The identification concept ID would then allow the retrieval of a documentation of the identification process, including both the coded terminology IDs and the label representations in the language that was used. The latter (and free-form text and note information viewed during identification) would be frozen to the state at the time of identification. Furthermore, a documentation of the images or other media resources used during documentation may also be valuable. In contrast to citing a published reference, an identification concept would provide not the full description, but only the parts that were actually verified during identification.

In any case, such a documentation of identification concepts will allow a substantially improved understanding of the relations between the taxon concepts in independent identifications. Whether the analysis and comparison of concepts requires machines or humans depends on the amount of coded terminology used (as opposed to free-form text) and whether a shared terminology was used.

For taxa where the majority of characteristics can be studied in a voucher specimen (e. g., in plants) this may still be inferior to a voucher specimen. However, the new procedure would not necessarily replace voucher specimens. It would fill the gap where currently no information is available at all. Furthermore, in taxa that are difficult to preserve or where essential characteristics can only be studied in culture, this may be more valuable than preserving a dead voucher specimen.

Although such use may currently be unrealistic in field studies, digital technology is progressing fast, and in laboratory studies the use of digital identification and documentation tools would already save valuable time for the researcher. An important feature in this context is the storage of original descriptive observation data like individual measurements. These data should be stored directly and the synthesized statistical summary information should be automatically created from them.

9. References

- Adit 2004. AditKey. A species identification and key construction system. <http://www.adit.co.uk/html/aditkey.html>. [Last modified 2004-09-26, last retrieved 2007-04-16]
- Agarwal, Gaurav; Ling, Haibin; Jacobs, David; Shirdhonkar, Sameer; Kress, W. John; Russell, Rusty; Belhumeur, Peter; Dixit, Nandan; Feiner, Steve; Mahajan, Dhruv; Sunkavalli, Kalyan; Ramamoorthi, Ravi & White, Sean 2006. First steps toward an electronic field guide for plants. *Taxon* **55** (3): 597-610. [Preprint at: http://herbarium.cs.columbia.edu/pubs/First_Steps_Toward_an_Electronic_Field_Guide_for_Plants.pdf, last retrieved 2007-04-20]
- Agerer, Reinhard; Ammirati, Joe; Blanz, Paul; Courtecuisse, Régis; Desjardin, Dennis E.; Gams, Walter; Hallenberg, Nils; Halling, Roy; Hawksworth, David L.; Horak, Egon; Korf, Richard P.; Mueller, Greg M.; Oberwinkler, Franz; Rambold, Gerhard; Summerbell, Richard C.; Triebel, Dagmar & Watling, Roy 2000. Always deposit vouchers. *Mycological Research* **104**: 641-644; *Nova Hedwigia* **71** (3-4): 539-543; *Mycorrhiza* **10** (2): 95-97.
- Alexander, Greg 2006a. SLIKS-Alike Interactive Key Software (SAIKS). <http://galexander.org/saiks>. [Software version 2.1; page last modified 2006-11-16; last retrieved 2007-04-10]
- Alexander, P. J. 2006b. Key to the *Boechea* of New Mexico. <http://boechea.com/keys/nm/boechea.html>. [Last retrieved 2007-04-10]
- Alexander, P. 2006c. Organ Mountains Flora. <http://organmountainsflora.com>. [Last retrieved 2007-04-10]
- Allkin, R. 1989a. Introduction to ALICE. *DELTA Newsletter* **3**: 1-4.
- Allkin, R. 1989b. ALICE and DELTA: where's the link? *DELTA Newsletter* **4**: 1-3.
- Allkin, R. 1996. The ALICE database system: recent news. *DELTA Newsletter* **12**: 21-23.
- Allkin, R. & Bisby, Frank A. 1988. The structure of monographic databases. *Taxon* **37** (3): 756-763.
- Allkin, R. & White, R. J. 1988. Data management models for biological classification. *In*: Bock, H. H. (ed.), *Classification and related methods of data analysis*. Elsevier: Amsterdam (NL), 653-660.
- Ambler, Scott W. 2002. *Agile modeling: Effective practices for extreme programming and the unified process*. John Wiley: New York (USA).
- Ambler, Scott W. 2003a. *The elements of UML style*. Cambridge University Press: Cambridge (UK).
- Ambler, Scott W. 2003b. *Evolutionary database development*. <http://www.agiledata.org/essays/evolutionaryDevelopment.html>. [Last retrieved 2007-02-06]
- Anagnostopoulos, I.; Anagnostopoulos, C.; Vergados, D.; Loumos, V. & Kayafas, E. 2003. A probabilistic neural network for human face identification based on fuzzy logic chromatic rules. *MED'03 — The 11th Mediterranean Conference on Control and Automation*. <http://www.medialab.ece.ntua.gr/medialab/Papers2003/2003-8/8.pdf>. [Last retrieved 2005-05-25; no longer available as of 2007-04-14]
- Anonymous 1996. Welcome to PollyClave – a multiple-entry identification key. <http://prod.library.utoronto.ca:8090/polyclave>. [Page last modified 1996-10-10, last retrieved 2007-04-14; linked identification interfaces show software version 1.05, dated 1997-05-22.]
- Ax, P. 1984. *Das phylogenetische System*. Gustav Fischer, Stuttgart.
- Ballew, Sharon & Pickering, John 2003. 20q XML tags & file structure. http://www.discoverlife.org/ed/tg/Building_Web_Pages/20q_xml_tags.html. [Last retrieved 2007-05-10]
- Baral, H.-O. 1992. Vital versus herbarium taxonomy: morphological differences between living and dead cells of Ascomycetes, and their taxonomic implications. *Mycotaxon* **44** (2): 333-390.
- Bartley, Michael & Cross, Noel 1999. NaviKey 2.0. <http://www.huh.harvard.edu/databases/legacy/navikey>. [Page last retrieved 2004-02-10, no longer accessible as of 2007-04-28; Google reports <http://blodeuwedd.huh.harvard.edu/databases/>

- legacy/navikey, last visited by Google 2005-11-13, again no longer accessible as of 2007-04-28]
- Baum, B. R. 1988. A simple procedure for establishing discrete characters from measurement data, applicable to cladistics. *Taxon* **37**: 63-70.
- Beck, Kent 2000. *Extreme programming explained: embrace change*. Addison-Wesley: Boston (USA).
- Berendsohn, W. G. & Geoffroy, M. 2007. Networking taxonomic concepts – uniting without “unitary-ism”. *In*: Curry, G. & Humphries, C. (eds.), *Biodiversity databases: from cottage industry to industrial networks*. Taylor & Francis: Boca Raton, Florida (USA).
- Berendsohn, W. G. 1995. The concept of “potential taxa” in databases. *Taxon* **44**: 207-212.
- Berendsohn, W. G. 1997. A taxonomic information model for botanical databases: The IOPI model. *Taxon* **46**: 283-309.
- Berendsohn, W. G. 2001a. “Biodiversity informatics”, the term. <http://www.bgbm.org/BioDivInf/TheTerm.htm>. [Page last modified 07-04-2001, last retrieved 2005-02-25]
- Berendsohn, W. G. 2001b. *Biodiversity Informatics*. Preprint of an article to be published in the *Proceedings of the Second National Colloquium on Global Change Research*, Bad Honnef, Jan. 26.-27., 2001. <http://www.bgbm.org/BioDivInf/default.htm>. [Page last modified 2001-08-14, last retrieved 2007-04-14]
- Berendsohn, W. G. 2005 (ed.). *Task Group on Access to Biological Collection Data (ABCD) – a joint CODATA and TDWG initiative supported by GBIF and BioCASE*. <http://www.bgbm.org/TDWG/CODATA/default.htm>. [Last edited 2005-03-06, last retrieved 2007-04-28]
- Berendsohn, W. G.; Anagnostopoulos, A.; Hagedorn, Gregor; Jakupovic, J.; Nimis, P. L. & Valdés, B. 1996a. The CDEFD information model for biological collections. (HTML and WinWord 6 format). <http://www.bgbm.fu-berlin.de/CDEFD/CollectionModel/cdefd.htm>. [Last retrieved 2007-04-28]
- Berendsohn, W. G.; Anagnostopoulos, A.; Jakupovic, J.; Nimis, P. L. & Valdés, B. 1996b. A framework for biological information models. *In*: Valdés, B. & Silvestre, S. (eds.), *Proceedings of the VIII OPTIMA meeting*. Lagascalia **19**: 667-672. Online: <http://www.bgbm.fu-berlin.de/cdefd/objects>. [Last retrieved 2007-04-28]
- Berendsohn, W. G.; Anagnostopoulos, A.; Hagedorn, Gregor; Jakupovic, J.; Nimis, P. L.; Valdés, B.; Güntsch, A.; Pankhurst, R. J. & White, R. J. 1999. A comprehensive reference model for biological collections and surveys. *Taxon* **48**: 511-562. Online: <http://www.bgbm.fu-berlin.de/biodivinf/docs/CollectionModel/ReprintTNR.pdf>. [Last retrieved 2007-04-28]
- Berendsohn, W. G.; Döring, M.; Geoffroy, M.; Glück, K.; Güntsch, A.; Hahn, A.; Kusber, W.-H.; Li, J.; Röpert, D. & Specht, F. 2003. The Berlin model: a concept-based taxonomic information model. *Schriftenreihe Vegetationskunde* **39**: 15-42.
- Berglund, Anders (ed.) 2006. *Extensible Stylesheet Language (XSL) Version 1.1. W3C Recommendation 05 December 2006*. <http://www.w3.org/TR/2006/REC-xsl11-20061205>. [Last retrieved 2007-04-01]
- Berners-Lee, Tim; Hendler, James & Lassila, Ora 2001. *The Semantic Web*. *Scientific American* May **2001**: 29-37. Also: <http://www.sciam.com/article.cfm?articleID=00048144-10D2-1C70-84A9809EC588EF21>. [Last retrieved 2007-05-09]
- BG-BASE Inc. 2007. *BG-BASE™ collection management software*. <http://rbg-web2.rbge.org.uk/bg-base>. [Last modified 2007-03-04, last retrieved 2007-05-10]
- BioAware 2007. *BioloMICS: biological data manager for identification classification & statistics*. <http://www.bio-aware.com/BioloMICS.aspx>. [Last retrieved 2007-05-10]
- Biron, Paul V. & Malhotra, Ashok (eds.) 2001. *W3C XML Schema Part 2: Datatypes. W3C Recommendation, 2 May 2001*. <http://www.w3.org/TR/2001/REC-xmlschema-2-20010502>. [Last retrieved 2007-04-16]
- Blackman, R. L.; Eastop, V. F. & Kibby, G. G. 1997. *Taxakey: Aphids on the world's crops*. CD-ROM. CAB International & Natural History Museum. ISBN 0 85199 172 6.
- Böhmer, Bernd & Wohanka, Walter 2002. *Die Pflanzenschutz-CD. Etwa 600 Farbfotos*. Eugen Ulmer Verlag: Stuttgart (Germany).
- Boos, Evelyn 1992. *Botanische Klassifikation und Taxonomie – Konzeption und Realisierung eines Informationssystems*. Diss. Univ. Ulm, Fakultät für Mathematik und Wirtschaftswissenschaften.
- Brach, Anthony R. & Hong Song 2005. *ActKey: a Web-based interactive identification key program*. *Taxon* **54**: 1041-1046. http://flora.huh.harvard.edu/china/PDF/misc/ActKey_Taxon_54_1041-1046_2005.pdf. [Last retrieved 2007-04-28]
- Brach, Anthony R. & Hong Song 2006. *eFloras: New directions for online floras exemplified by the Flora of China project*. *Taxon* **55**: 188-192. http://flora.huh.harvard.edu/china/PDF/misc/eFloras_Taxon_55_188-192_2006.pdf. [Last retrieved 2007-04-28]

- Bray, Tim; Paoli, Jean & Sperberg-McQueen, C. M. (eds.) 1998. W3C Extensible Markup Language (XML) 1.0. W3C Recommendation 10-February-1998. <http://www.w3.org/TR/1998/REC-xml-19980210>. [Last retrieved 2007-04-01]
- Bray, Tim; Paoli, Jean; Sperberg-McQueen, C. M.; Maler, Eve & Yergeau, François (eds.) 2004a. W3C Extensible Markup Language (XML) 1.0 (Third Edition). W3C Recommendation 04 February 2004. <http://www.w3.org/TR/2004/REC-xml-20040204>. [Last retrieved 2007-04-01]
- Bray, Tim; Paoli, Jean; Sperberg-McQueen, C. M.; Maler, Eve; Yergeau, François & Cowan, John (eds.) 2004b. W3C Extensible Markup Language (XML) 1.1. W3C Recommendation 04 February 2004. <http://www.w3.org/TR/2004/REC-xml11-20040204>. [Last retrieved 2007-04-01]
- Brickley, Dan & Guha, R. V. (eds.) 2004. W3C RDF Vocabulary Description Language 1.0: RDF Schema. W3C Recommendation 10 February 2004. <http://www.w3.org/TR/2004/REC-rdf-schema-20040210>. [Last retrieved 2007-04-01]
- Bruegge, Bernd & Dutoit, Allen H. 2004. Object-oriented software engineering: Using UML, patterns and Java. 2nd edition. Prentice-Hall: New Jersey (USA).
- Cannon, A. & McDonald, Sarah M. 2001. Prometheus II – Qualitative Research Case Study. Capturing and relating character concepts in plant taxonomy. http://www.dcs.napier.ac.uk/~prometheus/prometheus_2/Resources/Qualitative%20Research%20Report.pdf. [Last retrieved 2007-04-28]
- Carlisle, David; Ion, Patrick; Miner, Robert & Popelier, Nico (eds.) 2003. Mathematical Markup Language (MathML) Version 2.0 (Second Edition). W3C Recommendation 21 October 2003. <http://www.w3.org/TR/2003/REC-MathML2-20031021>. [Last retrieved 2007-04-01]
- Castlebury, Lisa A. & Farr, David F. 2002. The genus *Tilletia* in the United States. Systematic Botany & Mycology Laboratory, ARS, USDA. <http://nt.ars-grin.gov/taxadescriptions/tilletia>. [Page last modified 2002-03-21, last retrieved 2007-04-22]
- Catapano, Terry; Agosti, Donat; Sautter, Guido; Koning, Drew; Boehm, Klemens; Johnson, Norman F.; Heidorn, P. Bryan; Moritz, Thomas D.; Sarkar, Indra Neil & Stephenson, Christie 2006. TaxonX: A lightweight and flexible xml schema for mark-up of taxonomic treatments. *In*: Belbin, Lee; Rissoné, Adrian & Weitzman, Anna (eds.) Proceedings of TDWG, Abstracts of the 2006 Annual Conference of Biodiversity Information Standards (TDWG), St. Louis, USA, October 2006: 57-58. <http://www.tdwg.org/proceedings/article/view/34>. [Last retrieved 2007-05-01]
- CBIT 2004. Creating an Expert Route. <http://www.lucidcentral.com/lucid2/builder/webhelp/expertroute.htm>. [Page last modified 2004-06-15, last retrieved 2007-04-14]
- CBIT 2007a. Lucidcentral.com – identification and diagnostic tools. <http://www.lucidcentral.org/lucid3>. [Last retrieved 2007-04-10; referring to Lucid3 version 3.4]
- CBIT 2007b. Lucid Phoenix Version 1. <http://www.lucidcentral.com/phoenix>. [Last retrieved 2007-04-10]
- CED 1992. Collins English Dictionary and Thesaurus. Version 1.5, based on 3rd ed. of printed version. HarperCollins (CD-ROM).
- Chalubert, Antoine & Vignes Lebbe, Régine 2006. A new model for descriptive knowledge. Abstract: <http://www.tdwg.org/proceedings/article/view/32> and slide: http://tdwg2006.tdwg.org/fileadmin/2006meeting/slides/Chalubert_ModelForDescrKnowledge.pdf. [Last retrieved 2007-04-20]
- Chesmore, D.; Bernard, T.; Inman, A. J. & Bowyer, R. J. 2003. Image analysis for the identification of the quarantine pest *Tilletia indica*. EPPO Bulletin **33** (3): 495-499.
- Choo, M. H. C. 2002. DELIA – The DELTA integrator. An integrated DELTA database management software for Windows 95/98/NT. Department of Conservation and Land Management. <http://www.naturebase.net/content/view/2401/482>. [Last retrieved 2007-04-20, software version available for testing: 1.0 beta]
- Choo, M. H. C. & Spooner, A. 2001. Integration of taxonomic descriptive data across multiple database platforms and softwares (Weed Information Network – a case study). <http://delta-intkey.com/www/delia.htm>. [Last retrieved 2007-05-09. Originally at <http://www.tdwg.org/2001meet/choo.htm>, no longer available.]
- Clark, J. (ed.) 1999. XSL Transformations (XSLT) Version 1.0. W3C Recommendation 16 November 1999. <http://www.w3.org/TR/1999/REC-xslt-19991116>. [Last retrieved 2007-04-01]
- Clark, J. Y. 2003. Artificial neural networks for species identification by taxonomists. *BioSystems* **72** (1-2; special issue “Computational intelligence in bioinformatics”): 131-147.
- Clark, J. Y. & Warwick, K. 1998. Artificial keys for botanical identification using a multilayer perceptron neural network (MLP). *Artificial Intelligence Review, Special Issue on Applications in Biology and Agriculture* **12** (1-3): 95-115.

- Colless, D. H. 1985. On "character" and related terms. *Systematic Zoology* **34**: 229-233.
- Connolly, Thomas & Begg, Carolyn 2002. Database systems. A practical approach to design, implementation and management. 3rd edition. Addison Wesley: Harlow (UK). 1236 pp.
- Cowan, R. S.; Chase, M. W.; Kress, W. J. & Savolainen, V. 2006. 300 000 species to identify: problems, progress, and prospects in DNA barcoding of land plants. *Taxon* **55**: 611-616.
- Cracraft, J. 2002. The seven great questions of systematic biology: an essential foundation for conservation and the sustainable use of biodiversity. *Annals of the Missouri Botanical Garden* **89**: 127-144.
- Cross, N. 1997. Delta Access Perl FAQ. [Used to be available at http://www.herbaria.harvard.edu/computerlab/web_keys/delta_access_perl.html, but as of 2004 it was no longer found]
- Cui, Hong & Heidorn, P. B. 2007. The reusability of induced knowledge for the automatic semantic markup of taxonomic descriptions. *Journal of the American Society for Information Science and Technology* **58**(1): 133-149. [as of 2007-04-15 preprint retrieved at <http://hong.fims.uwo.ca/Research/jasist06.pdf>]
- Cui, Hong; Heidorn, P. Bryan & Zhang, Hong 2002. An approach to automatic classification of text for information retrieval. Proceedings of the 2nd ACM/IEEE-CS joint conference on digital libraries, Portland, Oregon, USA: 96-97.
- Cui, Hong; McCourt, Richard M. & Feist, Monique 2006. Unsupervised structure discovery for biodiversity information. Proceedings of the 6th ACM/IEEE-CS joint conference on digital libraries, Chapel Hill, NC, USA: 382-382.
- Dallwitz, Mike J. 1974. A flexible computer program for generating identification keys. *Systematic Zoology* **23**: 50-57.
- Dallwitz, Mike J. 1980. A general system for coding taxonomic descriptions. *Taxon* **29**: 41-46.
- Dallwitz, Mike J. 1993a. DELTA and Intkey. *In*: Fortuner, R. (ed.) *Advances in computer methods for systematic biology*. John Hopkins Univ. Press: Baltimore, USA chapter 18: 287-296.
- Dallwitz, Mike J. 1993b. Reply to Richard Pankhurst's comments on 'Preliminary suggestions for new features'. *DELTA Newsletter* **9**: 16-17.
- Dallwitz, Mike J. 1993c. Reply to Eric Gouda's comments on 'Preliminary suggestions for new features for the DELTA system'. *DELTA Newsletter* **9**: 23.
- Dallwitz, Mike J. 2005a. A comparison of interactive identification programs. <http://delta-intkey.com/www/comparison.pdf>. [Dated 13 September 2005, last retrieved 2007-03-31. First version 2000, originally at <http://biodiversity.uno.edu/delta/www/comparison.htm>, no longer available.]
- Dallwitz, Mike J. 2005b. Desirable attributes for interactive identification programs. <http://delta-intkey.com/www/idcriteria.pdf>. [Dated 21 October 2006, last retrieved 2007-03-01. First version 2000, originally at <http://biodiversity.uno.edu/delta/www/idcriteria.htm>, no longer available.]
- Dallwitz, Mike J. 2005c. Data requirements for natural-language descriptions and identification. <http://delta-intkey.com/www/descdata.htm>. [Dated 2005-09-13, last retrieved 2007-05-07]
- Dallwitz, Mike J. 2006. A comparison of formats for descriptive data. <http://delta-intkey.com/www/compdata.pdf>. [Dated 12 Sept. 2005, last retrieved 2007-03-01. First version 1999, originally at <http://biodiversity.uno.edu/delta/www/compdata.htm>, no longer available.]
- Dallwitz, Mike J. 2007. Programs for interactive identification and information retrieval. <http://delta-intkey.com/www/idprogs.pdf>. [Dated 14 Feb. 2007, last retrieved 2007-03-31. First version 1999.]
- Dallwitz, Mike J. & Paine, T. A. 1999. Definition of the DELTA format. Distributed as a MS Word document with the CSIRO DELTA editor for Windows, version 1.3.0.8. Dated 31. May 1999 [Last retrieved 2004-01-07. Previously at <http://biodiversity.uno.edu/delta/standard/standard.exe>, PC-executable self-extracting archive containing Word for Windows 95 document, no longer available.]
- Dallwitz, Mike J. & Paine, T. A. 2005. Definition of the DELTA format. <http://delta-intkey.com/www/standard.htm> or <http://delta-intkey.com/www/standard.pdf> [Last retrieved 2007-05-07]
- Dallwitz, Mike J.; Paine, T. A. & Zurcher, E. J. 1993. Preliminary suggestions for new features for the DELTA system. *DELTA Newsletter* **9**: 2-13.
- Dallwitz, Mike J.; Paine, T. A. & Zurcher, E. J. 2000a. User's guide to the DELTA system. A general system for processing taxonomic descriptions. Edition 4.12. 156 pp. CSIRO Division of Entomology: Canberra (Australia). [Included in the CSIRO DELTA program package]
- Dallwitz, Mike J.; Paine, T. A. & Zurcher, E. J. 2000b. User's Guide to Intkey. A Program for Interactive Identification and Information Retrieval. Edition 1.09. v + 23 pp. CSIRO Division of Entomology: Canberra (Australia). [Included in the CSIRO DELTA program package]

- Dallwitz, Mike J.; Paine, T. A. & Zurcher, E. J. 2005. Proposed New Features for the DELTA System. <http://delta-intkey.com/www/proposal.htm> [Last retrieved 2007-04-16. This is a revised version of the printed publication Dallwitz & al. 1993. Other dated revisions have been published but are no longer available. The versions dated 1995-05-08 (<http://biodiversity.uno.edu/delta/standard/newdelta.txt>, available until 1998), 1999-01-18 (<http://biodiversity.uno.edu/delta/www/proposal.htm>), and 2003-03-24 (under current URL) have been compared with the version dated 2004-07-30; the differences are relatively minor and largely corrections, clarifications and improvements in formatting. They can therefore be considered as a single publication as proposed by the authors.]
- Dallwitz, Mike J.; Paine, T. A. & Zurcher, E. J. 2006. Principles of interactive keys. <http://delta-intkey.com/www/interactivekeys.htm> and <http://delta-intkey.com/www/interactivekeys.pdf>. [Page last modified 2006-03-14, last retrieved 2007-05-07. First version 2000.]
- Dallwitz, Mike J.; Paine, T. A. & Zurcher, E. J. 2007. Interactive identification using the Internet. <http://delta-intkey.com/www/netid.htm> or <http://delta-intkey.com/www/netid.pdf>. [Page last modified 2007-04-07, last retrieved 2007-05-07. First version 2002.]
- Davis, P. H. & Cullen, J. 1989. The identification of flowering plant families. 3rd ed. Cambridge University Press: Cambridge (UK).
- Delgado Calvo-Flores, Miguel; Fajardo Contreras, W.; Gibaja Galindo, E. L. & Pérez-Pérez, R. 2005 (2005 online, 2006 print). XKey: A tool for the generation of identification keys. *Expert Systems with Applications* **30** (2): 337-351. <http://dx.doi.org/10.1016/j.eswa.2005.07.034>.
- Diederich, Jim 1997. Basic properties of biological databases: character development and support. *Mathematical Computer Modelling* **25** (10): 109-127.
- Diederich, Jim & Milton, Jack 1989. NEMISYS, An expert system for nematode identification. *In: Fortuner, R. (ed.) Nematode identification and expert-system technology*. Plenum Publishing Corp.: New York, 45-63.
- Diederich, Jim & Milton, Jack 1991. Creating domain specific metadata for scientific data and knowledge bases. *IEEE Transactions on Knowledge and Data Engineering* **3** (4): 421-434.
- Diederich, Jim & Milton, Jack 1993a. Expert workstations: a tool-based approach. Chapter 7 *In: Fortuner, R. (ed.) Advances in computer methods for systematic biology*. John Hopkins Univ. Press: Baltimore, USA, 103-123.
- Diederich, Jim & Milton, Jack 1993b. NEMISYS: a computer perspective. Chapter 10 *In: Fortuner, R. (ed.) Advances in computer methods for systematic biology*. John Hopkins Univ. Press: Baltimore, USA, 165-179.
- Diederich, Jim; Fortuner, Renaud & Milton, Jack 1989. Building a knowledge base for plant-parasitic nematodes: description and specification of metadata. *In: Fortuner, R. (ed.) Nematode identification and expert-system technology*. Plenum Publishing Corp.: New York, 65-76.
- Diederich, Jim; Fortuner, Renaud & Milton, Jack 1997. Construction and integration of large character sets for nematode morpho-anatomical data. *Fundamental and Applied Nematology* **20** (5): 409-424.
- Diederich, Jim; Fortuner, Renaud & Milton, Jack 1998. A general structure for biological databases. *In: Bridge, P.; Jeffries, P.; Morse, D. R. & Scott, P. R. (eds.) Information technology, plant pathology and biodiversity*. CAB International: Wallingford, UK: 47-58.
- Diederich, Jim; Fortuner, Renaud & Milton, Jack 1999. Computer-assisted data extraction from the taxonomical literature. <http://math.ucdavis.edu/~milton/genisys.html>. [Last retrieved 2007-05-10]
- Diederich, Jim; Fortuner, Renaud & Milton, Jack 2000a. Genisys and computer-assisted identification of nematodes. *Nematology* **2** (1) 17-30.
- Diederich, Jim, Fortuner, Renaud & Milton, Jack 2000b. A uniform representation for the plan of organization of nematodes of the order Tylenchida. *Nematology* **2** (8): 805-822.
- DiGIR 2005. Distributed Generic Information Retrieval (DiGIR). <http://digir.sourceforge.net>. [Dated "9 Dec Jun 2005", last retrieved 2007-04-28]
- Dmitriev, Dmitry A. 2006. 3I: On-line virtual taxonomic revisions. Proceedings of TDWG: Abstracts of the 2006 Annual Conference of Biodiversity Information Standards (TDWG), 15-22. October 2006, Missouri Botanical Garden, St. Louis, Missouri, U.S.A.: 26-27. <http://www.tdwg.org/proceedings/article/view/10>. [Last retrieved 2007-05-01]
- Dmitriev, Dmitry A. 2007. 3I interactive keys and taxonomic databases. <http://ctap.inhs.uiuc.edu/dmitriev>. [First published July 9, 2003-07-09, last updated "2007-04-20", last retrieved 2007-04-20]
- Do, M. T.; Harp, J. M. & Norris, K. C. 1999. A test of a pattern recognition system for identification of spiders. *Bulletin of Entomological Research* **89**: 217-224.

- Dodds, L. 1999. XDELTA – Deriving an XML based format for taxonomic information. <http://www.ldodds.com/delta>. [Last modified 1999-10-22, last retrieved 2007-05-13]
- Duke, J. A. 1969. On tropical tree seedlings I. Seeds, seedlings, systems and systematics. *Annals of the Missouri Botanical Garden* **56** (2): 125-161.
- EB 2001. Encyclopædia Britannica Deluxe Edition CD-ROM.
- Edwards, M. & Morse, David R. 1995. The potential for computer-aided identification in biodiversity research. *Trends in Ecology and Evolution* **10** (4): 153-158.
- Elix, J. A.; Johnston, J. & Parker, J. L. 1988. A computer program for the rapid identification of lichen substances. *Mycotaxon* **31** (1): 89-99.
- EOL.org 2007. Press release, May 9, 2007. http://www.eol.org/press_release.html. [Last retrieved 2007-04-16]
- Exeter Software (undated). XID Authoring System, Version 3, for Windows. <http://www.exetersoftware.com/cat/xid.html> and <http://www.exetersoftware.com/cat/xidinfo.html>. [Last retrieved 2007-05-01]
- Fajardo Contreras, W.; Gibaja Galindo, E. L.; Bailón Morillas, A. & Moral Lorenzo, P. 2003. An application of expert systems to botanical taxonomy. *Expert Systems with Applications* **25**: 425-430.
- Fallside, David C. (ed.) 2001. W3C XML Schema Part 0: Primer. W3C Recommendation, 2 May 2001. <http://www.w3.org/TR/2001/REC-xmlschema-0-20010502>. [Last retrieved 2007-04-01. See also Thompson & al. 2001 and Biron & Malhotra 2001]
- Farr, David F. 2006. On-line keys: more than just paper on the web. *Taxon* **55** (3): 589-596.
- Felsenstein, Joseph 1985. Confidence limits on phylogenies: An approach using the bootstrap. *Evolution* **39**: 783-791.
- Felsenstein, Joseph 2004. *Inferring phylogenies*. Sinauer: Sunderland (USA). 664 pp.
- Findling, Axel 1998a. DAP – Ein Web-Interface zu DeltaAccess. <http://www.axel-findling.de/programs/dap>. [Last retrieved 2007-05-01]
- Findling, Axel 1998b. DAWI – Ein Web-Interface zu DeltaAccess. <http://www.axel-findling.de/programs/dawi>. [Last retrieved 2007-05-01]
- Fortuner, Renaud 1989. A new description of the process of identification of plant-parasitic nematode genera. *In: Fortuner, R. (ed.) Nematode identification and expert-system technology*. Plenum Publishing Corp.: New York, 35-44.
- Fortuner, Renaud 1993. The NEMISYS solution to problems in nematode identification. Chapter 9 *In: Fortuner, R. (ed.) Advances in computer methods for systematic biology*. John Hopkins Univ. Press: Baltimore, USA: 137-164.
- Fortuner, Renaud 2002. Uniformity and representation of taxonomic and other characters and semi-automatic extraction using computer tools. *Nematology* **4** (5): 583-591.
- Fowler, Martin & Scott, Kendall 2001. UML distilled – second edition. A brief guide to the standard object modeling language. 7th printing (printings differ in content). Addison-Wesley: Boston (USA). 186 pp.
- Gaston, Kevin J. & O'Neill, Mark A. 2004. Automated species identification: why not? (One contribution of 19 to a theme issue 'taxonomy for the twenty-first century'). *Philosophical Transactions of the Royal Society B: Biological Sciences* **359**: 655-667.
- Geoffroy, M. 2003. Toward the implementation of the "transmission engine". *Schriftenreihe Vegetationskunde* **39**: 87-112.
- Geoffroy, M. & Berendsohn, W. G. 2003. The concept problem in taxonomy: importance, components, approaches. *Schriftenreihe Vegetationskunde* **39**: 5-14.
- Germeier, C. U. & Frese, L. 2001. A data model for the evaluation and characterisation of plant genetic resources. *In: Swiecicki, W.; Naganowska, B.; Wolkon, B. (eds.): Broad variation and precise characterisation – limitation for the future. Proceedings of the XVIth Eucarpia, Section Genetic Resources Workshop, May 16-20, 2001, Poznan, Polen: 174-177.*
- Gibaja Galindo, E. L. 2004. Modelos de representación del conocimiento para la identificación taxonómica y aplicaciones. Tesis doctoral, Universidad de Granada. hera.ugr.es/tesisugr/15759969.pdf
- Gielis, Johan 2003. A generic geometric transformation that unifies a wide range of natural and abstract shapes. *American Journal of Botany* **90**: 333-338.
- Götz, Erich 2003. *Pflanzen bestimmen mit dem PC. Farn- und Blütenpflanzen Deutschlands, 3300 farbige Pflanzenfotos*. 2. Aufl. Eugen Ulmer Verlag: Stuttgart (Germany).
- Gouda, E. J. 1993. Some questions and notes on the new features document. *DELTA Newsletter* **9**: 22.
- Gouda, E. J. 1996. TAXASOFT DELTA Editor. *DELTA Newsletter* **12**: 12-14.
- Gouda, E. J. 2001. TAXASOFT DELTA Programs. <http://www.delta-intkey.com/taxasoft>. [Last retrieved 2007-04-21. Previously at <http://www.delta-intkey.com/taxasoft>]

- biodiversity.uno.edu/delta/taxasoft, last retrieved 2004-02-10, no longer available.]
- Goujon, Pierre 2007. Identification assistée par ordinateur (IAO). Système d'identification interactive multimedia. <http://abiris.snv.jussieu.fr/identification/introduction.html>. [Page last modified 2007-03-14, last retrieved 2007-05-10]
- Gray, Asa 1878. Synoptical flora of North America. Vol. 2, part 1. Ivison, Blakeman, Taylor and Co.: New York (USA).
- Greenberg, Jane; Heidorn, Bryan; Seiberling, Stephen & Weakley, Alan S. 2005. Growing vocabularies for plant identification and scientific learning. International conference on Dublin Core and metadata applications (DC-2005, Sept 15, 2005), Madrid, Spain. <http://www.slais.ubc.ca/PEOPLE/faculty/tennis-p/dcpapers/paper14.pdf>. [Last retrieved 2007-05-01]
- Greenberg, Jane; Heidorn, Bryan; Seiberling, Stephen & Weakley, Alan S. 2006. Growing vocabularies for plant identification and scientific learning. *Bulletin of the American Society of Information Science & Technology*. June 2006. http://www.asis.org/Bulletin/Jun-06/greenberg_heidorn_sieberling_weakley.html. [Last retrieved 2007-05-01]
- Guala, Gerald F. 2006. SLIKS, Stinger's Lightweight Interactive Key Software. <http://www.stingersplace.com/SLIKS>. [Software version 2.1; page last modified: 2006-11-05; last retrieved 2007-04-10. First version 2004-06.]
- Haber, William A. 2006. Monteverde biodiversity: natural history field sheets for printing and lamination. <http://efg.cs.umb.edu/monteverde/Lam.html>. [Last modified: 2006-11-23, last retrieved 2007-04-21]
- Hagedorn, Gregor 1997. DeltaAccess – an SQL interface to DELTA (Description Language for Taxonomy), implemented in Microsoft Access. Version 1.0. http://www.DiversityWorkbench.net/Workbench/Descriptions/Download/10/DA97_10.exe. [Last retrieved 2007-04-21. Originally at <http://www.bgbm.fu-berlin.de/Projects/DeltaAccess/DeltaAccess.html>, no longer available.]
- Hagedorn, Gregor 1998a. Making Delta Accessible: Databasing descriptive information. IX OPTIMA congress, Paris 11-17 Mai. Organisation pour l'Etude Phyto-Taxonomique de la Région Méditerranéenne, Muséum National d'Histoire Naturelle, Paris. p. 31.
- Hagedorn, Gregor 1998b. Databases as working tools for the mycologist. Sixth International Mycological Congress IMC6. Jerusalem, Israel, August 23-28 1998, Abstracts. p. 55.
- Hagedorn, Gregor 1999a. DeltaAccess: 'Describe' & 'Identify'. User Guide and Documentation, version 1.6. (160 printed pages). <http://www.diversityworkbench.net/OldModels/Descriptions/Docu160/DeltaAccess.hlp> as Windows help file, or <http://www.diversityworkbench.net/OldModels/Descriptions/Docu160/DeltaAccess.html> as start page to a set of html pages. [Last retrieved 2007-04-21]
- Hagedorn, Gregor 1999b. TDWG working group: Structure of Descriptive Data. Minutes of the workgroup session at the TDWG 1999 meeting at Harvard University, 31. October 1999. http://www.diversitycampus.net/Projects/TDWG-SDD/Minutes/1999TDWG-SDD-Minutes_11.html. [Last retrieved 2007-05-01]
- Hagedorn, Gregor 2000a. Globales Informationssystem zur Biodiversität pflanzenpathogener Pilze (GLOPP). *Nachrichtenblatt des Deutschen Pflanzenschutzdienstes* **52** (6): 149.
- Hagedorn, Gregor 2000b. TDWG working group: Structure of Descriptive Data. Minutes of SDD session at the TDWG meeting in Frankfurt, 12. Nov. 2000. http://www.diversitycampus.net/Projects/TDWG-SDD/Minutes/2000TDWG-SDD-Minutes_11.html. [Last retrieved 2007-05-01]
- Hagedorn, Gregor 2001a. Making DELTA accessible: Databasing descriptive information. *Bocconea* **13**: 261-280.
- Hagedorn, Gregor 2001b. TDWG working group: Structure of Descriptive Data. Minutes of session at the TDWG meeting in Sydney, 10. November 2001. http://www.diversitycampus.net/Projects/TDWG-SDD/Minutes/2001TDWG-SDD-Minutes_10.html. [Last retrieved 2007-05-01]
- Hagedorn, Gregor 2001c. Documentation of the information model for DiversityReferences (Special Indexing, 0.9). http://www.diversityworkbench.net/OldModels/References/Model/2001-05-08/DiversityReferencesData_Indexing_Model.html. [Last retrieved 2007-05-01. Originally at http://www.diversitycampus.net/Workbench/References/Model/2001-05-08/DiversityReferencesData_Indexing_Model.html]
- Hagedorn, Gregor 2002a. Information systems can improve the efficiency of biodiversity research and promote collaboration and sharing of information (Symposium contribution 240). *In*: IMC7 Book of abstracts. The 7th International Mycological Congress, Oslo, 11-17. Aug. 2002. p. 77.
- Hagedorn, Gregor 2002b. TDWG working group: Structure of Descriptive Data. Minutes of working sessions in Australia, 11-14. March

2002. http://www.diversitycampus.net/Projects/TDWG-SDD/Minutes/2002Australia-SDD-Minutes_11.html. [Last retrieved 2007-05-01]
- Hagedorn, Gregor 2002c. TDWG SDD: Structure of Descriptive Data. Convener's report. <http://www.cria.org.br/eventos/tdbi/tdwg/presentations/tdwgsdd.ppt>. [Last retrieved 2007-05-01]
- Hagedorn, Gregor 2002d. Diversity Workbench: A biodiversity component framework. Draft Version 0.3. http://www.diversityworkbench.net/OldModels/Framework/DW_Framework03.pdf. 19 printed pages. [Last retrieved 2007-05-01. Originally at http://www.DiversityCampus.net/Workbench/Framework/DW_Framework03.pdf]
- Hagedorn, Gregor 2003a. TDWG working group: Structure of Descriptive Data (SDD) – Minutes of working sessions in Indaiatuba, Brazil, 14-17. October 2002. (24 printed pages). http://www.DiversityCampus.net/Projects/TDWG-SDD/Minutes/2002TDWG-SDD-Minutes_10.html. [Last retrieved 2007-05-01]
- Hagedorn, Gregor 2003b. TDWG working group: Structure of Descriptive Data (SDD) – Minutes of the SDD meeting in Paris, France, 13-16. February 2003. (24 printed pages). http://www.DiversityCampus.net/Projects/TDWG-SDD/Minutes/2003Paris-SDD-Minutes_10.html. [Last retrieved 2007-05-01]
- Hagedorn, Gregor 2003c. TDWG working group: Structure of Descriptive Data (SDD) – Minutes of the SDD meeting in Lisbon, Portugal, 20-26. October 2003. (24 printed pages). http://www.DiversityCampus.net/Projects/TDWG-SDD/Minutes/2003Lisbon-SDD-Minutes_10.html. [Last retrieved 2007-05-01]
- Hagedorn, Gregor 2003d. TDWG working group: Structure of Descriptive Data (SDD) – XML schema (version SDD 0.9). (89 printed pages). http://www.DiversityCampus.net/Projects/TDWG-SDD/Minutes/2003Lisbon_schema/SDD_09.xsd. [Last retrieved 2007-05-01]
- Hagedorn, Gregor 2003e. TDWG working group: Structure of Descriptive Data (SDD) – Overview of available schema documentation (version SDD 0.9). (3 printed pages). http://www.DiversityCampus.net/Projects/TDWG-SDD/Minutes/2003Lisbon_schema/SDD_09_DocuOverview.html. [Last retrieved 2007-05-05]
- Hagedorn, Gregor 2003f. SDD proposal: Free-form text data elements. Version 2. http://www.DiversityCampus.net/Projects/TDWG-SDD/Docs/SDD_P_DataTypes_Text.html. [Last retrieved 2007-05-06]
- Hagedorn, Gregor 2003g. SDD proposal: Scoring sequence of states in descriptions. Version 2. http://www.DiversityCampus.net/Projects/TDWG-SDD/Docs/SDD_P_DescrScoringSequence.html. [Last updated 2003-10-31, last retrieved 2007-05-06]
- Hagedorn, Gregor 2004a. DiversityModelDocumenter (release 2.6). <http://www.diversityworkbench.net/OldModels/ModelDocumenter/Docu/DiversityModelDocumenter.html>. [Last retrieved 2007-05-05. Originally at <http://www.DiversityCampus.net/Workbench/ModelDocumenter/Docu/DiversityModelDocumenter.html>.]
- Hagedorn, Gregor 2004b. SDD proposal: Indicators of coding status in class or object descriptions. Version 3. http://www.DiversityCampus.net/Projects/TDWG-SDD/Docs/SDD_P_Data_CodingStatus03.html. [Last retrieved 2007-05-05]
- Hagedorn, Gregor 2004c. TDWG working group: Structure of Descriptive Data (SDD) – Overview of available schema versions and documentation (UBIF and SDD 1.0). (4 printed pages). http://www.DiversityCampus.net/Projects/TDWG-SDD/Minutes/2004NZ_schema/DocuOverview.html. [Last retrieved 2007-05-05]
- Hagedorn, Gregor 2004d. TDWG working group: Structure of Descriptive Data (SDD) – Minutes of the SDD meeting in Christchurch, New Zealand, 11-17. October 2004. (23 printed pages). http://www.DiversityCampus.net/Projects/TDWG-SDD/Minutes/2004Christchurch-SDD-Minutes_09.html. [Last retrieved 2007-05-05]
- Hagedorn, Gregor 2005a. SDD document: Rating parameters for character guidance in identification. Version 1. http://www.DiversityCampus.net/Projects/TDWG-SDD/Docs/SDD_P_ID_Ratings.html. [Last retrieved 2007-05-05]
- Hagedorn, Gregor 2005b. DiversityDescriptions (DeltaAccess) (version 1.9). Documentation of the information model. <http://www.diversityworkbench.net/OldModels/Descriptions/Model/2005-03-30/DModelDD19.html>. [Last retrieved 2007-05-05. Originally at <http://www.diversitycampus.net/Workbench/Descriptions/Model/2005-03-30/DModelDD19.html>.]
- Hagedorn, Gregor 2006. Minutes of the SDD Berlin 2006 Meeting (April). <http://wiki.tdwg.org/twiki/bin/view/SDD/SDD2006BerlinMinutes>. [Last retrieved 2007-05-05]
- Hagedorn, Gregor & Kohlbecker, Andreas 2006. DiversityResources information model (version 1.3). http://www.diversityworkbench.net/Portal/wiki/ResourcesModel_v1.3. [Last retrieved 2007-05-05]

- Hagedorn, Gregor & Rambold, Gerhard 2000. A method to establish and revise descriptive data sets over the Internet. *Taxon* **49**: 517-528.
- Hagedorn, Gregor & Weiss, Markus 2002. DiversityCollection information model. <http://www.diversityworkbench.net/OldModels/Collection/Model/2002-11-15/DiversityCollectionModel.html>. [Last retrieved 2007-05-05. Originally at <http://www.DiversityCampus.net/Workbench/Collection/Model/2002-11-15/DiversityCollectionModel.html>.]
- Hagedorn, Gregor; Deml, Günther; Burhenne, Matthias; Guerrero Cartin, O. M.; Gräfenhan, T. & Weiss, Markus 2000. Synoptische, computergestützte Identifizierung von Pflanzenpathogenen. 52. Deutsche Pflanzenschutztagung in Weihenstephan (Technische Universität München) vom 9. bis 12. Oktober 2000. p. 541.
- Hagedorn, Gregor; Deml, Günther; Triebel, Dagmar; Piepenbring, Meike & Oberwinkler, Franz 2001. GLOPP – Global information system for the biodiversity of plant pathogenic fungi. BIOLOG German Programme on Biodiversity and Global Change (Phase I, 2000-2004) Funded by BMBF, Status Report 2001, 208-209.
- Hagedorn, Gregor; Glied, Matthias.; Weiss, Markus & Gräfenhan, Tom 2002. DiversityWorkbench – A framework to manage biodiversity information. (Oral contribution in the symposium “Storage and Retrieval of Morphological Data for Phylogenetic Analysis (S22)”, Saturday 14th September 2002.) *In: ICSEB VI Sixth International Congress of Systematic and Evolutionary Biology “Biodiversity in the Information Age”*. Abstracts. Sept. 9-16, 2002, Patras, Greece, p. 198.
- Hagedorn, Gregor; Oberwinkler, F.; Berndt, R.; Braun, U.; Burhenne, Matthias; Deml, Günther; Glied, M.; Göker, M.; Gräfenhan, T.; Hou, C.; Kainz, C.; Piepenbring, M.; Riethmüller, A.; Ritschel, A.; Scholler, M.; Triebel, Dagmar & Weiss, Markus 2003a. The Global Plant Pathogen Index (GLOPP). [Poster G13 in BIOLOG Biodiversity Informatics] *In: International Symposium “Sustainable use and conservation of biological diversity – A challenge for society.”* Symposium Report Part A. 1-4 December 2003, Berlin, p. 441
- Hagedorn, Gregor; Deml, Günther & Triebel, Dagmar 2003b. Expansion of the GLOPP information system through integration of the data collection of H. & H. Doppelbauer. [Poster in GBIF-D: Fungi & lichens]. *In: International Symposium “Sustainable use and conservation of biological diversity – A challenge for society.”* Symposium Report Part A. 1-4 December 2003, Berlin, p. 248.
- Hagedorn, Gregor; Thiele, Kevin; Morris, Robert & Heidorn, P. Bryan 2005. The “Structured Descriptive Data (SDD)” w3c-xml-schema, version 1.0. <http://rs.tdwg.org/UBIF/2005/rddl.html>. [Last retrieved 2007-05-05]
- Hagedorn, Gregor; Thiele, Kevin; Morris, Robert & Heidorn, P. Bryan 2006. The “Structured Descriptive Data (SDD)” w3c-xml-schema, version 1.1. <http://rs.tdwg.org/UBIF/2006/rddl.html>. [Last retrieved 2007-05-05]
- Hall, A. V. 1970. A computer-based system for forming identification keys. *Taxon* **19**: 12-18.
- Haller, B. & Probst, W. 1989. Exkursionen im Sommerhalbjahr. 2. Auflage, Gustav Fischer, Stuttgart.
- Hawksworth, D. L. 1991. The fungal dimension of biodiversity: magnitude, significance, and conservation. *Mycological Research* **95**: 641-655.
- Hawksworth, D. L. & Kalin-Arroyo, M. T. 1995. Magnitude and distribution of biodiversity. *In: Heywood, V. H. (ed.) Global Biodiversity Assessment*, Cambridge Univ. Press, Cambridge: 107-191.
- Heidorn, P. Bryan, Palmer, Carole & Wright, Dan 2007. From bioinformatics to biological informatics specialists. *Journal of Biomedical Discovery and Collaboration* (**2**) 1. <http://www.j-biomed-discovery.com/content/2/1/1>. [Last retrieved 2007-04-13]
- Hein, Jotus 1989. A new method that simultaneously aligns and reconstructs ancestral sequences for any number of homologous sequences, when the phylogeny is given. *Molecular Biology and Evolution* **6** (6): 649-668.
- Hernandez, J. R.; Hennen, J. F.; Farr, David F. & McCray, E. 2004. A model for presenting systematic data on the internet. *Mycological Research* **108**: 3-4.
- Higgins, D. G.; Sharp, Paul M. 1989. Fast and sensitive multiple sequence alignments on a microcomputer. *Computer Applications in the Biological Sciences (CABIOS)* **5**: 151-153.
- Hoppe, Jürgen R. 1998. Practical suggestions for database implementations. *In: Thiéry, M.; Stevens, A.-D.; Hoppe, J. R.; Charles-Dominique, P. & Schuchmann, K.-L. Angiosperm pollination and seed dispersal, a review. Ecotropica* **4**: 69-91.
- Hoppe, Jürgen R.; Boos, Evelin & Stützel, Thomas 1999. SysTax – ein Datenbanksystem für Systematik und Taxonomie. *In: Begemann, F.; Harrer, S. & Jiménez Krause, J. D. (eds.) Dokumentationen und Informationssysteme im Bereich pflanzen genetischer Ressourcen in Deutschland. Schriftenreihe zu Genetischen Ressourcen (Zentralstelle für Agrardokumentation und -information – ZADI, Informations-*

- zentrum für Genetische Ressourcen – IGR): 64-79.
- Hoppe, Jürgen R.; Boos, Evelin; Ludwig, Thorsten & Wiedemann, Michael 2004. SysTax – a database system for systematics and taxonomy. ER-diagram. http://www.biologie.uni-ulm.de/systax/documentation/er/er_poster.pdf. [Last retrieved 2007-05-10]
- Hoppe, Jürgen R.; Boos, Evelin; Ludwig, Thorsten; Wiedemann, Michael & Stützel, Thomas 2007. SysTax – a database system for systematics and taxonomy. <http://www.biologie.uni-ulm.de/systax>. [Last retrieved 2007-05-10]
- Houston, W. W. K.; Payne, S. & Fitzsimmons, N. J. 2002. Platypus: A database package for taxonomists. CD-ROM. ISBN: 0643066284. CSIRO Publishing/Australian Biological Resources Study (ABRS).
- Huelsenbeck, J. P. & Rannala, B. 2003. Detecting correlation between characters in a comparative analysis with uncertain phylogeny. *Evolution* **57**(6): 1237-1247.
- Hull, David L. 1988. Science as a process: An evolutionary account of the social and conceptual development of science. University of Chicago Press.
- Hull, David L. & Ruse, Michael 1998. The philosophy of biology. Oxford University Press.
- IdentifyLife 2005. IdentifyLife. <http://www.identifylife.org>. [Last modified 2005-08, last retrieved 2007-04-17]
- Ilic, Katica; Kellogg, Elizabeth A.; Jaiswal, Pankaj; Zapata, Felipe; Stevens, Peter F.; Vincent, Leszek P.; Avraham, Shulamit; Reiser, Leonore; Pujar, Anuradha; Sachs, Martin M.; Whitman, Noah T.; McCouch, Susan R.; Schaeffer, Mary L.; Ware, Doreen H.; Stein, Lincoln D. & Rhee, Seung Y. 2006. Plant structure ontology. Unified vocabulary of anatomy and morphology of a flowering plant. *Plant Physiology Preview*: <http://www.plantphysiol.org/cgi/rapidpdf/pp.106.092825v1>.
- Inglis, W. G. 1991. Characters: the central mystery of taxonomy and systematics. *Biological Journal of the Linnean Society* **44**: 121-139.
- Ingrisch, S.; Lampe, K.-H.; Riede, K. & Dietrich, C. 2001. DORSA – German Orthopteran collections. BIOLOG German Programme on Biodiversity and Global Change (Phase I, 2000-2004) Funded by BMBF, Status Report 2001, 200-201.
- Jacobson, Ivar; Booch, Grady & Rumbaugh, James 1999. The unified software development process. Addison-Wesley: Boston (USA).
- Janzen, D. H. 1991. How to save tropical biodiversity. *American Entomologist* **37**: 159-171.
- Janzen, D. H. 2004. Now is the time. *Philosophical Transactions of the Royal Society B: Biological Sciences* **359**: 731-732.
- Jensen, R. J. 1990. Detecting shape variation in oak leaf morphology: a comparison of rotational-fit methods. *American Journal of Botany* **77**: 1279-1293.
- Kennedy, Jessie 2003. Supporting taxonomic names in cell and molecular biology databases. *OmicS, a Journal of Integrative Biology* **7**(1): 13-16.
- Kennedy, M. J. & Spooner, N. R. 1994. The use of fuzzy-logic to aid in microorganism identification. A case study of *Haemophilus* species identification. *Binary-Computing in Microbiology* **6**: 132-135.
- Kirchgeßner, Norbert; Scharr, Hanno & Schurr, Uli 2002. Robust vein extraction on plant leaf images. <http://www.fz-juelich.de/icg/icg-iii/datapool/DataScharr/KirchgeßnerVIIP2002.pdf>. [Last retrieved 2007-04-26]
- Kirejtshuk, Alexander G., Lobanov, A. L. & Graničhin, O. N. 2005. Ипоект WebKey-X. <http://www.zin.ru/projects/WebKey-X>. [Published 2005-12, last retrieved 2007-04-20]
- Kirk, P. M.; Cannon, P. F. David, J. C. Stalpers, J. A. (eds.) 2001. Ainsworth and Bisby's dictionary of fungi. 9th Edition. CAB International: Wallingford, UK.
- Kirkbride, J. H. & Dallwitz, Mike J. 1993. Edited correspondence on DELTA enhancement proposals. *DELTA Newsletter* **9**: 18-21.
- Kirkup, D.; Malcolm, P.; Christian, G. & Paton, A. 2005. Towards a digital African Flora. *Taxon* **54**(2): 457-466.
- Klimov, P. 2001. Visual Key. <http://insects.ummz.lsa.umich.edu/ACARI/pklimov/VK>. [Last retrieved 2007-04-16]
- Klyne, Graham & Carroll, Jeremy J. (eds.) 2004. W3C Resource Description Framework (RDF): Concepts and Abstract Syntax. W3C Recommendation 10 February 2004. <http://www.w3.org/TR/2004/REC-rdf-concepts-20040210>. [Last retrieved 2007-04-01]
- Korf, R. P. 1972. Synoptic key to the genera of the Pezizales. *Mycologia* **64**: 937-994.
- Kornerup, A. & Wanscher, J. H. 1967. Methuen Handbook of Colour. 2nd edition. Methuen Co.: London (UK), 243 pp + 30 two-page color plates.
- Kuhn, T. S. 1970. The structure of scientific revolutions. 2nd edition. University of Chicago Press.
- Lane, M. A. & Edwards, J. L. 2007. The Global Biodiversity Information Facility. Chapter 1 *in*: Curry, G. & C. Humphries (eds.), Biodiversity databases: from cottage industry to industrial

- networks. Taylor & Francis, Boca Raton, Florida, USA.
- Lauber, Konrad & Wagner, Gerhart 2001. Flora Helvetica 2.0. CD-ROM. Ein interaktiver Führer durch die Pflanzenwelt der Schweiz. Haupt digital.
- Lawrence, George H. M. 1951. Taxonomy of vascular plants. Macmillan, New York.
- Leary, P. & Hagedorn, Gregor 2004. Converting LIF to SDD. <http://wiki.tdwg.org/twiki/bin/view/SDD/ConvertingLIF2SDD>. [Last retrieved 2007-04-28]
- Lebbe, J. 1984. Manuel d'utilisation du logiciel XPER. Micro Application, Paris. [An online version of this manual, file date 2003-09-19, is available at <http://lis.snv.jussieu.fr/apps/xper/doc/XPER.html>, last retrieved 2007-05-01]
- Lebbe, J. 1991. Représentation des concepts en biologie et en médecine. Introduction à l'analyse des connaissances et à l'identification assistée par ordinateur. Thèse de doctorat; Université Pierre et Marie Curie: Paris. xii + 282 + xxiv pp.
- Lebbe, J. & Vignes, R. 1989. Introduction to XPER. DELTA Newsletter **4**: 3-4.
- Lebbe, J. & Vignes, R. 1998. Modelling taxonomy description for identification. *In*: Bridge, P.; Jeffries, P.; Morse, D. R. & Scott, P. R. (eds.). Information technology, plant pathology and biodiversity. CAB International: Wallingford, UK: 37-46.
- Lebbe, J. & Vignes, R. 2003. Utilitaires XPER. <http://lis.snv.jussieu.fr/apps/xper/doc/utilxper>. [Last retrieved 2007-05-01]
- Lebbe, J.; Vignes, R. & Dedet, J. P. 1989. Computer aided identification of insect vectors. Parasitology Today **5**: 301-304.
- Leenhouts, P. W. 1966. Keys in biology. I. A survey and a proposal of a new kind. Proc. Koninklijke Nederlandse Akademie van Wetenschappen (Ser. C) **69**: 571-596.
- Leinberger, Dirk M.; Schumacher, Ulrike; Autenrieth, Ingo B. & Bachmann, Till T. 2005. Development of a DNA microarray for detection and identification of fungal pathogens involved in invasive mycoses. Journal of Clinical Microbiology **43**(10): 4943-4953.
- Lindh, Magnus 2003. Interaktiva nycklar – en enkel och effektiv metod för artbestämning! Utvärdering av interaktiva nycklar samt karaktärsdatabas och interaktiv nyckel över knappnålslavar i Norden. [= Interactive keys – an easy and efficient method for species identification! Evaluation of interactive identification programs and a character database and key of calicioid lichens of the Nordic countries.] Master thesis by Magnus Lindh 2003, Department of Conservation biology, Swedish University of Agricultural Sciences. Master thesis Uppsala. http://www.borealis.nu/exjobb/Interaktiva_nycklar.pdf. [Last retrieved 2007-04-28, Swedish]
- Lindh, Magnus & Thor, G. 2004. An interactive identification key to the calicioid lichens and fungi of the Nordic countries. Graphis Scripta (Stockholm) **16**: 28-30. Online: [http://www.nordiclichensociety.org/Graphis%20Scripta/Graphisindex/Articles/16\(1\)/interactkey28-30.pdf](http://www.nordiclichensociety.org/Graphis%20Scripta/Graphisindex/Articles/16(1)/interactkey28-30.pdf). [Last retrieved 2007-04-29]
- Little, Elbert L., Jr. 2002. Notes on tropical dendrology. Chapter 9. *In*: Vozzo, J. A. (ed.) 2002. Tropical tree seed manual. Agriculture Handbook 721. USDA Forest Service: Washington DC (USA). Free of charge. 899 pp.
- Lobanov, Andrei & Dianov, Mikhail 1999. PICKEY – Pictured interactive computerized biological key. <http://www.zin.ru/projects/pickey>. [Last modified 1999-03-23, last retrieved 2007-04-16]
- Lobanov, A. L.; Stepanjants, S. D. & Dianov, M. B. 1996. Dialogue computer system BIKEY as applied to diagnostics of Cnidaria (illustrated by an example of hydroids of the genus *Symplectoscyphus*). Scientia Marina (Special volume: S. Piraino, F. Boero, J. Bouillon, P. F. S. Cornelius & J. M. Gili (eds.) Advances in hydrozoan biology) **60**(1): 211-220.
- Lobanov, A. L., Dianov, Mikhail B., Kirejtshuk, Alexander G. & Vakhitov, A. T. 2005. Сравнение характеристик интерактивных диагностических программ. <http://www.zin.ru/projects/webkey-x/keyscomp.htm>. [Published 2005-12, last retrieved 2007-04-20]
- Louhivuori, Mikko 1996. DELTA data format and pottery classification. DELTA Newsletter **12**: 5-6.
- Loy, Alexander; Lehner, Angelika; Lee, Natuschka; Adamczyk, Justyna; Meier, Harald; Ernst, Jens; Schleifer, Karl-Heinz & Wagner, Michael 2002. Oligonucleotide microarray for 16S rRNA gene-based detection of all recognized lineages of sulfate-reducing prokaryotes in the Environment. Applied and Environmental Microbiology **68**(10): 5064-5081.
- Ludwig, Wolfgang; Strunk, Oliver; Westram, Ralf; Richter, Lothar; Meier, Harald; Yadhukumar, Buchner, Arno; Lai, Tina; Steppi, Susanne; Jobb, Gangolf; Förster, Wolfram; Brettske, Igor; Gerber, Stefan; Ginhart, Anton W.; Gross, Oliver; Grumann, Silke; Hermann, Stefan; Jost, Ralf; König, Andreas; Liss, Thomas; Lüßmann, Ralph; May, Michael; Nonhoff, Björn; Reichel, Boris; Strehlow, Robert; Stamatakis, Alexan-

- dros; Stuckmann, Norbert; Vilbig, Alexander; Lenke, Michael; Ludwig, Thomas; Bode, Arndt & Schleifer, Karl-Heinz. 2004. ARB: a software environment for sequence data. *Nucleic Acids Research* **32** (4):1363-1371.
- Lyon, Marcus Ward, Jr. 1936. Mammals of Indiana. *American Midland Naturalist* **17** (1): 1-373.
- Macfarlane, T. D. 1993a. Utilities for DELTA data entry and taxon summary generation. *DELTA Newsletter* **8**: 11-12.
- Macfarlane, T. D. 1993b. Spotlight on DELTA features: The Confor directives Absolute Error and Percent Error. *DELTA Newsletter* **8**: 13.
- Maddison, Wayne P. & Maddison, David R. 2006. Mesquite: a modular system for evolutionary analysis. Version 1.12. <http://mesquiteproject.org>. [Last retrieved 2007-05-02]
- Maddison, David R.; Swofford, David L. & Maddison, Wayne P. 1997. NEXUS: An extensible file format for systematic information. *Systematic Biology* **46** (4): 590-621.
- Mandel, Theo 1997. The elements of user interface design. John Wiley: New York (USA).
- Manola, Frank & Miller, Eric (eds.) 2004. W3C RDF Primer. W3C Recommendation 10 February 2004. <http://www.w3.org/TR/2004/REC-rdf-primer-20040210>. [Last retrieved 2007-04-01]
- Maxted, Nigel; White, R. J. & Allkin, R. 1993. The automatic synthesis of descriptive data using the taxonomic hierarchy. *Taxon* **42**: 51-62.
- McGillicuddy, Dan 2005. Key to dietary supplements and related botanicals, Navikey 3.0. <http://www.virtualherbarium.org/dietarysupplements/NavikeyInfo.html>. [Last modified 2005-01-01, last retrieved 2007-04-16]
- McGuinness, Deborah L. & van Harmelen, Frank (eds.) 2004. W3C OWL Web Ontology Language. Overview. W3C Recommendation 10 February 2004. <http://www.w3.org/TR/2004/REC-owl-features-20040210>. [Last retrieved 2007-04-01]
- Meacham, C. 2007. The MEKA home page. <http://ucjeps.berkeley.edu/meacham/meka>. [Last retrieved 2007-05-02]
- Metcalf, Z. P. 1954. The construction of keys. *Systematic Zoology* **3**: 38-45.
- Meyke, Evgeniy 2004. Bio-Tools.Net TAXIS 3.5. <http://www.bio-tools.net>. [Last retrieved 2007-04-16 (update to version 3.5.3.35, dated 2005, was not retested)]
- MIDI 2007. Sherlock Microbial Identification Systems. <http://www.midi-inc.com/index.html>. [Last retrieved 2007-05-10]
- Morris, Robert A.; Asiedu, Jacob K.; Stevenson, Robert D. & Tang, Hua 2003. Semi-automatic polyclave key generation (Abstract). Taxonomic Databases Working Group Annual Meeting, 25-26 October 2003 Instituto Gulbenkian de Ciência, Oeiras, Lisboa, Portugal. http://www.tdwg.org/2003meet/paperabstracts/TDWG_2003_Papers_Robert_Morris_1.htm. [Last retrieved 2007-04-17]
- Morris, R.; Thiele, Kevin; Paterson, Trevor & Hagedorn, Gregor 2004. The problem of sex. <http://wiki.tdwg.org/twiki/bin/view/SDD/TheProblemOfSex>. [Last retrieved 2007-04-17]
- Morris, Robert A.; Asiedu, Jacob K.; Haber, William; SaintOurs, Fred; Stevenson, Robert D. & Tang, Hua 2007. Database-backed decision trees with application to biological informatics. *Journal of Intelligent Information Systems* (online version appeared, doi:10.1007/s10844-006-0029-5, print version to appear).
- Morse, L. E. 1974. Computer programs for specimen identification key construction, and description printing using taxonomic data matrices. Publications of the Museum of the Michigan State University (Biological Series) **5**: 1-128. East Lansing, Michigan.
- Morse, David R.; Tardivel, G. M. & Spicer, J. 1996. A comparison of the effectiveness of a dichotomous key and a multi-access key to woodlice. Technical Report 14-96, Computing Laboratory, University of Kent, Canterbury, UK, August 1996. <http://www.cs.kent.ac.uk/pubs/1996/44/index.html> [Available as postscript format, last retrieved 2007-04-21]
- Müller, H. J. (ed.) 1985. Bestimmung wirbelloser Tiere im Gelände. Bildtafeln für zoologische Bestimmungsübungen. Gustav Fischer Verlag Stuttgart.
- Munsell color charts for plant tissues 1977. Munsell color, 2nd ed. Macbeth Division of Kollmorgen Corporation: Baltimore, Maryland (USA).
- Munsell Soil Color Charts 1992. Macbeth Division of Kollmorgen Instruments Corporation, revised ed. Newburgh, New York, USA.
- Naiburg, Eric J. & Maksimchuk, Robert A. 2001. UML for database design. Addison-Wesley: Boston (USA). 300 pp.
- Naskrecki, Piotr 2007. MANTIS – a manager of taxonomic information and specimens. <http://140.247.119.138/mantis>. [Last updated 2007-01-07, last retrieved 2007-04-17. First version 1996.]
- National Herbarium of New South Wales 2007. PlantNET. Interactive dichotomous keys. http://plantnet.rbgsyd.nsw.gov.au/interactive_keys.htm. [Last retrieved 2007-05-10]
- Neubacher, Dieter & Rambold, Gerhard 2007a. NaviKey 4 – a Java applet for accessing descriptive data coded in DELTA format. <http://>

- www.navikey.net. [Last updated 2007-03-15, last retrieved 2007-04-17]
- Neubacher, Dieter & Rambold, Gerhard 2007b. DiversityNavigator® – a Java rich client for accessing biodiversity databases. <http://www.diversitynavigator.net>. [Software version 0.97.21 dated 2007-02-06, last retrieved 2007-05-07; authorship cited according to http://141.84.65.132/BSM-Mycology/Dali/DALIDataBase_DALI_Details.cfm?ListNumber=7449]
- Nimis, P. L. 2007. Strumenti per l'identificazione / Identification tools. <http://dbiodbs.univ.trieste.it/dryades/prova/id/h3.html>. [13 freely accessible keys, 8 further keys with restricted access; last retrieved 2007-04-10]
- Nimis, P. L.; Martellos, S. & Poldini, L. 2005a. Interactive guide to the plants of M. Valerio (Trieste). Illustrations by A. Moro. http://dbiodbs.univ.trieste.it/mval/mval_en.html. [FRIDA-based key; last retrieved 2007-04-10]
- Nimis, P. L.; Martellos, S. & Prosser, F. 2005b. The plants of Villa Welsperg (Paneveggio – Pale di S. Martino Natural Park, Province of Trento). Illustrations by A. Moro. http://dbiodbs.univ.trieste.it/paneveggio/welsperg_en.html. [FRIDA-based key; last retrieved 2007-04-10]
- Noda, Natsuko & Kishi, Tomoji 1999. On aspect-oriented design – applying “multi-dimensional separation of concerns” on designing quality attributes. <http://www.cs.ubc.ca/~murphy/multid-workshop-oopsla99/position-papers/ws09-noda.pdf>. [Last retrieved 2007-04-17]
- Oberdorfer, Erich 1983. Pflanzensoziologische Exkursionsflora. 5. Aufl. Eugen Ulmer Verlag: Stuttgart (Germany). 1051 pp.
- OConnor, Barry & Klimov, Pavel B. 2004a. Visual Key. http://insects.ummz.lsa.umich.edu/beemites/vk_bees/Visual_key.htm. [Created 2004-03-18, last modified 2004-04-27, last retrieved 2007-04-16]
- OConnor, Barry & Klimov, Pavel B. 2004b. Comparison between Visual Key and others interactive identification programs. http://insects.ummz.lsa.umich.edu/beemites/vk_bees/Comparison.htm. [Last modified 2004-04-28, last retrieved 2007-04-16]
- Ogden, E. C. 1943. The broad-leaved species of *Potamogeton* of North America, north of Mexico. *Rhodora* **45**: 57-105, 119-163, 171-124.
- OMG 2001. Unified Modeling Language Specification Version 1.4, September 2001. <http://www.omg.org/cgi-bin/apps/doc?formal/01-09-67.pdf>. [Last retrieved 2007-04-15]
- OMG 2003. Unified Modeling Language Specification March 2003 Version 1.5, formal/03-03-01. <http://www.omg.org/cgi-bin/apps/doc?formal/03-03-01.pdf>. [Last retrieved 2007-04-15]
- OpenKey 2003. OpenKey project plant description XML schema, ver. 2.20. http://www.ibiblio.org/openkey/schema/plant_description_2_20.xsd. [Page last modified 2003-04-30, last retrieved 2007-04-15; also in conceptual form at: http://www.ibiblio.org/openkey/schema/Plant_Description_Table2.20.html, last updated 2003-04-17]
- Osborne, D. V. 1963. Some aspects of the theory of dichotomous keys. *New Phytologist* **62**: 144-160.
- Page, Roderic D. M. 1993. COMPONENT: Tree comparison software for Microsoft Windows, version 2.0. The Natural History Museum: London (UK).
- Page, Roderic D. M. 2001a. NDE. NEXUS Data Editor for Windows. <http://taxonomy.zoology.gla.ac.uk/rod/NDE/nde.html>. [Last modified 2001-09-13, last retrieved 2007-04-15]
- Page, Roderic D. M. 2001b. COMPONENT. <http://taxonomy.zoology.gla.ac.uk/rod/cpw.html>. [Last retrieved 2007-04-17; Free download of Component version 2.0.]
- Page, Roderic D. M. 2004. Phyloinformatics: Towards a phylogenetic database. *In*: Wang, Jason, et al. (eds) Data mining in bioinformatics. pp. 219-241.
- Palm, G. & Dietrich, C. 2001. Automated Identification of Bioacoustic Signals. BIOLOG Statusseminar, Bonn, <http://www.bgbm.fu-berlin.de/BioDivInf/biolog/Statusseminar1/20011206-B1-Palm-DORSA.pdf>. [Last retrieved 2007-04-17; slides of talk held at the meeting; there is no corresponding abstract.]
- Pankhurst, R. J. 1970a. A computer program for generating diagnostic keys. *Computer Journal* **13**: 145-151.
- Pankhurst, R. J. 1970b. Key generation by computer. *Nature* **227**: 1269-1270.
- Pankhurst, R. J. 1983. An improved algorithm for finding diagnostic taxonomic descriptions. *Mathematical Biosciences* **65**: 209-218.
- Pankhurst, R. J. 1988. An interactive program for the construction of identification keys. *Taxon* **37**: 747-755.
- Pankhurst, R. J. 1991. Practical taxonomic computing. Cambridge University Press: Cambridge (UK). 202 pp.
- Pankhurst, R. J. 1993a. Principles and problems of identification. *In*: Fortuner, R. (ed.) Advances in computer methods for systematic biology. John Hopkins Univ. Press: Baltimore, USA: 125-136.
- Pankhurst, R. J. 1993b. Taxonomic databases: the Pandora system. *In*: Fortuner, R. (ed.) Advances in computer methods for systematic biology.

- gy. John Hopkins Univ. Press: Baltimore, USA: 229-240.
- Pankhurst, R. J. 1993c. Comments on new features for the DELTA system. *DELTA Newsletter* **9**: 14-15.
- Pankhurst, R. J. 1998. A historical review of identification by computer. *In*: Bridge, P.; Jeffries, P.; Morse, D. R. & Scott, P. R. (eds.). *Information technology, plant pathology and biodiversity*. CAB International: Wallingford, UK: 289-303.
- Pankhurst, R. J. 2003. PANKEY – Programs for the identification and description of plants of animals. Exeter Software (47 Route 25A, Suite 2, Setauket, New York 11733-2870). <http://www.exetersoftware.com/cat/pankey/Pansheet.pdf>. (The content of the file is undated, but as of 2007-04-16 the pdf file was dated 2003-01-31. See also <http://www.exetersoftware.com/cat/pankey/pankey.html>; the usually cited link <http://www.rbge.org.uk/research/Pankey.htm> is no longer available since at least Jan. 2004).
- Pankhurst, R. J. & Pullan, Martin R. 1996. DELTA in PANDORA. *DELTA Newsletter* **12**: 16-20.
- Pankhurst, R. J. & Pullan, Martin 1998. The PANDORA taxonomic database system. <http://www.ibiblio.org/pub/academic/biology/ecology+evolution/software/pandora>. [Last retrieved 2007-05-01. Page is formally authored by Una Smith, 1999-2002, but considered here a formal error as citation for Pandora. The widely quoted link: <http://www.rbge.org.uk/research/Pandora.htm> is as of Jan. 2004 no longer active.]
- Paterson, T. 2004. The Prometheus Database Project: Capturing botanical descriptions for taxonomy. (Powerpoint presentation, DILS 2004). http://www.dcs.napier.ac.uk/~prometheus/prometheus_2/talk/DILStalkv2.ppt. [Last retrieved 2007-05-01]
- Paterson, Trevor; Kennedy, Jessie B.; Pullan, Martin R.; Cannon, Alan; Armstrong, Kate; Watson, Mark F.; Raguenaud, Cédric; McDonald, Sarah & Russell, Gordon 2004. A universal character model and ontology of defined terms for taxonomic description. *In*: Rahm, Erhard (ed.); *Data integration in the life sciences DILS*, Leipzig, Germany, March 25-26 2004, Proceedings; Springer Lecture Notes in Bioinformatics (Springer, Berlin) Vol. 2994: 63-78. Abstract: <http://springerlink.metapress.com/openurl.asp?genre=article&issn=0302-9743&volume=2994&spage=63>; preprint: http://www.dcs.napier.ac.uk/~prometheus/prometheus_2/publications/Paterson_etal_DILS.pdf. [Last retrieved 2007-05-01]
- Payne, R. W. & Preece, D. A. 1977. Incorporating checks against observer error into identification keys. *New Phytologist* **79**: 201-207
- Pennisi, Elizabeth 1994. Name that fly: Computers help make species identification child's play. *Science News* **145** (7): 108-111.
- Percudani, Riccardo; Rivetti, Claudio & Zambonelli, Alessandra 2006. Tuberkey. <http://www.truffle.org/tuberkey/tuberkey-english.html>. [File date 2006-05-09; last retrieved 2007-05-01]
- Pickering, John 2007. Discover life. <http://www.discoverlife.org>. [Last retrieved 2007-05-10]
- Piepenbring, Meike; Hou, C.-L.; Hagedorn, Gregor; Deml, Günther & Oberwinkler, F. 2001. GLOPP – Smut fungi. BIOLOG German Programme on Biodiversity and Global Change (Phase I, 2000-2004) Funded by BMBF, Status Report 2001, 218-219.
- Piepenbring, Meike; Hagedorn, Gregor; Deml, Günther; Hou, C.-L. & Oberwinkler, F. 2003. Global Information System for the Biodiversity of Smut fungi (Ustilaginales s. l. & Microbotryales, Basidiomycota). <http://www.DiversityCampus.net/Glopp/Smut/>. [Last retrieved 2007-05-01]
- Prillinger, H.; Oberwinkler, F.; Umile, C.; Tlachac, K.; Bauer, R.; Dörfler, C. & Taufrazthofer, E. 1993. Analysis of cell wall carbohydrates (neutral sugars) from ascomycetous and basidiomycetous yeasts with and without derivatization. *Journal of General and Applied Microbiology* **39**: 1-34.
- Pujar, Anuradha; Jaiswal, Pankaj; Kellogg, Elizabeth A.; Ilic, Katica; Vincent, Leszek; Avraham, Shulamit; Stevens, Peter; Zapata, Felipe; Reiser, Leonore; Rhee, Seung Y.; Sachs, Martin M.; Schaeffer, Mary; Stein, Lincoln; Ware, Doreen & McCouch, Susan 2006. Whole plant growth stage ontology for Angiosperms and its application in plant biology. *Plant Physiology* **142**: 414-428. <http://www.pubmedcentral.nih.gov/picrender.fcgi?artid=1586063&blobtype=pdf>.
- Pullan, Martin R.; Watson, Mark F.; Kennedy, Jessie B.; Raguenaud, Cédric & Hyam, Roger 2000. The Prometheus taxonomic model: a practical approach to representing multiple classifications. *Taxon* **49**: 55-75.
- Pullan, Martin R.; Armstrong, Kate E.; Paterson, Trevor; Cannon, Alan; Kennedy, Jessie B.; Watson, Mark F.; McDonald, Sarah & Raguenaud, Cédric 2005. The Prometheus Description Model: an examination of the taxonomic description-building process and its representation. *Taxon* **54** (3): 751-765.

- Purvis, Andy & Hector, Andy 2000. Getting the measure of biodiversity. *Nature* **405** (6783): 212-219.
- Radford, A. E.; Dickison, W. C.; Massey, J. R. & Bell, C. R. 1974. *Vascular Plant Systematics*. Harper & Row: New York (USA).
- Raguenaud, Cédric; Pullan, Martin; Watson, M.; Kennedy, Jessie; Newman, M. & Barclay, P. 2002. Implementation of the Prometheus Taxonomic Model: a comparison of database systems. *Taxon* **51** (1): 131-142.
- Rambold, Gerhard 1997. LIAS – the concept of an identification system for lichenised and lichenicolous ascomycetes. – *In*: Türk, R. & Zorer, R. (eds.), *Progress and problems in lichenology in the Nineties – IAL 3. – Bibliotheca Lichenologica* **68**: 67-72.
- Rambold, Gerhard 2002. Computer-aided identification systems for biology, with particular reference to lichens (chapter 31). *In*: Kranner, I.; Beckett, R. P. & Varma, A. K. (eds.) *Protocols in lichenology: culturing, biochemistry, ecophysiology and use in biomonitoring*. Springer: Berlin, Heidelberg (Germany). pp. 536-553.
- Rambold, Gerhard & Hagedorn, Gregor 1998. The distribution of selected diagnostic characters in the Lecanorales. *Lichenologist* **30** (4-5): 473-487.
- Rambold, Gerhard; Triebel, Dagmar 2007. Genera of lichenized and lichenicolous Ascomycetes. LIAS. A global information system for lichenized and non-lichenized Ascomycetes. www.lias.net/Taxa/DataForms/genera/index.html. [First version 1996, last retrieved 2007-04-15]
- Rambold, Gerhard; Hagedorn, Gregor; Begerow, D. & Weiss, Markus 2003. The Diversity Workbench Modules within the framework of the German GBIF Node for Mycology [Poster in GBIF-D: Fungi & lichens]. *In*: International Symposium “Sustainable use and conservation of biological diversity – A challenge for society.” Symposium Report Part A. 1-4 December 2003, Berlin, p. 438.
- Rayner, R. W. 1970. *A mycological colour chart*. Commonwealth Mycological Institute and British Mycological Society, Kew, England.
- Richards, A. J. 1986. *Plant breeding systems*. Allen & Unwin, London.
- Ridgway, Robert 1912. *Color Standards and Color Nomenclature*. Washington, D.C.; published privately by the author. 43 pp + 53 color plates.
- Rivetti, Claudio 1999. WebDelta – Web-based DELTA interface. <http://www.truffle.org/webdelta/webdelta.html>; <http://alice.bio.unipr.it/download/WebDelta1.2.tar>. [Last retrieved 2007-05-01]
- Rohlf, F. J. 1993. Feature extraction in systematic biology. *In*: Fortuner, R. (ed.) *Advances in computer methods for systematic biology*. John Hopkins Univ. Press: Baltimore, USA: 375-392.
- Rohlf, F. J. 1996. Morphometric spaces, shape components and the effects of linear transformations. *In*: L. F. Marcus et al. (eds.), *Advances in morphometrics* **284**: 117-129. Plenum Publishing: New York (USA).
- Rothmaler, W.; Schubert, R., Jäger, E. & Werner, K. 1985. *Exkursionsflora für die Gebiete der DDR und BRD. Band 3. Atlas der Gefäßpflanzen*. 6. Aufl. Verl. Volk und Wissen: Berlin (Germany).
- Ryan, B. D.; Bungartz, F.; Hagedorn, Gregor & Rambold, Gerhard 2005. LIAS glossary – a Wiki-based online dictionary for ascomycete terminology. <http://glossary.lias.net>. [Last retrieved 2007-05-01]
- Saarenmaa, H. 1999. *The Global Biodiversity Information Facility: Architectural and implementation issues*. European Environment Agency, Technical Reports 34. 34 pp. Copenhagen. Online: version 15. http://reports.eea.eu.int/Technical_report_No_34. [Last retrieved 2007-05-01. Originally at <http://www.eionet.eu.int/gbif/gbif-implementation-latest.html>, no longer available.]
- Saarenmaa, H. 2002. Technological opportunities and challenges in building a global biological information infrastructure. *In*: Saarenmaa, H. & Nielsen, E. S. (eds.) *Towards a global biological information infrastructure: Challenges, opportunities, synergies, and the role of entomology*. XXI International Congress of Entomology in Iguassu Falls, Brazil, on 24 Aug. 2000. Technical report No 70, European Environment Agency: Copenhagen (Denmark). 72 pp. http://reports.eea.eu.int/technical_report_2001_70. [Last retrieved 2007-05-01]
- Sautter, G.; Agosti, D. & Böhm, K. 2007. Semi-automated XML markup of biosystematics legacy literature with the GoldenGATE editor. *In*: *Proceedings of Pacific Symposium on Bio-computing*, Wailea, HI, USA **12**: 391-402. <http://psb.stanford.edu/psb-online/proceedings/psb07/sautter.pdf>. [Last retrieved 2007-04-13]
- Schalk, Peter H. & Heijman, Rob P. 1996. ETI's Taxonomic Linnaeus II Software: A new tool for interactive education. *UniServe Science News*, University of Sydney 1996 (3): 7-8.
- Schell, Spencer; Lockwood, Jeff; Schell, Scott & Zimmerman, Kiana (not dated, seen 2004). *Grasshoppers of Wyoming and the West. Field Guide to Common Western Grasshoppers. List of Species Fact Sheets*. <http://www.sdvc>.

- uwoy.edu/grasshopper/facttoc.htm. [Last retrieved 2007-05-10]
- Schilowa, Barbara (undated). 614 Bäume sicher erkennen. Interaktive Bestimmungsschlüssel. dialobis edition Berlin. ISBN: 978-3-9805520-4-2. DVD Box.
- Schmeil, O. & Fitschen, J. 1988. Flora von Deutschland und seinen angrenzenden Gebieten. 88. Aufl. Quelle & Meyer: Heidelberg (Germany).
- Schmeil, O.; Fitschen, J. & Seibold, S. 2006. Flora von Deutschland und angrenzender Länder. 93. Aufl. Quelle & Meyer: Heidelberg (Germany).
- Schuster, Rudolf M. 1958. Keys to the orders, families and genera of Hepaticae of America north of Mexico. *The Bryologist* 61 (1): 1-66.
- Schweigkofler W.; Lopandic K.; Molnár O.; Prillinger H. 2002. Analysis of phylogenetic relationships among Ascomycota with yeast phases using ribosomal DNA sequences and cell wall sugars. *Organisms Diversity & Evolution* 2 (1): 1-17.
- Seethalakshmi, K. K. & Muktesh Kumar, M. S. 1998. Bamboos of India. A compendium. INBAR Technical Report 17. 342 pp. [Online: <http://www.inbar.int/publication/txt/tr17>, last retrieved 2007-04-28]
- Seybold, S.; Koltzenburg, M. & Zauner, G. (eds.) 2001. Schmeil-Fitschen interaktiv. Die umfassende Bestimmungs- und Informationsdatenbank der Pflanzenwelt Deutschlands und angrenzender Länder. CD ROM. Quelle & Meyer. Wiebelsheim.
- Seybold, S.; Koltzenburg, M. & Zauner, G. (eds.) 2004. Schmeil-Fitschen interaktiv. Die Flora von Deutschland und angrenzender Länder. CD ROM, 2. Aufl. Quelle & Meyer. Wiebelsheim.
- Shattuck, S. & Fitzsimmons, N. 2000. BioLink: The Biodiversity Information Management System (software and documentation). CSIRO Publishing, Collingwood, Victoria, Australia.
- Smets, E. & Laboratory of Plant Systematics. 2003. Morphocode. <http://www.kuleuven.be/bio/sys/mc>. [File last modified 2003-09-23, last retrieved 2007-04-28]
- Smith, Marion R. 1943. A generic and subgeneric synopsis of the male ants of the United States. *American Midland Naturalist* 30 (2): 273-321.
- Sokal, R. R. & Rohlf, F. J. 1981. Biometry. 2nd ed. W. H. Freeman: New York (USA). 859 pp.
- Stanford Encyclopedia of Philosophy. 2003. Modal Logic. <http://plato.stanford.edu/entries/logic-modal>. [Last retrieved 2007-04-20]
- Steinhage, V.; Arbuckle, T.; Schröder, S.; Cremers, A. B. & Wittmann, D. 2001. ABIS: Automated Identification of Bee Species, BIOLOG Workshop, Dec. 5-7, 2001, Bonn. German Programme on Biodiversity and Global Change, Status Report 2001. German Ministry of Education and Research (BMBF), Bonn: 194-195.
- Stevens, P. F. 1991. Character states, morphological variation, and phylogenetic analysis: a review. *Systematic Botany* 16: 553-583.
- Stevenson, R. D.; Haber, W. A. & Morris, R. A. 2003. Electronic field guides and user communities in the eco-informatics revolution. *Conservation Ecology* 7 (1): 3. [Online: <http://www.consecol.org/vol7/iss1/art3>, last retrieved 2007-05-09]
- Sutton, B. C. 1980. *The Coelomycetes*. CAB: Kew, Surrey (UK). 696 pp.
- Swofford, D. L. 1990. PAUP – Phylogenetic analysis using parsimony. Natural History Survey: Champaign, Illinois (USA).
- Swofford, D. L. 2000. PAUP*. Phylogenetic analysis using parsimony (*and other methods). Version 4. Sinauer Associates, Sunderland, Massachusetts (USA).
- SysTax 2004. SysTax – a database system for systematics and taxonomy. ER-diagram documentation, interfaces. <http://www.biologie.uni-ulm.de/systax/documentation>. [Multiple pages, last modified 2004-01 to 2004-06; last retrieved 2007-04-28]
- Taylor, Andrew J. 1995. Extracting knowledge from biological descriptions. 2nd international conference on building and sharing very large-scale knowledge bases, Amsterdam, April 1995, IOS Press, Enschede (The Netherlands): 114-119. [Online: http://www.cse.unsw.edu.au/~andrewt/papers/nlp_vlkb95/nlp_vlkb95.html, last retrieved 2007-05-09]
- TDWG 1998. International Working Group on Taxonomic Databases (TDWG). A workshop on metadata and interoperability in biodiversity data systems. Report of the 1998 Annual meeting at the Centre for Plant Diversity & Systematics, University of Reading, UK, September 12-15, 1998. http://www.nhm.ac.uk/hosted_sites/tdwg/nwsltr_p3.html. [Last retrieved 2007-05-09. Originally at http://www.tdwg.org/nwsltr_p3.html, now forwarded.]
- TDWG 1999a. TDWG Newsletter – IUBS Taxonomic Databases Working Group Number 9, March 1999. http://www.nhm.ac.uk/hosted_sites/tdwg/newsletter.html [Last retrieved 2007-05-09. Originally at <http://www.tdwg.org/newsletter.html>, now forwarded.]
- TDWG 1999b. International Working Group on Taxonomic Databases: 1999 TDWG Meeting, Harvard Herbarium, Cambridge, USA, 29th-31st October, 1999. http://www.nhm.ac.uk/hosted_sites/tdwg/rep1999.html. [Last retrieved

- 2007-05-09. Originally at <http://www.tdwg.org/rep1999.html>, now forwarded.]
- Théry, Marc; Stevens, Albert-D.; Hoppe, Jürgen R.; Charles-Dominique, Pierre & Schuchmann, Karl-L. 1998. Angiosperm pollination and seed dispersal, a review. *Ecotropica* **4** (1-2): 69-91.
- Thiele, Kevin 1993. The holy grail of the perfect character: the cladistic treatment of morphometric data. *Cladistics* **9**: 275-304.
- Thiele, Kevin 2003. SDD part 0: Introduction and primer to the SDD standard. Version "3. Dec. 2003". <http://www.DiversityCampus.net/Projects/TDWG-SDD/Primer/index.htm>. [Last retrieved 2007-03-31]
- Thiele, Kevin & Sharp, Donovan 2006. SDD part 0: Introduction and primer to the SDD standard. <http://wiki.tdwg.org/wiki/bin/view/SDD/Primer/WebHome>. [Last retrieved 2007-03-31]
- Thiele, Kevin T.; Rutter, G. & Yeates, D. K. 1998. LucID Professional interactive key software system version 1. The Cooperative Research Centre for Tropical Pest Management, Brisbane.
- Thompson, F. C. 1996. Names: The keys to Biodiversity. *In*: Reaka-Kudla, M. L.; Wilson, D. E. & Wilson, E. O. (eds.), *Biodiversity II*. J. Henry Press, Washington (USA), 199-211.
- Thompson, Henry S.; Beech, David; Maloney, Murray & Mendelsohn, Noah (eds.) 2001. W3C XML Schema Part 1: Structures. W3C Recommendation, 2 May 2001. <http://www.w3.org/TR/2001/REC-xmlschema-1-20010502>. [Last retrieved 2007-04-01]
- Tiwari, S. & Gallager, S. 2003. Identification of bivalve larvae using multiscale texture and color invariants. Technical report, Woods Hole Oceanographic Institution. <http://4dgeo.whoi.edu/lihdatt/waveletpaper.pdf>. [Last retrieved 2007-03-31]
- Trappe, J. M. 1982. Synoptic keys to the genera and species of zygomycetous mycorrhizal fungi. *Phytopathology* **72**: 1102-1108.
- Tree of Life web project 2007. About the Tree of Life web project. <http://tolweb.org/tree/home.pages/abouttol.html>. [Last retrieved 2007-05-02]
- UBio 2004. Tools – X:ID. <http://www.ubio.org/index.php?pagename=XID/key>. [Last retrieved 2007-04-16]
- Vanel, J. M. 2004. Data preparation. Worldwide botanical knowledge base project. <http://wwbota.free.fr/project/data/data-processing.html>. [Last update 2004-08-16, last retrieved 2007-04-13]
- Voss, E. G. 1952. The history of keys and phylogenetic trees in systematic biology. *Journal of the Scientific Laboratories of Denison University* **43**: 1-25.
- W3C 2007. RDF: Resource Description Framework. <http://www.w3.org/RDF>. [Last retrieved 2007-04-01]
- Watson, Ian (ed.) 2002. *Applying Knowledge Management: Techniques for Building Corporate Memories*. Elsevier: Amsterdam (NL). 250 pp.
- Watson, L. & Dallwitz, Mike J. 1991. The families of angiosperms: Automated descriptions, with interactive identification and information retrieval. *Australian Systematic Botany* **4** (4): 681-695.
- Weitzman, Anna L. & Lyal, Christopher H. C. 2005. An XML schema for taxonomic literature – taXMLit. <http://www.sil.si.edu/digitalcollections/bca/documentation/taXMLitv1-3Intro.pdf>. 23 pp. [File date 2005-10-28, last retrieved 2007-05-13]
- White, Helen 1994. Data or concepts – what should we be coding? *DELTA Newsletter* **10**: 13-14.
- White, I. M. & Sandlant, G. R. 1998. Computerised insect identification: a comparison of differing approaches and problems. *In*: Bridge, P.; Jeffries, P.; Morse, D. R. & Scott, P. R. (eds.). *Information technology, plant pathology and biodiversity*. CAB International: Wallingford, UK: 261-271.
- White, I. M. & Scott, P. R. 1994. Computerized information resources for pest identification: A review. *In*: Hawksworth, D. L. (ed.) *Identification and characterisation of pest organisms*, CAB International (UK): 129-137.
- White, R. J.; Allkin, R. & Winfield, P. J. 1993. Systematic databases: The BAOBAB design and the ALICE system. *In*: Fortuner, R. (ed.) *Advances in computer methods for systematic biology*. John Hopkins Univ. Press: Baltimore, USA: 297-311.
- Wilson, Nathan 1994. Identifying organisms with computers: an implementation of a computerized synoptic identification system with fungi as a test case. Master Thesis, University of California: Santa Cruz (USA). Online: <http://collectivesource.com/taxy/thesis.html>. [Last retrieved 2007-05-10]
- Wilson, E. O. 2003. Biodiversity in the information age. *Issues in Science and Technology Online*, Summer 2003. <http://www.issues.org/19.4/wilson.html>. [Last retrieved 2007-05-10]
- Winder, L.; Lefley, M. & Smith, B. 1997. A key for freshwater invertebrates using fuzzy logic. *Bioinformatics* **13**: 169-174.
- Wright, J. F.; Morse, David R. & Tardivel, G. M. 1995. An investigation into the use of hypertext

- as a user-interface to taxonomic keys. *Computer Applications in the Biosciences* **11**: 19-27.
- XID Services 2007. XIDServices, Inc. Expert identification system. <http://www.xidservices.com>. [Last retrieved 2007-04-13]
- Yoon, N. & Rose, J. 2001. An information model for the representation of multiple biological classifications. *In*: Alexandrov, V. N.; Dongarra, J. J.; Juliano, B. A.; Renner, R. S.; Tan, C. J. K. (eds.) *Computational Science. ICCS 2001: International Conference, San Francisco, CA, USA, May 2001 Proceedings, Part 1*. Springer: New York (USA), 937-946.
- Ytow, N.; Morse, David R. & Roberts, D. 2001. Nomenclurator: a nomenclatural history model to handle multiple taxonomic views. *Biological Journal of the Linnean Society* **73** (1): 81-98.
- Zar, J. H. 1984. *Biostatistical Analysis*. 2nd ed. Prentice-Hall: New Jersey (USA). 718 pp.
- Zhong, Y; Jung, S.; Pramanik, S. & Beaman, J. H. 1996. Data model and comparison query methods for interacting classifications in taxonomic databases. *Taxon* **45**: 223-241.

10. Appendix

10.1. Brief history of SDD

At the TDWG meeting 1998 the author of this thesis presented an analysis showing that the proposed “New DELTA” (see above) would break compatibility with previous DELTA versions. It was proposed that TDWG would initiate a new standard process rather than automatically accepting fundamental changes in the existing DELTA standard (TDWG 1998, TDWG 1999a). At the next TDWG meeting in Harvard many of the major developers of descriptive databases (TDWG 1999b) convened and intensive discussions ensued in person and on an email discussion group started after the meeting. Since then, the SDD group consists of a small number of core members and a large number of occasional participants and informants. Substantial progress has been made in recent years during face-to-face meetings. The activities of the SDD group are documented in a publicly accessible Wiki (<http://wiki.tdwg.org/twiki/bin/view/SDD/>), an archived email list, and in the following reports and minutes of the working sessions: Harvard, USA (TDWG 1999b, Hagedorn 1999b); Frankfurt, Germany (TDWG meeting, Hagedorn 2000b); Sydney, Australia (TDWG meeting, Hagedorn 2001b); Canberra and Sydney, Australia (SDD meetings, Hagedorn 2002b); Indaiatuba, Brazil (convener's report: Hagedorn 2002c; minutes: Hagedorn 2003a); Paris, France (SDD meetings, Hagedorn 2003b); Oeiras/Lisbon, Portugal (TDWG meeting, minutes: Hagedorn 2003c, schema v. 0.9: Hagedorn 2003d, and documentation: Hagedorn 2003e); Christchurch, New Zealand (TDWG meeting, Hagedorn 2004c, 2004d).

The latest developments were the release of the SDD w3c XML schema in version 1.0 and its approval as an international data exchange standard at the TDWG meeting in St. Petersburg, Russia). In response to experience with the schema and new requirements by GBIF on a further meeting (Berlin, Germany, Hagedorn 2006), a new version was developed and as of March 2007 – after two release candidates – released as version 1.1.

10.2. Code fragments for evaluating character applicability rules

This appendix first provides a procedural programming fragment and then code in SQL to evaluate character applicability rules as discussed in the chapter “Character applicability rules” (p. 76).

Procedural pseudo-code for character applicability

Because of the multitude of incompatible programming languages, it is considered more generally useful to express the algorithm in a verbose pseudo-code language, intended for communication with human programmers. It is similar to easily readable programming languages such as SQL, Pascal, Delphi, Modula or many Basic dialects, but it is hoped that the code below can be understood by programmers preferring other languages (C, Perl, Java, etc.) as well. Variable names are italicized, comments enclosed in “/* */”.

The following code fragment deals with the evaluation of multiple controlling characters, of a controlling character being unknown, inapplicable, or having data, and in the latter case with the evaluation of applicable-if and inapplicable-if rules. Not included is code to test whether the controlling character is made inapplicable through an “inapplicable” coding status value that is not combined with other character data. This has been purposely omitted because additional data structures for the coding status would have to be introduced.

```
Function IsApplicable(ControlledChar as CharacterDefinition,
                    Description as Collection of CharacterData) as Boolean
Define b as Boolean = True;
For Each ControllingCharData In Description
  /* Multiple controlling chars are combined with AND;
     any one may make the controlled character inapplicable */
  b = b And IsApplicableFor1(ControlledChar, ControllingCharData);
End For Each;
Return b;
End Function;

Function IsApplicableFor1(ControlledChar as CharacterDefinition,
                        ControllingCharData as CharacterData) as Boolean
Define b as Boolean;
If Not IsApplicable(ControllingCharData.Character,
                  ControllingCharData.Description) Then
  /* Note: a) this assumes some back-pointers being passed with
     character data; ControllingCharacter and Description could also
     be passed as parameters
     b) this is a highly inefficient recursion, see comment after code... */
  Return False;
Else If ControllingCharData.Count=0 Then
  b = True; /* no scored states present in controlling character data is
     considered as applicable; alternatively one might consider unknown
     controlling character as unknown applicability) */
Else
  b = False; /* char. is inapplicable if no state makes it applicable */
  For Each ScoredState in ControllingCharData
    /* i. e. for all states for a character in a description */
    b = b Or StateMakesApplicable(ControlledChar, ScoredState);
  End For Each; /* Scored states within controlling char. combined with OR */
  Return b;
End If;
End Function;

Function StateMakesApplicable(ControlledChar as CharacterDefinition,
                             ScoredState as StateData) as Boolean
/* ScoredState is a data set wide unique object, not a local identifier
   within a character as the state numbers are in DELTA */
Define b as Boolean = True; /* Default if no rule exists at all */
For Each Rule in ApplicabilityRules
  /* ApplicabilityRules here is assumed to be globally accessible.
     Assumes each controlling state/controlled character-combination
     is considered one rule object. Looping is inefficient, real code
     would use index or hash table or might organize Rules in a tree. */
  If Rule.ControllingState = ScoredState
    And Rule.ControlledChar = ControlledChar Then /* evaluate type of rule */
```

```

    If Rule.ApplicabilityRule Then /* type of rule = applicable-if */
      b = True; /* return true if scored state found in rules */
    Else /* type of rule = inapplicable-if */
      b = False; /* return false if scored state found in rules */
    End If;
  Exit For; /* Exit loop directly: only a single rule may exist! */
End If;
End For Each;
Return b;
End Function;

```

Clearly, the procedural pseudo-code shown above is rather inefficient. For each character that might be controlled it loops repeatedly through all states of controlling characters in a description, and further tests for cascading dependencies in a highly inefficient repeated recursion. A better strategy might be to first build a tree of controlling characters organized by an analysis of cascading dependencies (see p. 82), to avoid the additional recursion, organize the ApplicabilityRules by controlling characters, and then for each description store the results of the controlled characters in that tree. This would reduce the number of characters studied, would allow the reuse of the character data from the description if multiple characters are controlled, and would prevent recursion. As soon as a controlled character becomes inapplicable in this tree, all other controlled characters would automatically become inapplicable.

SQL code for character applicability in relational databases

One may define two entities, Description and CharApplicabilityRule (Table 61):

Table 61. Example data to illustrate SQL code for character applicability rules.

Description			CharApplicabilityRule (for Inapplicable-if)			
DescrID	CharID	ScoredState	Controlling CharID	Controlling State	Controlled CharID	IsAppli- cable
1	1	1.a				
1	1	1.b	1	1.a	2	False
2	1	1.a	1	1.b	2	False
3	1	1.b	1	1.a	9	True
4	1	1.a	1	1.b	9	True
4	1	1.b				
4	1	1.c				
5	1	1.b				
5	1	1.c				
6	1	1.a				
6	1	1.c				
7	1	1.c				

IsApplicable = False and True for *inapplicable-if*, and *applicable-if* rules, respectively.

As discussed in “Analysis of convertibility” (p. 80), for each combination of a controlling and controlled character, the states defined in the character definition may be split into two non-overlapping subsets: those states making the controlled character applicable (*A*) and those making it inapplicable (*I*). The association of these with the controlling or non-controlling states depends on the kind of applicability rule (*applicable-if* or *inapplicable-if*).

Terminology approach. This approach may be useful if the status of all states (controlling and non-controlling) in the controlling character shall be analyzed or evaluated. It only deals with terminology information and does not evaluate the rules in combination with a given description yet. The terminology approach is useful when character applicability calculations shall be performed in programming code external to the DBMS and the necessary data for doing so shall be

obtained from the database in a compact, pre-processed form. For example, identification data may be stored not as “ad-hoc” descriptions in the database and the evaluation of character applicability based on these data would most likely occur outside the database.

Step T1: The query “*CharApplicability_T1ControlledControllingWithAllStates*” reduces CharApplicabilityRule to unique ControlledCharID/ControllingCharID combinations and joins with all character states defined for these character (obtained from a table CharState not shown above):

```
SELECT ControlledCharID, IsApplicable, ControllingCharID, CharState.StateID
FROM CharState INNER JOIN CharApplicabilityRule
  ON CharState.CharID = CharApplicabilityRule.ControllingCharID
GROUP BY ControlledCharID, IsApplicable, ControllingCharID, StateID;
```

Note that rather than defining a table with one record per controlling state as outlined above (and similar to the DiversityDescriptions model), an alternative ER-model might provide one table for unique controlling/controlled character combinations, and a related table for the controlling states. Advantages of that model would be that the constraint of a single kind of rule (applicable-if or inapplicable-if, expressed in the Boolean IsApplicable field) would be enforced by the model and that the group-by clause above could be omitted.

Step T2: In a second query (which may be called “*CharApplicability_T2ControlledControlling-AllStateStatus*”) the query from step 1 (which is given the short alias name “CCAllStates”) is joined with a left outer join again with the applicability rules (using the primary key, i. e., the controlling states plus controlled character). All records for which a rule can be joined belong to the controlling and all records for which the outer table contains Null values belong to non-controlling states. The association between controlling/non-controlling and the *A* and *I* subsets is obtained by referring back to the IsApplicable value from the first query. The calculation requires an if/else function, the name and syntax of which usually differs between SQL dialects. Here it is called Iif and has the syntax “Iif(Boolean-condition; result-if-true, result-if-false)”:

```
SELECT CCAllStates.*, Iif(ControllingState Is Null,
  Not CCAllStates.IsApplicable,
  CCAllStates.IsApplicable) AS MakesApplicable
FROM CharApplicability_T1ControlledControllingWithAllStates AS CCAllStates
LEFT JOIN CharApplicabilityRule
  ON (CCAllStates.ControlledCharID = CharApplicabilityRule.ControlledCharID)
  AND (CCAllStates.StateID = CharApplicabilityRule.ControllingState);
```

For those accustomed to a graphical query designer, the following screenshots representing the SQL in the MS Access qbe editor may be helpful (Figs. 232-233):

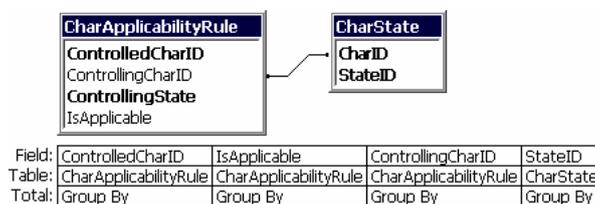


Figure 232. *CharApplicability_T1ControlledControllingWithAllStates* (step 1) in MS Access qbe-view. See text for equivalent SQL.



Figure 233. *CharApplicability_T2ControlledControllingAllStateStatus* (step 2) in MS Access qbe-view. See text for equivalent SQL.

B: Optimized approach to evaluate descriptions. In principle, the result of the previous query could be joined with the Description table and the character applicability rules evaluated. However, doing so might be inefficient. For each description only a subset of states need to be analyzed and the combination of an aggregation, an outer join, and a calculated value is likely to be in-transparent to the query optimizer of most DBMS. Therefore, a slightly more direct approach is presented in the following.

Step D1: The view “*CharApplicability_D1JoinDescription*” joins the applicability rules with descriptions. The join occurs on the controlling character rather than on the states; the result thus include both controlling states (those present in the applicability rule) and non-controlling states. The latter are essential since character applicability rules are evaluated based on both kinds of states. The description data must then be grouped by description, by controlling characters, and by controlled characters (multiple characters may be controlled by the same controlling states). The result is equivalent to step T1 above, except that the results are obtained for each description, and only include those states present (scored) in that description:

```
SELECT AppRule.ControlledCharID, AppRule.IsApplicable,
       AppRule.ControllingCharID, Description.DescrID, Description.ScoredState
FROM Description INNER JOIN CharApplicabilityRule AS AppRule
ON Description.CharID = AppRule.ControllingCharID;
```

Step D2: The view “*CharApplicability_D2HavingStatesMakingApplicable*” evaluates the applicability rules in part. It already abstracts *applicable/inapplicable-if* rules and groups by controlling/controlled character within description, thus abstracting from which of the states have contributed a result making a description/controlling/controlled character combination applicable:

```
SELECT D1.ControlledCharID, D1.ControllingCharID, D1.DescrID
FROM CharApplicability_D1JoinDescription AS D1
LEFT JOIN CharApplicabilityRule AS AppRule
ON (D1.ScoredState = AppRule.ControllingState)
AND (D1.ControlledCharID = AppRule.ControlledCharID)
WHERE (D1.IsApplicable=True AND AppRule.ControllingState Is Not Null)
OR (D1.IsApplicable=False AND AppRule.ControllingState Is Null)
GROUP BY D1.ControlledCharID, D1.ControllingCharID, D1.DescrID;
```

The strategy may be explained as follows: By joining the results from D1 again with CharApplicabilityRule in a left outer join specific to controlling state and controlled character one obtains D1 (identical) plus additional columns that are null if no rule (*applicable-if* or *inapplicable-if*) matches a character state in the description. From the possible conditions for applicable characters (compare p. 81), the condition $D \cap A \neq \emptyset$ is selected as most appropriate for testing in SQL (testing for $D \cap I \neq D$ or $D \cap I = D$ would involve a comparison of counts which is usually less efficient). The where condition performs the conversion of controlling/non-controlling states into A , i. e., the set of states making a character applicable. This conversion is the opposite for *applicable-if* and *inapplicable-if* rules. Finally, by grouping on a combination of description/controlling/controlled character multiple records containing states that make a controlled character applicable are reduced to a single record.

Step D3: The query “*CharApplicability_D3InapplicableChar*” performs two major tasks: it reverses the set of characters known to be applicable into those which must consequently be inapplicable. This reversal is specific to a combination of description and controlled and controlling character. Therefore, rather than basing the reversal on Description, it is again based on the view from step 1. It again constrains the result to those with empty intersection ($ControllingCharacters \cap ControllingCharactersEvaluatingToApplicable \neq \emptyset$) using a left outer join and requiring the right-hand side to be Null. Next, the evaluation of multiple characters controlling the same controlled character is performed by grouping only to Description and controlled character:

```
SELECT D1.DescrID, D1.ControlledCharID
FROM CharApplicability_D1JoinDescription AS D1
LEFT JOIN CharApplicability_D2HavingStatesMakingApplicable AS D2
ON (D1.ControlledCharID = D2.ControlledCharID)
AND (D1.ControllingCharID = D2.ControllingCharID)
AND (D1.DescrID = D2.DescrID)
WHERE D2.DescrID Is Null
GROUP BY D1.DescrID, D1.ControlledCharID;
```

Reviewing the sequence of queries, the strategy of first grouping for applicability of controlled characters within a description/controlling/controlled combination, and then grouping for inapplicability of controlled characters within a description/controlled combination is the central part of the strategy. As discussed in the main discussion of character applicability (p. 76) and also in the pseudo-code fragment above, the evaluation of these two cases is the opposite of each other.

If any state of a controlling character exists in a description that makes the controlled character applicable, the result is applicable. If any controlling character exists in a description that makes the controlled character inapplicable, the result is inapplicable. In SQL, “*or-ing*” between records using a combination of group-by and where clause is simpler than “*and-ing*” between records (which usually also requires a comparison of counts).

If the execution speed of the queries is problematic, one may want to test omitting the group-by clause (GROUP BY D1.ControlledCharID, D1.ControllingCharID, D1.DescrID) from “*CharApplicability_D2HavingStatesMakingApplicable*” with a data set of realistic size. The results of step 3 do not depend on an aggregation in step 2, because the aggregation in step 3 is stronger than the one in step 2. Depending on the query optimizer of the DBMS, omitting the group-by clause may be more or less effective.

Again, the following screenshots representing the SQL in the MS Access qbe editor may be helpful (Figs. 234-236):

Field:	ControlledCharID	IsApplicable	ControllingCharID	DescrID	ScoredState
Table:	CharApplicabilityRule	CharApplicabilityRule	CharApplicabilityRule	Description	Description
Show:	<input checked="" type="checkbox"/>				

Figure 234. *CharApplicability_D1JoinDescription* (step 1) in MS Access qbe-view. See text for equivalent SQL.

Field:	ControlledCharID	ControllingCharID	DescrID	IsApplicable	ControllingState
Table:	CharApplicability_D1JoinDescription	CharApplicability_D1JoinDescription	CharApplicability_D1JoinDescription	CharApplicability_D1JoinDescription	CharApplicabilityRule
Total:	Group By	Group By	Group By	Where	Where
Show:	<input checked="" type="checkbox"/>	<input checked="" type="checkbox"/>	<input checked="" type="checkbox"/>	<input type="checkbox"/>	<input type="checkbox"/>
Criteria:				True	Is Not Null
or:				False	Is Null

Figure 235. *CharApplicability_D2HavingStatesMakingApplicable* (step 2) in MS Access qbe-view. See text for equivalent SQL.

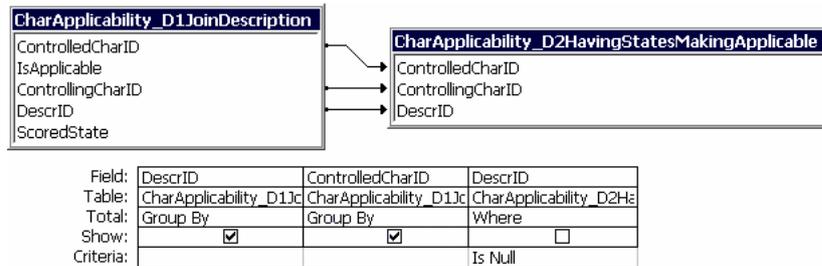


Figure 236. *CharApplicability_D3InapplicableChar* (step 3) in MS Access qbe-view. See text for equivalent SQL.

Notes: Both the terminology and the description SQL code shown deal with the evaluation of multiple controlling characters, of a controlling character being unknown or having data, and in the latter case with the evaluation of applicable-if and inapplicable-if rules.

Not included is the evaluation of exclusive inapplicable coding status values (those not accompanied by character data or other coding status values). An inapplicable controlling character results in inapplicable controlled characters. In a simple case this can be done by separately testing for characters that have no character data (character states) in a given description and where “inapplicable” is the only coding status value. Note that a combination of “inapplicable” with other coding status values (such as DELTA U or SDD values where data “Exist” or “May exist”; compare Table 16, p. 75) should be evaluated such that the controlling character is potentially applicable.

Furthermore, the issue of “Cascading character applicability rules” (p. 82) is not covered. A dynamic evaluation of cascading applicability rules is complicated because standard SQL lacks language constructs to handle recursions of unknown depth. Probably each database vendor provides methods to deal with this problem in stored procedures or using proprietary extensions of SQL such as provided by p-SQL in Oracle databases.

An attractive alternative approach might be to preprocess the character applicability rules, creating direct rules for each cascading indirect applicability rule. Such data may be stored in a second internal table, containing data similar to those resulting from the terminology approach (p. 390) and could be continuously updated through database triggers. Doing so not only makes the evaluation of cascading applicability rules highly efficient. Without sacrificing the management advantages of cascading definitions or the two forms of character applicability rules (*inapplicable-if* and *applicable-if*), the evaluation logic could then be based on simpler rules.

10.3. Preferred, alternative, and rejected terms for identification keys

The selection of certain terms like “computer-aided”, “interactive identification”, “branching” and “multi-access” keys in this thesis is not based on a general consensus, and a critical discussion of available alternative terms is necessary. Doing this has been deferred until now because it was desirable to first fully introduce the various aspects of keys.

“Interactive identification”

Many software programs (e. g., CSIRO Intkey, Visual Key, Navkey, DAP, DAWI, or Polly-Clave; but not CBIT Lucid) call themselves an “interactive key”. Previously, “interactive identification” was defined as identification processes involving an intensive interaction between human reasoning and knowledge and a – printed or digital – knowledge base, opposed to automatic identification (“Levels of interaction”, p. 230).

Printed multi-access keys (e. g., Fig. 129 on p. 250) do require a high degree of “interaction” and the free selection of characters is based on interactions with the reasoning and experience of the user very similar to that found in computer-aided multi-access keys. However, it is undisputed that computer-aided keys provide much richer options for interaction than printed keys. The choice of terminology in “Levels of interaction” could thus be revised in the future and the term “interactive” reserved for the degree of interaction typically found only in computer-aided keys. Unfortunately, in the context of “Levels of interaction”, “*interactive*” seems to be a more logical antonym to “*automatic*” than the possible alternative “*manual*”. The latter seems odd for an identification process involving a computer and human knowledge and reasoning.

The greater problem with the term “interactive key” is that, because the majority of interactive keys are multi-access keys, over time it seems to have become a synonym for such keys. In current usage, the term thus not only excludes *printed* but also computer-aided *branching* keys. This usage should be avoided. Branching keys on the web range from simple hyperlinked keys (e. g., Fig. 114 on p. 234) to quite complex presentations with hyperlinks to terminology or illustrations (e. g., Figs. 125-126 on p. 247), sometimes reminding more of a web application than of a hyperlinked sequence of web pages (e. g., Fig. 237, below).

As a consequence, use of “*interactive key*” or “*interactive identification*” in a general sense may well be continued. Although the software authors may often understand it in a more narrow sense, the software programs mentioned also fit a wider definition and the fact that computer-aided identification is being discussed is clear from the context. “Interactive identification” should be avoided when discussing aspects that are specific to multi-access keys and its use should be carefully considered when discussing aspects that depend on the use of computers. The term “**computer-aided identification**” is recommended when specifically discussing increased efficiency of identification processes by using a computer (regardless whether a branching or multi-access keys is being used). A suitable antonym may be “*identification using printed keys*”.

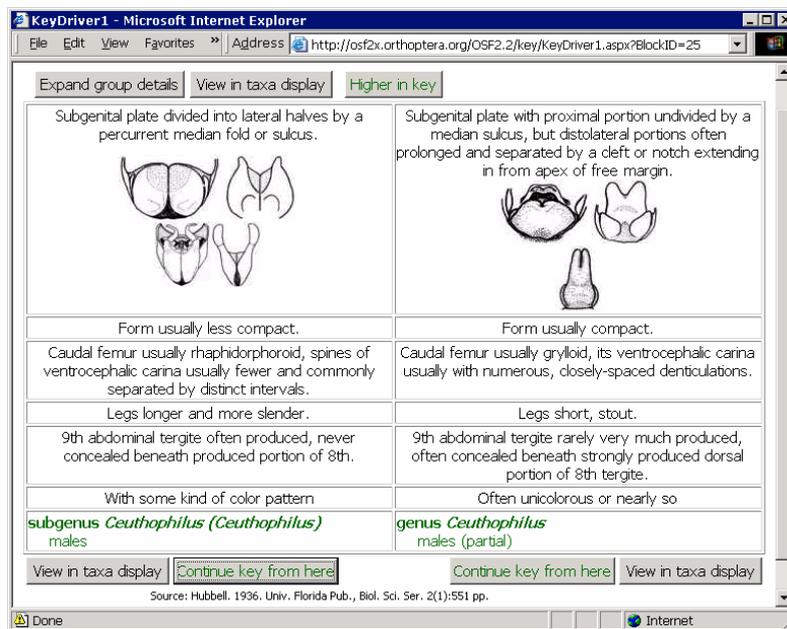


Figure 237. An example of an interactive dichotomous key, split into one question per page (key to the species of the genus *Ceuthophilus* Scudder; accessible from “Orthoptera Species File Online”, <http://osf2x.orthoptera.org>).

Branching keys

Available terms for keys that completely guide users through the identification process in a pre-defined sequence of questions are (based on discussions in the SDD group and an independent survey):

- **Dichotomous/polytomous key:** The first term is perhaps the most frequently used term. Its major disadvantage is that it is explicit about the number of choices in the key, which, especially in the context of computer-aided identification, becomes secondary in nature. The term polytomous key is usually understood to imply a key that is not dichotomous, and is only rarely used. See also p. 234 for use of “polychotomous”. To refer to the generalized concept of such keys, the combined term “Dichotomous/polytomous key” is sometimes used and probably generally understood, but are rather clumsy.
- **Bifurcating and multifurcating key:** this is a Latin equivalent of dichotomous/polytomous and only very rarely used (Google, 2005-04-13, 2 “bifurcating key”, 0 “multifurcating key”).
- **Branching key:** The term seems to be somewhat common in British usage. On the web several definitions can be found explaining the term as “a synonym of dichotomous key”. Somewhat contrary to the definition, the intuition behind the term does not exclude its use for polytomous keys. It is here considered the most intuitive alternative to the use of “dichotomous/polytomous key”.
- **Sequential key:** This seems to be another appropriate term, highlighting an essential feature of “dichotomous” or “polytomous” keys. It is currently very rarely used and could be found on the web only in two glossaries (B. W. Coad & D. E. McAllister: “Dictionary of Ichthyology”; www.briancoad.com/Dictionary/S.htm and www.fishbase.org/Glossary/Glossary.cfm?TermEnglish=sequential%20key), one botanical introduction to keys (“How to use keys”; http://protea.worldonline.co.za/key_how.htm), and one introductory lecture (by Tosak Seelanan; <http://www.sc.chula.ac.th/botany/eClass/2305313/Taxonomy.htm>).
- **Single-access key, single-entry key:** These terms are occasionally used as an intuitive antonym to multiple-access/multiple-entry key. They have the advantage of being logical antonyms when used in discussions of the relative advantages of these two key types – and the disadvantage of being probably not very intuitive when used alone.
- **Decision tree:** This term is used by Morris & al. 2007 and appropriately describes the use of identification keys. Strictly, branching keys with redirection are not trees but DAGs (p. 234), but this needs not be preventive of its adoption.
- **Pathway key:** This term is used in CBIT Lucid Phoenix (CBIT 2007b), a program specialized in digitizing or building dichotomous or polytomous keys. It is clear that the intended picture must be that of the potential paths to all results (regardless of technology, any successful identification will have a single path). Pathway key seems to be less illustrative of the structure of these potential paths than “branching key”. A possible emendation could be “*single-pathway key*”, which, in the case of keys with redirection or duplicate results, would be too strict.
- **Fixed path key:** The term “fixed” refers to the static nature of the sequence and number of user choices to reach an identification result. It was proposed during SDD discussions, but participants were split whether to consider it appropriate and intuitive. It is unclear what quality shall be expressed by the adjective “fixed”. In multi-access keys the *number* of available paths to a specific result is very large, whereas in branching keys only a single path (or a small number of paths in keys with redirection or duplicate results) exists. In a sense, however, each path is fixed (see “pathway key” above).
- **Predefined graph key:** An available appropriate term, but probably not very intuitive to most biologists.

Terms that should be avoided when referring to the *structure* of branching keys are:

- **Guided key:** this term was proposed by the author of the present work himself in previous SDD sessions. It is confusing because the intuition is not that the key is “guided”, but “guiding” the user. “Guiding key” would be possible, but because of the many forms of character guidance (p. 267) available in both branching and multi-access keys, it seems to be inappropriate.
- **Authored key or designed key:** Although the structure of classical dichotomous keys in taxonomic monographs is usually under the control of an author, keys with an equivalent structure may be the result of computer algorithms. *Authored key* is a useful term to denote this difference, but not to express the structural aspects of a branching key.
- **Analytical key** (e. g., Leenhouts 1966) and **Diagnostic key** (e. g., Pankhurst 1991): These equivalent terms are useful when talking about purposes, i. e., identification versus knowledge representation (“natural key” in Leenhouts 1966, or “synoptic key” in Pankhurst 1991). Due to the traditional dominance of branching keys, the terms are occasionally used as if restricted to this key method. The purposes of analysis and diagnosis may, however, be equally well fulfilled by multi-access keys (especially printed ones, see p. 249). These functional terms should therefore not be used to denote the structure of a key.

Deciding on the best term for branching keys is especially difficult. It may be noted that all of *branching*, *sequential*, or *decision tree* denote features that can also be found in multi-access keys, if the perspective is that of a single instance of *key-usage* rather than *key design* or *structure*. When using a multi-access key, a decision tree, creating a branching pattern in a given sequence is created. The true difference between a multi-access key and a branching key is that the sequence and branching of decisions is fixed by the author of the key, rather than selectable by the user.

Couplet

The set of two or more (in dichotomous or polytomous keys, respectively) statements or answering options in each step in a branching key are usually called a couplet. Strictly, in English language *couplet* and *couple* in most senses imply exactly two, which would make couplet applicable to dichotomous keys only. The unspecific sense “a small number, a few, several” (as in “a couple of oranges, a couple of species”) is listed, e. g., in CED (1992) only as sense “7b”. In contrast to other languages, for English speakers the term “couplet” may therefore be confusing and an alternative term desirable. A possible term might be “branchlets”. In this thesis the term “couplet” is upheld and used in the general sense as being inclusive of more than two alternative propositions.

Multi-access keys

Terms found in current usage for multi-access keys are:

- **Multiple-access key or multi-access key:** These terms are widely used and advocated, e. g., by Pankhurst (1991), Pankhurst (1998), and CBIT Lucid Phoenix (CBIT 2007b). It is accepted in the current thesis as well.
- **Multiple-entry key** (versus *single entry*): The term is used, e. g., by the MEKA program (Meacham 2007); MEKA itself is an abbreviation of “Multiple-Entry Key Algorithm”. This term *multiple entry* (and to a lesser extent *multiple-access*) may be misunderstood as referring to the starting point of an identification process, rather than to the choice of questions available at each *next* step. A key that truly only has multiple initial entry points would be equivalent to a branching key where the first couplet has many leads. A lesser problem is that a branching key may be entered at any couplet, if due to external knowledge the remaining taxa

could already be excluded. Logically and intuitively, the term starting point is usually understood to include only points from where all taxa keyed out may still be reached.

- **Multikey, polykey:** These terms may be potentially misunderstood whether they refer to the multiplicity of choice of characters or couplets or to multiple appearances of a key.
- **Polyclave:** According to Pankhurst (1991), the term “polyclave” was introduced by Duke (1969); it is the preferred term in Radford & al. (1974). Edwards & Morse (1995) state that it is the name for paper-based punched card systems. A major disadvantage of this term is that it is not a qualifier denoting the kind of key, but includes the term *key* itself. It is deceptive to talk of a *polyclave key* (i. e., “*multikey key*”). On 2007-04-15, Google found 84 results for this.
- **Dynamic polyclave:** Used by Radford & al. (1974) combining interactivity of a computer program with the principle of a multi-access key.
- **Nonsequential or non-branching key:** These would be the logical antonyms to sequential and branching keys; the terms are considered clumsy by native speakers of English.
- **Random access key:** The emphasis of “randomness” is misleading; no meaningful use of such a key starts with random access points. The word *random* is probably used in analogy to the term “Random Access Memory” (RAM) in computers. RAM itself is a misnomer. It is not accessed randomly, but is a volatile storage location that can be accessed directly in a non-sequential way. Indeed, the synonymous term “direct access memory” is occasionally preferred for RAM. As a consequence of the connotations of RAM, the term Random access key is intuitive to technical users, but highly confusing to most biologists.
- **Direct access key:** In analogy to renaming RAM to “direct access memory”, one might want to consider this term for the key analogy as well. However, the sense of *direct* in the context of a key seems to be rather ambiguous and perhaps intuitive to computer experts.
- **Dynamic path key:** This is a possible antonym to *fixed path key*; compare the discussion above.
- **Matrix key:** The data structure behind multi-access keys is a relatively completely (although not necessarily fully completely) filled matrix of characters or character states \times items (see Fig. 149 ff, p. 260 ff). One problematic aspect of using the term “matrix key” is that it can be misunderstood for what is here called a “tabular key” (Fig. 146, p. 257).

Terms that should be avoided are:

- *Interactive key:* The term is useful but does not correctly capture the difference between a sequential and multi-access key; see the previous section for further discussion.
- *Synoptic key:* The term has multiple contradictory definitions, see the following section for further discussion.

Most multi-access keys allow the user to select among all available characters or key questions. It may thus be surprising to see the prefixes *multi-* (Latin) or the equivalent *poly-* (Greek), both indicating “many” instead of, e. g., *omni-* or *toti-* (both based on Latin for “all” and “entire”, respectively). However, in certain styles it may indeed be beneficial to not display all characters, especially not those for which only few of the remaining taxa are coded (compare “Progressive revelation”, p. 270). The “many” is therefore appropriate. It will usually be understood to signify a substantial choice of entry points.

“Synoptic key”, a confused term

The term **synoptic key** (or “*synoptical key*”; occurring in Google, 2005-05-03, at a frequency of 1.2% relative to synoptic) seems to be one of the more frequently used terms for identification keys. Several text books (the oldest studied being Lawrence 1951) define the term as having the structure of a branching key, but intended for knowledge representation rather than diagnosis. Pankhurst (1991) defines synoptic keys as “*like conventional keys, but [...] intended to present a classification, rather than to identify actual taxa. The classification may well have been simplified or idealised, so such a key should not be used to make identifications.*” (p. 96).

The difference between a diagnostic and a synoptic key (or simple “synopsis”, e. g., Metcalf 1954) in this sense may be explained in an example. If a family contains small herbs, woody climbers, and shrubs, a “diagnostic key” would perhaps use these convenient characters at the expense of keying out the family in three different places. In contrast, a “synoptic key” would keep the family together at the expense of using inconvenient anatomical, embryological, ultra-structural, etc. characters. Furthermore, even if these characters can be studied, a synoptic key may not be able to key out all members of a taxonomic group, especially if some members are extremely deviant or reduced.

The history of this sense of “synoptic key” could not be traced precisely. Certainly influential is Gray’s “Synoptical flora of North America” (Gray 1878), which omits all artificial dichotomous keys and arranges the descriptions from genus upwards in a concise manner that is structurally similar to a nested key. It could be verified that a long tradition of using synoptic key in this sense exists in zoology and botany. Lawrence 1951 from botany was already mentioned, further examples are Lyon (1936, nested key of mammals), Smith (1943, linked key of ants), and Schuster (1958, nested key to Hepaticae). These citations are certainly not the oldest: all imply that the term “synoptic key” is commonly understood.

In contrast to these definitions, “synoptic key” in current usage frequently seems to be used as a synonym for a multi-access key. The use of “synoptic key” in this sense for multi-access keys was popularized and perhaps introduced by Leenhouts (1966). The claim of current usage may be supported by an informal case study analyzing the first 100 Google results for “synoptic key” (on 2005-03-05). Of these, 39 results referred to a so-called alphabetically indexed glossary and fact-sheet like structure in a publication by C. M. Boger on alternative medicine (“homeopathy”). Several other items could not be assessed (broken or pay-for-view links). Of the 23 results that could be analyzed, 18 referred to multi-access keys (78%), 2 to tabular branching keys (9%), and 3 to branching (dicho-/polytomous) keys (13%).

A somewhat surprising aspect is that 12 of the 18 multi-access keys that are called “synoptic” had mycological content (compare, e. g., Fig. 137, p. 254). This is out of proportion with the general use of multi-access keys. The use of “synoptic key” for multi-access keys was probably popularized in mycology by Korf (1972). Several lines of tradition seem to originate from there. Korf and PezWeb (<http://mgd.nacse.org/hyperSQL/pezweb/mainx.html>) have the common topic of Pezizales and in turn PezWeb uses tools (HyperSQL) that it shares with a “Lichenland” and a “plant nematode key” called both “synoptic”. Trappe is collaborator in PezWeb and author of Trappe (1982), which, being about mycorrhizal fungi, may lead to the mycorrhizal key “DDE” (http://dde.forrex.org/biodiversity/ecto/index_e.html), another mycorrhizal project. Another widely used mycological publication is Sutton (1980; the printed multi-access key example in Fig. 129, p. 250, is originally called “synoptic key” as well). Similarly to the Google search, a bibliographic query for “synoptic key” returned several uses of “synoptic key” in the sense of multi-access key; all of these were from mycology and after 1972.

Clearly, conflicting usage patterns for the term “synoptic key” exists. Furthermore, “synoptic key” also fits well for tabular keys; on p. 256 the “synoptic qualities” of these keys were stressed. For which sense should the term be reserved? Literally, the term “synopsis” means “viewing together”. Examples of dictionary definitions are “a condensation or brief review of a subject; summary” (CED 1992) and “a brief summary or general survey of something” (EB 2001). The problem with the reception of *synoptic key* seems to be that this may be interpreted as referring to:

- a condensed organization of information that is meant to give an overview of *information content*, especially features correlating with a taxonomic classification;
- a structural representation, where the *results* of identification steps can be easily overlooked, as in branching keys of the nested or tabular style;
- a printed multi-access key, where both the character list and the resulting taxa may be viewed together;

- a computer-aided multi-access key, where neither the content nor the results are synoptic, but the list of characters from which the user must choose can be seen together.

The rationale for calling a printed or a computer-aided multi-access key “synoptic” is quite different, but as the examples show, current usage in Mycology uses the term “synoptic key” for both forms of multi-access keys. In the light that two contradictory definitions of “synoptic key” exist in biology, it is recommended to avoid the use of “synoptic key” for multi-access keys.

In view of the general terminological confusion, and given the different ways in which “synopsis” may be interpreted, it is further recommended to avoid “synoptic key” altogether. Leenhouts (1966) and Korf (1972) use the term “natural key” for the older sense of “synoptic key” as a branching keys intended for knowledge representation rather than diagnosis. Other candidates for a new term for this sense are “phylogenetic key”, “systematic key”, or “taxonomic key” (all proposed here).

“Synoptic” may still be used to express that a given key is intended for a synopsis of a specific aspect. It should, however, not be used in a sense of conveying information on a specific key structure or on what kind of synopsis is intended.

10.4. SQL code for DiversityDescriptions 1.9

SQL Data Definition Language code to create a DiversityDescriptions 1.9 implementation. Note that only Not Null and Default constraints are present in this code, further constraints may be found in the “Data dictionary” (p. 339). SQL92 is used wherever possible. “National Character” is the somewhat counterintuitive way of expressing “international character set” (i. e. Unicode) in SQL92.

```
CREATE TABLE DD_CHAR (
  CID SMALLINT NOT NULL PRIMARY KEY,
  CharName NATIONAL CHARACTER VARYING(255) NOT NULL UNIQUE,
  Unit NATIONAL CHARACTER VARYING(255) NULL,
  Notes NATIONAL TEXT NULL,
  Type NATIONAL CHARACTER VARYING(2) NOT NULL DEFAULT 'UM',
  Mandatory BOOLEAN NOT NULL DEFAULT 0,
  MultiStateType TINYINT NOT NULL DEFAULT 1,
  Reliability FLOAT(7) NOT NULL DEFAULT 5,
  Availability FLOAT(7) NOT NULL DEFAULT 5,
  Fuzziness FLOAT(7) NOT NULL DEFAULT 0,
  FuzzinessIsPercent BOOLEAN NOT NULL DEFAULT 0,
  KeyStates NATIONAL CHARACTER VARYING(255) NULL,
  CharHeading SMALLINT NULL,
  HeadingLink SMALLINT NULL,
  CharWording NATIONAL CHARACTER VARYING(255) NULL,
  CharWording2 NATIONAL CHARACTER VARYING(255) NULL,
  UnitIsPrefix BOOLEAN NULL,
  FormatString NATIONAL CHARACTER VARYING(255) NULL,
  ParagraphLink INTEGER NULL DEFAULT 1,
  SentenceLink INTEGER NULL,
  CommaLink INTEGER NULL,
  UseComma2 BOOLEAN NULL,
  OmitFinalComma BOOLEAN NOT NULL DEFAULT 0,
  OmitValues NATIONAL CHARACTER VARYING(1) NULL,
  Emphasize BOOLEAN NULL DEFAULT 0,
  OmitPeriod BOOLEAN NULL DEFAULT 0,
  NumStates SMALLINT NOT NULL DEFAULT 2,
  CharID INTEGER IDENTITY NOT NULL UNIQUE
);

CREATE TABLE DD_CHAR_Translation (
  CharID INTEGER NOT NULL,
  Language NATIONAL CHARACTER VARYING(2) NOT NULL,
```

```
CharName NATIONAL CHARACTER VARYING(255) NOT NULL,
CharWording NATIONAL CHARACTER VARYING(255) NULL,
CharWording2 NATIONAL CHARACTER VARYING(255) NULL,
Unit NATIONAL CHARACTER VARYING(50) NULL,
UnitIsPrefix BOOLEAN NULL,
Notes NATIONAL TEXT NULL,
FormatString NATIONAL CHARACTER VARYING(255) NULL,
PRIMARY KEY (CharID,Language)
);

CREATE TABLE DD_CHAR_Heading (
  HID SMALLINT NOT NULL PRIMARY KEY,
  HeadingName NATIONAL CHARACTER VARYING(255) NOT NULL UNIQUE,
  HeadingWording NATIONAL CHARACTER VARYING(255) NULL,
  Notes NATIONAL TEXT NULL,
  AutoGroup NATIONAL CHARACTER VARYING(255) NULL,
  ParentHeadingID INTEGER NULL,
  HeadingID INTEGER IDENTITY NOT NULL UNIQUE
);

CREATE TABLE DD_CHAR_Heading_Translation (
  HeadingID INTEGER NOT NULL,
  Language NATIONAL CHARACTER VARYING(2) NOT NULL,
  HeadingName NATIONAL CHARACTER VARYING(255) NOT NULL UNIQUE,
  HeadingWording NATIONAL CHARACTER VARYING(255) NULL,
  Notes NATIONAL TEXT NULL,
  PRIMARY KEY (HeadingID,Language)
);

CREATE TABLE DD_CHAR_Heading_Link (
  HID SMALLINT NOT NULL,
  CID SMALLINT NOT NULL,
  PRIMARY KEY (HID,CID)
);

CREATE TABLE DD_CS (
  CID SMALLINT NOT NULL,
  CS NATIONAL CHARACTER VARYING(16) NOT NULL,
  CharStateName NATIONAL CHARACTER VARYING(255) NOT NULL UNIQUE,
  Notes NATIONAL TEXT NULL,
  StateWording NATIONAL CHARACTER VARYING(255) NULL,
  StateFormatString NATIONAL CHARACTER VARYING(255) NULL,
  Implicit BOOLEAN NOT NULL DEFAULT 0,
  UseEdit BOOLEAN NULL DEFAULT 1,
  UseIdentify BOOLEAN NULL DEFAULT 0,
  UseDescr BOOLEAN NULL DEFAULT 0,
  UsePhylo BOOLEAN NULL DEFAULT 0,
  UseOther BOOLEAN NULL DEFAULT 0,
  MinValue FLOAT NOT NULL DEFAULT -1E+308,
  MaxValue FLOAT NOT NULL DEFAULT 1E+308,
  StateID INTEGER IDENTITY NOT NULL UNIQUE,
  PRIMARY KEY (CID,CS)
);

CREATE TABLE DD_CS_Translation (
  StateID INTEGER NOT NULL DEFAULT 0,
  Language NATIONAL CHARACTER VARYING(2) NOT NULL,
  CharStateName NATIONAL CHARACTER VARYING(255) NOT NULL,
  Notes NATIONAL TEXT NULL,
  StateWording NATIONAL CHARACTER VARYING(255) NULL,
  StateFormatString NATIONAL CHARACTER VARYING(255) NULL,
  PRIMARY KEY (StateID,Language)
);
```

```
CREATE TABLE DD_MOD (
  Usage NATIONAL CHARACTER VARYING(255) NOT NULL,
  Modifier NATIONAL CHARACTER VARYING(255) NOT NULL PRIMARY KEY,
  Reliability TINYINT NOT NULL DEFAULT 5,
  MisinterpretationMarker BOOLEAN NOT NULL DEFAULT 0,
  Postfix BOOLEAN NOT NULL,
  UseBlank BOOLEAN NOT NULL DEFAULT 1,
  Operator TINYINT NOT NULL DEFAULT 0,
  Notes NATIONAL CHARACTER VARYING(255) NULL,
  LowerFreq FLOAT(7) NULL,
  UpperFreq FLOAT(7) NULL
);
```

```
CREATE TABLE DD_MOD_Translation (
  Modifier NATIONAL CHARACTER VARYING(255) NOT NULL,
  Language NATIONAL CHARACTER VARYING(2) NOT NULL,
  ModifierTranslation NATIONAL CHARACTER VARYING(255) NOT NULL,
  PRIMARY KEY (Modifier,Language)
);
```

```
CREATE TABLE DD_MOD_Link (
  CID SMALLINT NOT NULL,
  Modifier NATIONAL CHARACTER VARYING(255) NOT NULL,
  PRIMARY KEY (Modifier,CID)
);
```

```
CREATE TABLE DD_DEP (
  CID SMALLINT NOT NULL,
  CS NATIONAL CHARACTER VARYING(16) NOT NULL,
  InapplicableCID SMALLINT NOT NULL,
  PRIMARY KEY (CID,CS,InapplicableCID)
);
```

```
CREATE TABLE DD_ITEM (
  IID INTEGER NOT NULL PRIMARY KEY,
  ItemName NATIONAL CHARACTER VARYING(255) NOT NULL,
  ItemWording NATIONAL CHARACTER VARYING(255) NULL,
  Notes NATIONAL TEXT NULL,
  Abundance FLOAT(7) NOT NULL DEFAULT 5,
  CollUnit NATIONAL CHARACTER VARYING(255) NULL,
  LitRef NATIONAL CHARACTER VARYING(255) NULL,
  LitKey INTEGER NULL,
  LitRefDetail NATIONAL CHARACTER VARYING(255) NULL,
  ItemID INTEGER IDENTITY NOT NULL UNIQUE
);
```

```
CREATE TABLE DD_DESCR (
  IID INTEGER NOT NULL,
  CID SMALLINT NOT NULL,
  Modifier NATIONAL CHARACTER VARYING(255) NULL,
  CS NATIONAL CHARACTER VARYING(16) NOT NULL,
  X FLOAT NULL,
  TXT NATIONAL TEXT NULL,
  Notes NATIONAL TEXT NULL,
  SEQ INTEGER NULL,
  PRIMARY KEY (CID,CS,IID)
);
```

```
CREATE TABLE DD_RSC (
  ItemID INTEGER NULL,
  CharID INTEGER NULL,
  StateID INTEGER NULL,
  Resource NATIONAL CHARACTER VARYING(255) NOT NULL,
  Caption NATIONAL TEXT NULL,
  Language NATIONAL CHARACTER VARYING(2) NULL,
  Role NATIONAL CHARACTER VARYING(1) NOT NULL DEFAULT 'S',
```

```
ItemUsage NATIONAL CHARACTER VARYING(1) NULL,
CharUsage NATIONAL CHARACTER VARYING(1) NULL,
Notes NATIONAL TEXT NULL,
DisplayOrder INTEGER NOT NULL DEFAULT 0,
ResourceID INTEGER IDENTITY NOT NULL PRIMARY KEY
);

CREATE TABLE DD_PROPERTY (
PropertyName NATIONAL CHARACTER VARYING(255) NOT NULL PRIMARY KEY,
TextValue NATIONAL TEXT NULL,
DateTimeValue DATETIME NULL,
NumericValue FLOAT NULL,
Language NATIONAL CHARACTER VARYING(2) NOT NULL DEFAULT 'en'
);

CREATE TABLE DD_CurrentLanguage (
ID INTEGER NOT NULL PRIMARY KEY DEFAULT 1,
Language NATIONAL CHARACTER VARYING(2) NOT NULL
);

ALTER TABLE DD_CHAR ADD
FOREIGN KEY (CharHeading) REFERENCES DD_CHAR_Heading (HID)
ON UPDATE CASCADE,
FOREIGN KEY (HeadingLink) REFERENCES DD_CHAR_Heading (HID)
ON UPDATE CASCADE
ALTER TABLE DD_CHAR_Translation ADD
FOREIGN KEY (CharID) REFERENCES DD_CHAR (CharID)
ON UPDATE CASCADE ON DELETE CASCADE
ALTER TABLE DD_CHAR_Heading ADD
FOREIGN KEY (ParentHeadingID) REFERENCES DD_CHAR_Heading (HeadingID)
ON UPDATE CASCADE
ALTER TABLE DD_CHAR_Heading_Translation ADD
FOREIGN KEY (HeadingID) REFERENCES DD_CHAR_Heading (HeadingID)
ON UPDATE CASCADE ON DELETE CASCADE
ALTER TABLE DD_CHAR_Heading_Link ADD
FOREIGN KEY (HID) REFERENCES DD_CHAR_Heading (HID)
ON UPDATE CASCADE ON DELETE CASCADE,
FOREIGN KEY (CID) REFERENCES DD_CHAR (CID)
ON UPDATE CASCADE ON DELETE CASCADE
ALTER TABLE DD_CS ADD
FOREIGN KEY (CID) REFERENCES DD_CHAR (CID)
ON UPDATE CASCADE ON DELETE CASCADE
ALTER TABLE DD_CS_Translation ADD
FOREIGN KEY (StateID) REFERENCES DD_CS (StateID)
ON UPDATE CASCADE ON DELETE CASCADE
ALTER TABLE DD_MOD_Translation ADD
FOREIGN KEY (Modifier) REFERENCES DD_MOD (Modifier)
ON UPDATE CASCADE ON DELETE CASCADE
ALTER TABLE DD_MOD_Link ADD
FOREIGN KEY (CID) REFERENCES DD_CHAR (CID)
ON UPDATE CASCADE ON DELETE CASCADE,
FOREIGN KEY (Modifier) REFERENCES DD_MOD (Modifier)
ON UPDATE CASCADE ON DELETE CASCADE
ALTER TABLE DD_DEP ADD
FOREIGN KEY (InapplicableCID) REFERENCES DD_CHAR (CID)
ON UPDATE CASCADE ON DELETE CASCADE,
FOREIGN KEY (CID,CS) REFERENCES DD_CS (CID,CS)
ON UPDATE CASCADE ON DELETE CASCADE
ALTER TABLE DD_DESCR ADD
FOREIGN KEY (CID,CS) REFERENCES DD_CS (CID,CS)
ON UPDATE CASCADE ON DELETE CASCADE,
FOREIGN KEY (IID) REFERENCES DD_ITEM (IID)
ON UPDATE CASCADE ON DELETE CASCADE,
FOREIGN KEY (Modifier) REFERENCES DD_MOD (Modifier)
ON UPDATE CASCADE ON DELETE CASCADE
ALTER TABLE DD_RSC ADD
```

```

FOREIGN KEY (CharID) REFERENCES DD_CHAR (CharID)
ON UPDATE CASCADE ON DELETE CASCADE,
FOREIGN KEY (StateID) REFERENCES DD_CS (StateID)
ON UPDATE CASCADE ON DELETE CASCADE,
FOREIGN KEY (ItemID) REFERENCES DD_ITEM (ItemID)
ON UPDATE CASCADE ON DELETE CASCADE

```

Note: additional indexes, table and some field constraints are not shown here, compare section “Data dictionary” (p. 339) for additional information on tables and fields.

10.5. Index of figures

Fig. Abbreviated caption	Page
1 Estimated species diversity of major groups and proportion of known taxa	14
2 “... mycologists inadvertently redescribe already known species”	14
3 Elements of UML use case diagrams	24
4 UML use case dialects may use different symbols for system and business actors	24
5 Relationships in UML static class diagrams	25
6 Potential DiversityWorkbench components grouped by similarity	29
7 Relation of DiversityWorkbench components with DiversityDescriptions	29
8 Recognition of object parts is based on recognition of properties, life-cycle stage, and composition	37
9 Interpretation of the recognition of object parts as a heuristic cycle	37
10 Additional dependencies on observation methodology and taxonomic diversity	38
11 Natural language descriptions are useful when comparing an existing object with a description, but often a poor method to visualize objects or compare descriptions with each other	40
12 Example of an editing form and the corresponding natural language description	41
13 Comparison of informal presentation, UML object diagram and a corresponding class diagram	42
14 Abstract descriptive data information corresponding to the specific model shown in Fig. 13	43
15 A collection of objects that would require a refinement of the properties used in Fig. 13	45
16 Illustration of categorical measurement scales (unordered, ordered linear, tree, and cyclic)	50
17 Two-dimensional visualization of discrete or continuous reality and categorizations applied to it	54
18 Squares with rounded corners form a continuum of intermediate shapes between square and circle	65
19 Mapping of quantitative length measurements to fine and coarse categorizations	67
20 Mappings from measurement to categorical data may be based on confidence intervals	67
21 Symbolic presentation of a “value space” that is partitioned into fine-grained categories and for which a generalization mapping to coarse-grained categories is provided	68
22 Mappings may provide for ambiguity or error tolerance	68
23 Illustration of a potential tree-view user interface displaying ambiguously mapped categorical values	69
24 Multiple measurements may have fewer degrees of freedom than variables	72
25 Depending on measuring method, length measures may not be additive	72
26 Generalization hierarchy of character data and coding status values	75
27 Character applicability rule and complementary rule applied to the complementary set of states	79
28 Original applicable-if-semantics are difficult to preserve when new states are added to a character after the original rule has been converted to the complementary inapplicable-if rule	79
29 The results of an inapplicable-if rule, and the results of the complementary applicable-if rule applied to complementary states are identical	80
30 Character applicability rules for the same combination of controlling and controlled character may be explicitly or implicitly contradictory	81
31 Character applicability rules may cascade several levels deep	82
32 Selection from a generalization hierarchy of statistical measures	86
33 Examples of multistate (= polymorphic) situations in a single individual at a single point in time	94
34 Compatibility of item descriptions	103
35 Illustration of the CSIRO DELTA Windows editor in grid view	111
36 Illustration of the DiversityDescriptions database editor	111
37 A flat character list may be organized under different aspects through multiple concept hierarchies	126
38 Multiple concept hierarchies (trees) may be defined for a character list	126
39 A sparsely filled matrix with 200 out of the 1000 potential combinations selected for data entry	127
40 In the SDD model, views (for data entry, report generation, sorting, etc.) may be added as another concept tree; in the Prometheus model, a separate selection mechanism (called “pro-forma”) is used	127

Fig. Abbreviated caption	Page
41 Compositional and a methodological concept hierarchy associated with a flat character list	130
42 Multiple concept hierarchies may be combined, providing additional information wherever the primary hierarchy contains multiple characters at a concept	130
43 Abstract example of an object composed of other objects (components)	132
44 An excerpt from a morphological composition hierarchy of human body parts	132
45 An inner, anatomical composition hierarchy aligned with the outer composition hierarchy	133
46 An excerpt from an anatomical composition hierarchy of humans showing a detailed hierarchy for lungs	133
47 Interference between morphology and anatomy at the example of the human respiratory system	134
48 Teeth of a dentate margin are usually not considered components, whereas hairs at the margin or on any surface are often considered component objects	134
49 Object color may be due to surface structures, pigments, or soluble molecules	134
50 Ambiguous object decomposition	135
51 Recognition of “head” and “body” is based on other features if head and neck are becoming reduced	135
52 Examples of species with gradually changing leaf shapes, defying a strict classification	137
53 An abstract, generalized list of plant parts is filtered through a list of present/absent characters	138
54 In description models providing an explicit compositional hierarchy, hierarchy information may be inherited by the actual composition for a specific taxon	138
55 Hierarchy attempting to use only generalized terms	140
56 Six instances of the leg class associated with a thorax instance of a particular insect	143
57 Static class diagram with an attempt to model an insect as a simple composition	144
58 Static class diagram extending Fig. 57 to include sample objects	144
59 Multiplicity may be expressed through quantitative properties and could replace the presence/absence filter	146
60 Static class diagram similar to Fig. 57, but expressing multiplicity through a “Quantity” attribute	146
61 Objects obtain an absolute orientation (e. g., top/bottom) through convention	147
62 Parts of composite objects may have relative orientation	148
63 Orientation of shapes may result in different shape categories (e. g., ovate / obovate)	148
64 Geometric composite object with bilateral symmetry	149
65 Minor variations may break strict symmetry, but an abstract concept of “approximate symmetry” remains	149
66 Density gradient	150
67 Rectangular parent object with multiple attached child objects of variable size	150
68 Three class diagrams attempting to model object adjacency	151
69 Class diagram of the insect example introducing a tentative association of “adjacent” stereotype	152
70 Class diagram showing an object generalization and composition model for geometric object	153
71 Class diagrams showing the nine major organ systems of the human body	154
72 Class diagram modeling the surface (e. g., of a plant leaf)	154
73 Class diagram adding a leaf context to the surface abstraction shown in Fig. 72	155
74 Further detail added to the ontology already shown in Fig. 73	161
75 Five different leaf shapes, all of which might be described as the abstract shape “elliptical”	165
76 Patterns that can either be described as object compositions or as named pattern categories	166
77 The concept of a pattern abstracts from concrete color to the ability to distinguish parts	166
78 Two populations (clouds) with four patterned objects each	167
79 Example for a dependency of shape and size properties	168
80 Orientation (upper/lower side) may be significant or insignificant when recording object properties	168
81 Measurements may result in different results depending on measurement procedures	169
82 Class diagram showing methods as a generalization of experiment and observation	171
83 Class diagram showing a selection from a generalization hierarchy of observation methods	172
84 Class diagram showing that properties can be observed only by specific methods	173
85 Class diagram showing that some methods may be combined to observe a property	174
86 Dependencies of object properties on “states” of the observation situation are structurally very similar to dependencies among different properties	176
87 Class diagram attempting to outline a property/method/object-part/character model for the color example	177
88 Modified version of Fig. 87, adding an interpretation of the “basic property” concept	178
89 Simplified comparison of a character plus concept hierarchy model (DELTA, SDD), a character decomposition model (Nemisys/Genisys), and a potential part-property-method decomposition model	179
90 Options for federating, modularizing, and extending descriptive terminologies	181
91 Use case diagram for some use cases involving external (federated) terminologies	182
92 Example for a hierarchy of terminology modules that follows a taxonomic hierarchy	184
93 Combining multiple terminologies (“character definitions”) can also be useful to combine characters defined for different methods and add them to the current project as needed	184
94 External terminology may be copied or linked, the latter optionally with a local cache	185

Fig. Abbreviated caption	Page
95 Network namespace model for federated terminologies	186
96 Template model for federated terminologies	187
97 In the declarative model each term of the local terminology contains, among other data elements, an optional reference to a standard terminology	188
98 Consensus terminology created by a join of multiple terminologies from multiple sites on the internet	189
99 Simplified comparison of DELTA- and SDD-like models in regard to modifiers and free-form text annotations ...	190
100 Excerpt from a scoring scheme for fungal spores	191
101 Simplified comparison of Lucid3 and Nemisys/Genisys in regard to modifiers and free-form text annotations	197
102 Three dimensions of spatial and temporal change resulting in $3 \times 5 \times 3 = 45$ different specialized classes	201
103 Objects that belong to the same class may have variability	206
104 Treating sex similar to infraspecific taxon ranks	223
105 Visualization of the nested nature of the taxonomic hierarchy	223
106 Developmental life cycle stages run across the taxonomic hierarchy	223
107 Sex and stage arbitrarily nested inside the taxonomic hierarchy	224
108 The conceptual dimensions of sex and life cycle stages may be dependent and nested	224
109 Character \times description matrix where development stages are expressed through separate sets of characters ..	225
110 Character \times description matrix where spore stages of rust fungi are expressed through separate sets of characters	225
111 Character \times description matrix where development stages are designated using a secondary classifier mechanism	226
112 Visualization of objects with taxon references that require a secondary classifier mechanism	227
113 The class references from class hierarchy (in biology = taxonomic hierarchy) to class definitions, and from synonyms to class names do not require a secondary classifier mechanism	227
114 An example for a printable, hyperlinked dichotomous key	234
115 Principle of a branching key	235
116 A variant class of branching keys includes redirections (or "reticulations")	235
117 Visualization of possible user interaction steps in a branching key	236
118 A polytomous key split into one question per page, image captions providing mnemonics (Electr. Field Guide) .	239
119 Example of analytical illustrations, detailing diagnostically relevant parts of the organism	241
120 Examples of the linked and nested styles of branching keys in lead style	244
121 Variant styles of branching keys (adding a/b and backtracking; displaying start of the lead text in bold)	244
122 Variants of branching keys; indentation present in linked and omitted in "indented" nested style	245
123 Branching keys in nested style, adapted for use with polytomous couplets	245
124 An example of a branching key in graphical style	246
125 An example for a printable, hyperlinked dichotomous key	247
126 An abbreviated example of an online dichotomous key, split into one question per page	247
127 A picture-based branching key with one question per page, tracking the history of previous statements	248
128 An example of CBIT Lucid Phoenix	249
129 An example of a printed multi-access key in "character-list style"	250
130 An example of a printed multi-access key in "character formula style"	250
131 An example of an Identify key running in DiversityDescriptions	252
132 An example of a CSIRO Intkey key running under Windows	252
133 An example of a Lucid 3 key running under Windows	252
134 CBIT Lucid 3 may also be used over the internet if Java is permitted	253
135 Navikey 4.09 Java applet running in an internet browser	253
136 Navikey 4.09 using a quantitative character	253
137 An example of a multi-access key that is limited to a few pre-selected characters	254
138 An example of a multi-access key using DeltaAccess-Perl (DAP)	254
139 An example of an X:ID key running in the web browser	254
140 An example of a multi-access key using ActKey	254
141 An example of a 3I web key	255
142 3i is multilingual, here displayed in Russian	255
143 An example of a SAIKS key in identification mode	255
144 An example of a SAIKS key in "show-character" mode	256
145 An example of a SAIKS key showing a subkey with back-reference to previous key	256
146 An example of tabular key combined with short diagnostic descriptions	257
147 An example of tabular key including illustrations	257
148 Multiple subkeys may follow a single lead ("result-and-continue pattern" or list of available keys)	259
149 Hierarchically linked keys	260
150 Scores recorded during identification may be transferred from one key to the other	260

Fig. Abbreviated caption	Page
151 Multi-access key with sub matrices designated for coding and "Not to be coded" coding status value	261
152 Multi-access key with areas relevant to subgroups described as item and character subsets	261
153 A branching key connected with a multi-access key	263
154 Illustration of equality criteria	265
155 Available options in Identify for specifying quantitative measurement obtained from the object to be identified ..	266
156 Effect of unevenness on H values	271
157 Area diagram illustrating three states with 0 to 100% overlap	273
158 Effect of overlapping p-values on H for 2 to 5 states with equal frequencies	273
159 Linear correction for missing data	273
160 Entropy function H_c , correcting overlap by dividing through sum-of-frequencies	273
161 Potential UML use case actors and their hierarchy	279
162 Use case diagram for project management	281
163 Use case diagram showing division into ontological and operational terminology	282
164 Use case diagram for the definition of ontological terminology	284
165 Use case diagram for the definition of operational terminology	285
166 Use case diagram showing inclusion and extension use cases for defining "label, abbreviations, wordings", "elaborate definitions", and "associate terminology with media resources"	286
167 Use case diagram showing a modification of Fig. 166; association of media resources is now an extension	286
168 Inclusion of label and glossary use cases in operational terminology	286
169 Use case diagram showing the specific use cases of abstract "Associate with media resource"	287
170 Use case diagram showing options for defining coding status values and statistical measures	287
171 Use cases derived from "Record descriptive data" for individual objects and class descriptions	289
172 Use case diagram extending Fig. 171 by interactions derived from "Record published descriptive data"	290
173 Use case diagram showing various modes of recording character data	291
174 Example for a multi-description editor (DiversityDescriptions)	293
175 Use case showing markup of legacy descriptions and keys	294
176 Use case diagram showing an extension to the cases described in Fig. 175	294
177 Use case diagram showing the recording of descriptive data in the context of identifications	296
178 Use case diagram showing recording of repeated original observation data	297
179 Use case diagram showing a sample of use cases that include language and audience selection	298
180 Use cases specializing the selection of a branching key	299
181 Use case diagram showing identification as a specialized use case of queries to find descriptions	300
182 All analysis use cases include the selecting of descriptive data in a query	301
183 Diagram showing a common cyclic sequence of analysis and editing use cases	301
184 Use case diagram for data quality analysis	302
185 Visualizations of character relationships may also be used for quality control	302
186 Example of a possible quality control report for quantitative characters	303
187 Diagram of potentially sequential process steps involved in character correlations analysis	304
188 Use case diagram showing the analysis of character correlations	305
189 Use case diagram showing the specific use cases of "Generalize or aggregate descriptions"	305
190 A possible cascade of use cases invoking other use cases to update information before aggregating it	305
191 Use case diagram showing the categorization of objects through data analysis	306
192 Use case diagram showing the review of character distribution superimposed on a phylogenetic hierarchy	307
193 Use case diagram showing the analysis of character distribution by tree correlation	307
194 Use case diagram showing creation of diagnostic subsets	307
195 Use case diagram showing the identification of an object through branching or multi-access keys	308
196 Extension of identification with lookup and report generation of terminological definitions	309
197 Use case diagram showing switching between different identification methods	309
198 Use case diagram showing the confirmation phase after an initially successful identification	310
199 Use case diagram showing potential methods to broaden the result set of an identification	310
200 Use case diagram showing the verification step after identification	311
201 Identification of potential taxon concepts viewed as a specialization of general name-based identification	311
202 Branching keys may be created automatically, assisted, fully manually, or through markup of legacy data	312
203 Creation of interactive or branching keys as a special case of object categorization	313
204 Phylogenetic categorization may be relevant in the creation of branching keys	314
205 Use case diagram showing that report generation includes finding/selecting descriptions to be reported	315
206 Use case diagram showing report generation of descriptive data and associated terminology	315
207 Use case diagram showing the case of diagnostic descriptions that are abbreviated to contain only characters differentiating between members of a limited item/taxon group	316

Fig. Abbreviated caption	Page
208 Use case diagram showing the rendering of printable multi-access keys	317
209 Use case diagram showing the rendering of printable branching keys	318
210 Report generation for terminology and descriptions may involve the creation of indices or tables of content	318
211 Relation between descriptive data and other biodiversity data areas shown as a package diagram	320
212 Taxon pages are created combining descriptive information with other information sources	320
213 Use case diagram showing export of descriptive data	321
214 Partial or complete import of descriptive data, associated terminology, and project metadata	321
215 UML subsystem and package diagram for the descriptions model in DiversityDescriptions 1.9	324
216 Class diagram for major classes of the Terminology package	325
217 Conceptual class diagram showing subclasses for different kind of character data	326
218 Class diagram showing details of three different heading variants and attributes for the generation of natural language descriptions	327
219 Diagram illustrating value-based link group attributes on characters in a natural language description	328
220 Diagram illustrating the hierarchical nature of link groups	328
221 Diagram illustrating how link groups may be translated into "report elements" defining delimiters	329
222 Class diagram illustrating character dependency	329
223 Class diagram showing the Descriptions package (left) and its relations to terminology	330
224 Class diagram showing the conceptual superclass/subclass design of the DESCR (Descriptions package) and CS (Terminology package)	330
225 Class diagram illustrating the Resource entity and relations to entities from other packages	332
226 Project subsets; in DiversityDescriptions these are based on views, allowing both analysis and editing	334
227 Reduced entity relationship diagram of DiversityDescriptions 1.9 containing only the principal tables	335
228 The different character-grouping mechanisms in DELTA	336
229 Complete entity relationship diagram of DiversityDescriptions 1.9 (physical model)	337
230 Simplified Resource table and its relations with other packages	338
231 DiversityNavigator grid view editor for descriptive data	360
232 CharApplicability_T1ControlledControllingWithAllStates (step 1) in MS Access qbe-view	391
233 CharApplicability_T2ControlledControllingAllStateStatus (step 2) in MS Access qbe-view	391
234 CharApplicability_D1JoinDescription (step 1) in MS Access qbe-view	393
235 CharApplicability_D2HavingStatesMakingApplicable (step 2) in MS Access qbe-view	393
236 CharApplicability_D3InapplicableChar (step 3) in MS Access qbe-view	394
237 An example of an interactive dichotomous key, split into one question per page	395

10.6. Index of tables

Tab. Abbreviated caption	Page
1 Common multiplicity indicators (i. e., ranges of allowable cardinalities) used in UML class associations	25
2 Generic data types used in ER-/UML-class diagrams	26
3 Overview of usage of terms in descriptive information models and computer science	34
4 Consequences of changing the terminology in an extended character/state model	45
5 Different implementation concepts for software based on an abstract model plus domain-specific terminology	49
6 Interaction between continuous/discrete variation and measurement scales	51
7 Examples of different classes of categorical measurements	55
8 DELTA character types	61
9 "Basic morpho-anatomical properties" proposed in Diederich & al. (1997, 1998)	63
10 A modified basic-property-concept extended with a generalization hierarchy	64
11 Example for mixed (quantitative and categorical) data recording	67
12 Different scenarios depending on the data recording format used during identification and the format available in the knowledge base used for comparison	70
13 Examples of potentially useful mappings of measurement characters to "calculated" characters	73
14 Example showing potential combinations of available measurements for leaf length measurements	74
15 Coding status values (= "pseudo-values") in DELTA	74
16 Coding status values defined in SDD	75
17 Examples of evaluating applicable/inapplicable-if rules involving multiple states	78
18 Classifications of univariate statistical methods used in SDD	87
19 Aggregating statistical measures	89
20 Repeated quantitative and categorical measurements for independently measured data for a single object	91
21 Example data to test 'and' and 'or' statements on different aggregation levels	95
22 Values of the StateCollectionModelEnum in SDD	97

Tab. Abbreviated caption	Page
23 Enumerated values in the "DataOriginEnum" in SDD, documenting the origin of a descriptive data value	102
24 Character × entity matrix	104
25 Character state × entity matrix	105
26 Mixed model, using the character × entity matrix for "character 1" with exclusive states	106
27 Presentation of the matrix models as a list	106
28 Asymmetry of positive and negative state scores during aggregation (induction of knowledge "up the tree")	109
29 Asymmetry in the intuitiveness of positive and negative scores inherited from higher to lower taxa	109
30 Examples of statistical measures in the list model	110
31 Comparison of statistical measures supported in descriptive software applications and exchange formats	112
32 Fundamental part/property model	118
33 An example based on the detailed Nemisys/Genisys model (including "name extensions" and "qualifiers")	118
34 Examples of conventional characters that are difficult to decompose into property and object part	121
35 Cases that may be termed a "relational character"	122
36 Character decomposition based on the steps required to observe and record a value for a character	124
37 Examples of related characters that are distinguished by three parameters of a single method used	125
38 Comparison of the character decomposition and concept hierarchy models	128
39 Examples of ambiguous or competing classifications of plant parts	136
40 Examples of related object parts distinguished by placement or otherwise	140
41 Examples of location statements taken from biology	152
42 Examples from biology for generalizations of object parts under three different generalization concepts	155
43 Examples of competing classifications of spores and sporogenous cells in ascomycetes	159
44 Examples of dictionary and UML definitions for attribute, feature, and property	164
45 Examples of dictionary and UML definitions for "modifier" and "qualifier"	192
46 Excerpt from template modifier definitions in DeltaAccess/DiversityDescriptions 1.0	194
47 Examples of use of name-extensions from Diederich & al. (1997)	195
48 Interpretation of the structure of relative modifiers, based on information in Pullan & al. (2005)	197
49 Enumerated modifier classes in SDD	198
50 Examples illustrating the diversity of spatio-temporal leaf spot locations	201
51 Three dimensions with $3 \times 3 \times 5 = 45$ resulting characters or character/modifier combinations	202
52 Alternative classification systems and sources of intra-class variation in biology and musical instruments	216
53 Examples of different presentations of sex and life cycle stage classifiers	217
54 Examples of dictionary definitions for "identification" and "identify"	229
55 Character equality parameters ("Set Match") in CSIRO Intkey	266
56 Information content H for binary/multistate characters, without/with overlap among state frequencies	272
57 Example calculation for overlap between result sets involving missing data	272
58 Possible combinations between object ID, naming/ identification, and publication status	290
59 Example of a quality control report for linearly ordered categorical characters	303
60 Statistical measures recognized by algorithms in DiversityDescriptions	356
61 Example data to illustrate SQL code for character applicability rules	390

10.7. Overview of collected requirements

- | | |
|--|--|
| 1. The provision of free-form text natural language descriptions (full or diagnostic) is desirable, for example, during the identification process.....41 | 5. A descriptive terminology is required as mediator between the abstract information model and the concrete properties and observation methods in a taxonomic group..... 45 |
| 2. These may either be authored (including digitized legacy descriptions) or automatically generated from more structured descriptive data forms. A number or data items is required for natural language generation – these are not discussed further in this thesis, but are present in the models presented.42 | 6. A dynamically evolving terminology that may be changed while recording descriptive data is possible and desirable..... 47 |
| 3. For many purposes, natural language descriptions are inadequate. Supplying more structured forms of descriptions is highly desirable.....42 | 7. Standardizing terminology is desirable and the information model should support this. 48 |
| 4. Although simple, taxon group and observation methodology-specific information models for structured descriptive information do have a | 8. A stable and global identifier method for objects of the descriptive terminology is required..... 48 |
| place, a more abstract and generalized information model is desirable..... 44 | 9. Rigid standardization is no alternative to providing support for a freely evolving terminological schema evolution. 48 |

10. Different concepts exist to implement software based on a combination of a generalized, abstract base model plus domain-specific terminology. The choice depends on available development tools and is *not* part of the model requirements.49
11. The concept of measurement scales is an important concept in statistical and phylogenetic data analysis and should be reflected in the descriptive data type system.50
12. The distinction between continuous and discrete is relevant, but different perspectives exist when using it for descriptive data. Depending on the perspective, the distinction does or does not interact with the concept of measurement scales. It is, however, never nested within measurement scales.52
13. The distinction between categorical and quantitative data is a fundamental that should be reflected in the descriptive data type system.53
14. In many object descriptions information about variation underlying the categorization is implied. A better agreed terminology seems to be desirable to accurately express knowledge about ordering, ranges, inclusiveness of border values, and presence of intermediate values in an unambiguous way. This topic requires further study; the terminology presented in Fig. 17, in Table 7 (both above) and 22 (p. 97) is intended only as a first attempt.55
15. The matching of descriptive data types with those implemented in specific processors, programming languages or DBMS is not a requirement.56
16. However, commonly used data types (current practice for other disciplines) may help to guide priorities and are relevant when estimating the cost of implementing a concept.56
17. Sequences or arrays of data types may be desirable. Based on current practices (no existing software implements sequences other than free-form text), the requirement seems to be weak.56
18. A data type for unconstrained, free-form text is desirable. The text may either be an extension of more structured information (such as a character state or quantitative value) or it may replace more structured information.57
19. A free-form text extension mechanism (e. g., "comments", "notes") may be part of the fundamental information model and always available.57
20. Whether free-form text may replace more structured information or not should be controllable by the designers of the terminology.57
21. It is desirable to provide support for multilingual free-form text.57
22. Molecular sequence data are a specialized data type that is highly relevant to current biological research.58
23. Original sequences and sets of aligned sequences are related, but different data types.58
24. The length of individual symbols may be assumed to be constant throughout the sequence, but it may be longer than one letter.58
25. Sequence positions with ambiguous data may be expressed through special symbols rather than requiring a separate syntax.58
26. It is desirable to provide a mapping from ambiguity symbols to the set of fundamental symbols they represent.58
27. Although many characteristics of an object are expressible through categorical variables, more complex data types are often desirable for data such as color or shape. Such data are often expressed using multiple values. Depending on the method the number and semantics of values may be fixed or variable. The descriptive information model should either provide a general model for all conceivable complex data types or should provide extension mechanisms to support the addition of additional complex types. 60
28. It is desirable to support sequences or sets of complex data values, e. g., to record shape variation or a color polygon. 60
29. The data type systems implemented in current descriptive software or exchange formats may dictate secondary requirements where import or export to these systems is intended. 62
30. Quantitative data may occur together with categorical states like "few", "many", "large", or "very small"; a general "indefinite large" and "indefinite small" may form a minimum requirement. A more extensible method may be desirable. 62
31. "Basic properties" according to the Nemisys / Genisys model are a derived type system optimized for morpho-anatomical data. The simple system of 20 basic properties with one level of hierarchy offers pragmatic guidance for structuring such data, but is incomplete and not suitable as a general information model for descriptive data. The selection of quantitative measures not subsumed under "Quantity" or "Number", and the selection of categorical properties not subsumed under "Kind" may be pragmatic for common morpho-anatomical data but is not essential. 66
32. A property classification, preferably with more than one level of hierarchy, is desirable to structure descriptive data. The model should be able to cope with different generalization hierarchies of properties, rather than fixing these. . 66
33. A generalized "property type" system is conceivable, but will be complex and may be expected to be under considerable terminological evolution for an extended period. The information model must be able to support property information in a way that does not affect existing applications relying on the information model. 66
34. The information model should provide means to make description based on different terminologies (using different property choices, different level of decomposition of value types, such as shape) comparable using machine-reasoning. 66
35. Mappings from univariate continuous data to categorical data should be supported. 68
36. A mapping for a category may be based on a single range with two limits, or a list of values or ranges. 68
37. A mapping may be based on single values or several statistical measures. The preferred source for the mapping should be definable. 68
38. Mixed forms of data (some data are quantitative, others are categorical with defined quantitative limits, other categorical with no or ambiguous definitions) should be supported. If fulfilled, this covers automatically requirement 30 (p. 62). 68
39. Mappings from narrow to broad categories should be supported. 69
40. Ambiguous mappings, where one narrow category is mapped to more than one broad category should be supported. 69
41. A hierarchy of categories within a single property or character is expressible with a general

- mapping mechanism and no additional support needs to be added to the information model.....69
42. Special mapping mechanisms are desirable where complex quantitative data are defined.70
43. A complex mapping may help applications provide visualizations of the extent and variability of categorical states to humans. It may therefore be desirable even if no complex quantitative data are actually recorded.....70
44. The failure to define general mappings may point to categorical definitions that are problematic or context-dependent. Whether it is more desirable to have multiple sets of states with exact definitions, or a single set of "generalized" states with multiple, context-dependent mappings remains an open question.....70
45. It is desirable to express the relation between free-form, unconstrained text and descriptive terminology in a special form of mapping. This mapping differs from those discussed so far in that it is defined in descriptive data rather than in terminology. This is more similar to a markup process (like html) than to formalized, mathematical mappings.71
46. It is desirable to be able to define a mapping between an original form of free-form text and its translations.....71
47. Mappings that express the relationship between complex characters ("typologies") and multiple basic or atomic characters are desirable.72
48. Support for calculated character values (based on one or multiple values from other characters) is desirable.....74
49. A standardized support for calculations in a descriptive information model is, however, highly problematic, both because of possibly complex dependencies and because notation systems for formulas are either specific of certain programming languages, or general but difficult to implement in a wide variety of applications. Support for calculated characters is not a priority...74
50. Support for coding status information in the information model is a central requirement to support knowledge management and collaboration scenarios for descriptive data.76
51. The existence of categorical or quantitative data as well as the lack of any data in a description for a character that is defined in the terminology may be considered implicit forms of coding status.76
52. A predefined list of coding status values is desirable to support interoperability. The hierarchical nature of coding status information may be implicit and does not have to be expressed in the data.....76
53. Character dependency definitions are important information items for data entry, character management, and analysis purposes.83
54. A general form of value-dependency may predict values in another character for some (but not all) values of a controlling character. It may be desirable to implement this, but it has not been pursued in current models.....83
55. A special form of value-dependency is that some values predict the applicability of another character (character applicability rules). This is highly desirable and implemented in several descriptive models.83
56. It is desirable that the controlling character may be of categorical or quantitative type. Current models only implement categorical controlling characters.83
57. Because of character evolution issues (adding states to existing characters) and to improve the clarity of expression, both positive and negative character applicability rules ("Applicable-if", "Inapplicable-if") are desirable..... 83
58. Combinations of applicable-if and inapplicable-if rules within a data set are desirable. However, any combination of controlling and controlled character may be covered only by one form of the rule. 83
59. Support for the evaluation of cascading character applicability rules is desirable. This may be expressed in specialized graph structures in the information model, but may also be supported only during evaluation of rules..... 83
60. Structured descriptive information models must provide methods to describe properties of sets of objects. 85
61. Both repeated sample data, and the results of statistical and non-statistical data aggregation methods should be supported. 85
62. Aggregation methods are required for descriptions of classes. In biology, sets of individuals form taxa, sets of taxa form higher taxa. No difference could be detected between aggregating from individual to lowest level class and lower level class to higher level class. 85
63. Aggregation methods are also required for descriptions of individuals, containing either multiple parts or changing over time (discussed in detail further down, p. 93)..... 85
64. The difference between a descriptive information model for individuals (e. g., in a specimen database) and taxonomic classes (e. g., in a taxonomic database) with respect to aggregation methods is negligible. 85
65. Classes or sets of objects may be defined by non-taxonomic means, e. g., through a geographic scope (see also "Secondary classification resulting in description scopes", p. 215)..... 85
66. A fixed sequence of aggregation levels (such as "object part, individual, taxon") is covering only a subset of aggregation cases and should not be part of the information model..... 85
67. Support for range of univariate statistical measures is required. The set of applicable measures depends on data type and measurement scale. 87
68. Most univariate statistical measures report only a single value. 87
69. Some univariate statistical measures such as percentiles or confidence interval limits are best represented as a combination of a result value and a method parameter. 87
70. In addition to exact measures, support for human estimates such as "typical range" is required..... 87
71. To support legacy data such as DELTA, support for undefined measures is desirable (e. g., in DELTA a value is known to be a central value, but not whether it is a single measurement, a mean, median, or mode). 87
72. Some standard descriptive statistics report a collection of data items rather than a single value for a statistical measure. Frequency distributions and distinct value lists must be supported. A distinct value list is a frequency distribution with unknown frequency. 87
73. It is desirable to support two forms of frequency distributions: with frequency values and with frequency categories (i. e. frequency modifiers). ... 87

74. Character metadata informing about the expected scope or recording level of a character may be required. For genetic traits the scope is summarized over the entire life-cycle, for diagnostic purposes individual points in time may be more appropriate.....88
75. In addition, or perhaps alternatively, character metadata informing about dependency of observation on circumstances or temporal development (and therefore the likeliness that data recorded on individuals represent the entire developmental cycle) may be required.88
76. Methods to aggregate aggregated data are highly desirable and can be devised in certain cases. Supporting all necessary data for this in the information model is highly desirable.89
77. Where statistical measures cannot be aggregated further, either access to sample data is required, or the member values may be listed, or in certain cases measures may be degenerated to lower-quality measures that in turn can be aggregated.89
78. In the case of ranges, if the union of ranges has considerable gaps, an aggregation as a set of ranges is more desirable than combining the ranges into a single range.....89
79. Structures for original sample data (sets of values observed under the same conditions) are required on individual object descriptions as well as on class descriptions.90
80. Sample data may be associated with metadata (conditions, time, place, etc.)90
81. The structure of multiple samples (each containing multiple observations) should be preserved.90
82. Repeated sample data should preserve the sampling context and linking of observations (multiple properties observed on the same part or individual).91
83. Aggregation of data should be able to preserve information on linked observations.91
84. Aggregation of data should be able to express the lack of information on linked observations.91
85. "Raw data" is no absolute category that has special properties. Data processing often occurs in multiple steps, each of which may be called to be based on data that are "raw" relative to the results.....93
86. Some data may not be intended for machine-processing at all, but rather provided as "information vouchers". It may be desirable to support this distinction through metadata, but it may also be possible to let a processor assume any data for which it cannot find aggregation or analysis methods, to belong to this category.93
87. Special data aggregation methods may be available for certain types of descriptive data, either truly aggregating data into a new form, or by selecting "representative" data items from the full set. The information model should provide both for linking base data and derived aggregated data, and for selecting some from repeated data as being representative (especially for media data).93
88. The support for data aggregation methods should be extensible, providing for future methods. It is currently unclear how data structures that might be necessary for specialized aggregation methods can be anticipated in the information model.93
89. Aggregation methods are required within individual specimens where observations are repeated over time or due to object parts occurring multiple times. Generalizing from specimen data to taxon data is only a special case of these general aggregation methods. 94
90. Boolean operators connecting descriptive statements that refer to the same property are problematic because they interact with implied semantics (knowledge whether an object part is repeated or not), and the customary data representation of a property. 97
91. The semantics of 'and' or 'or' in natural language descriptions or in DELTA data sets is often ambiguous. It may be desirable to be able to distinguish in the information model between an "ambiguous or" in the sense of one of 'and', 'or', and 'xor', and an 'or' defined in the sense of Boolean logic. 97
92. Boolean operators connecting descriptive statements that refer to different properties have similar problems to those mentioned under requirement 90, p. 97. 99
93. Addressing all potential Boolean combinations in a structured way easily leads to highly complicated models. A simpler requirement may be the support of multiple container levels with a defined operator behavior between them. This remains an open problem in current models. 99
94. For "data aggregation"/"data compilation" it may be desirable to add a feature enabling the documentation of aggregation / compilation source. The model should be flexible enough to provide machine-readable citations for data in the same information system, human-readable citations for external but citable sources, and explicit options to inform on ignorance, perhaps mixed with source references. 99
95. Support for data inheritance is desirable. Inherited (automatically updated) data may have a different level of reliability, and their nature must be communicated to the data consumer. 99
96. A distinction between manually compiled (aggregated and reviewed) aggregation data, and inherited (i. e., automatically compiled) aggregation data is desirable. This may be a metadata item on the data. 100
97. Support for deductive data inheritance from descriptions of higher classes to lower classes (or individuals) is highly desirable. 101
98. Support for deductive data compilation is *not* desirable. 101
99. Information whether an information model does or does not support data inheritance mechanisms may be important to assess data quality, especially whether data might have been copied downwards to improve the operation of identification tools. 102
100. Whether inheritance needs to be broken at the source (e. g., "do not allow this to be inherited") or the destination (e. g., "do not inherit from above, even if missing") by means other than adding data needs further study. It may be desirable, but complicates the system and no good example cases could be found. 102
101. Support for data inheritance and deduction removes the need to support defining "implicit states" in the terminology. Data using the "implicit states" model are convertible into a hierarchical data inheritance model. 103
102. For most purposes the "character matrix", "character state matrix", or "character state list" models are equivalent. For actual data exchange (especially when considering federated relational

- databases or XML formats) a list model may be the most flexible choice. 109
103. One area requiring different considerations in the three models is coding status and character dependency. 109
104. Whether information that states are considered "absent" or "false" should be preserved as data (and aggregated or inherited along the taxonomic hierarchy) is contentious. For the primary purposes of representing the descriptive data this is not necessary. However, a number of important secondary purposes exist, under which preserving this information may be valuable (negative statements, collaboration and discussion, evolution of terminology). In principle such information may be stored in all three models, but the state matrix model may be the most intuitive for this purpose. 109
105. If negative statements are supported, it may be desirable to *not* support certainty modifiers on these. 109
106. Dependency rules or "do-not-code" (also called "out-of-scope") rules controlling single states instead of entire characters are probably *not* desirable. 109
107. The number of statistical measures is large and no general agreement exists on a small subset to fit all purposes for which descriptive information models are intended. The various existing denormalized models all use different measures. A flexible model able to store a larger number of different statistical measures is desirable. 113
108. The statistical measure model should be extensible and offer generalizations that allow applications to support classes of statistical measures, rather than only individual measures. 113
109. The fundamental applicability of statistical measures depends only on the data type (measurement scale and continuous / discrete), but not on individual characters. As a consequence, no schema evolution issues exist when new statistical measures are added to the terminology. 113
110. The designer of the terminology should be able to limit the statistical methods available to data entry personnel. This leads to more concise data entry forms and can reduce errors. 113
111. Statistical measures are fundamentally applicable to all characters. If no data storage problems prevent this, it may be desirable to view the limitation of measures as a "recommendation" or "secondary filter", affecting only the primary data entry form rather than data storage. 113
112. No equivalent to explicit state-absent statements (which may be desirable in categorical data, see requirement 104) occurs with quantitative data. 113
113. A general order of characters and a general order of character states within a character are meaningful for communication with humans, even where it is not meaningful for machine interpretation or analysis (e. g., states on the nominal scale). 116
114. For characters, multiple alternative ordering definitions are desirable. 116
115. Negative requirement: It is not necessary to preserve, in a given description, the order in which data relating to different characters have been entered. 116
116. In a given description and character, the order in which multiple values or states have been entered may have to be preserved. This is unequivocal for repeated measurements in sample data, but restricted to special situations in summary data. . 116
117. In a given description and quantitative character, multiple occurrences of a statistical measure may have to be preserved in sequence (some models use this as a replacement for sample data). 116
118. In a given description and categorical character, it may be desirable to provide a method to let data set authors decide whether the sequence of multiple states may be rearranged according to the sequence in the terminology, or whether it is to be preserved. 116
119. When reordering the states in a given description and character, modifiers for which order has been defined as semantic (ranked modifiers) may have precedence over the state order. 116
120. It is desirable that the information model encourages distinguishing sample data and summary data in an unambiguous way, e. g., by preventing unqualified repeated occurrences of the same value or state in summary data. States with different modifiers or annotations, however, have to be accepted. 116
121. Summary statement: The Prometheus description model has very special requirements on the information model. It elaborates and modifies the concepts of the Nemisys / Genisys model. It is implemented and tested. The extent to which this model is specific to certain kinds of data needs to be assessed as experience with the model grows. 122
122. Summary statement: The Prometheus description model provides for the definition of a subset of all possible object-part / property combinations for data entry. For different projects, different sets of "enabled" object-part / property combinations may be defined. The union of all enabled selections is roughly equivalent to characters in character or character state matrix models. 122
123. A character may depend on more than one object part or on more than one property. A possibility to express this, either in descriptive terminology or in descriptive data, is desirable. 123
124. Some "relational characters" may be viewed as calculated characters. The multiple parts involved may then simply be discovered by analyzing the characters involved in calculations. However, often only values for the calculated, but not the base characters are available. The model should thus support analysis of multiple part-relations even if only the calculated character values are present. 123
125. For many taxonomic groups the character decompositions beyond object part and property are desirable. Examples are experimental conditions, measurements methods and instrumentation, and information representation (e. g., quantitative versus categorical representations). 125
126. Concept hierarchies that are superimposed on a flat list of character may be a desirable alternative to strict character decomposition models. 130
127. The combination of concept hierarchies with a flat character list is desirable when the support of existing ("legacy") data is a requirement. Concept hierarchies may be modeled as an optional part of the information model, whereas strict character decomposition models require decomposition information to be available to handle descriptive information. Concept hierarchies provide a large amount of the organizational and semantic advantages of character decomposition models without breaking compatibility with existing data.. 131

128. Multiple concept hierarchies are desirable to express – in addition to object-part and property classification – also aspects of methodology, instrumentation, or simply arbitrary character subsets / filters. 131
129. Morphological object composition includes aspects (multiplicity, adjacency, order) that are not immediately included in a part-of hierarchy. Support for these aspects is desirable. 133
130. Anatomical (inward) composition hierarchies and are not necessarily nested inside a morphological (outward) composition hierarchy. They can therefore not be displayed in a single tree and support for multiple composition hierarchies is a requirement. 134
131. Mechanisms to express dependency relations between multiple composition hierarchies may be desirable. 134
132. Whether physical objects should be considered atomic or a composition depends on perspective and conventions, and may depend in complex ways on interaction with other compositions and properties. It is desirable to add mechanisms that help in communicating the perspective and conventions between designer and consumer or a descriptive terminology. 136
133. An object composition may often be considered a property of the parent object. The information system needs mechanisms to relate (or “map”) property and object composition expressions. 136
134. Multiple morphological concepts and corresponding object composition hierarchies may exist. It is desirable to support alternative concepts of object parts and composition hierarchy. 137
135. The conventions whether something is considered a property or a composition, or which composition hierarchy should be preferred often depends on context, especially taxonomic scope. 137
136. The object-part-composition of individuals and classes may be expressed as part of the description (using characters or properties, depending on the description model). 141
137. The classical requirement is to record in descriptions whether a part is present. The hierarchical relations of the part composition are left to the terminology domain. 141
138. Whether a generalized object-part-composition hierarchy indeed belongs into the terminological domain or may be better placed in the description domain remains an open problem and needs further research. 141
139. In addition to object composition, the multiplicity of a composition must be supported in the information model. 142
140. It is not required to support composition and multiplicity information as part of the definitions of object parts in terminology. 142
141. Object-oriented practices to represent multiplicity / cardinality in a composition cannot easily represent variability of cardinality in a description (e. g., “3-7 leaflets per leaf”). 143
142. Representing variability of multiplicity / cardinality in a composition through a collection of instances may require a huge number of instances, making this probably impractical. 143
143. In addition to quantitative expression of multiplicity in object compositions, also categorical expressions such as “many” or “≥ 20” must be supported. 144
144. This issue is not a question of class versus object descriptions; the need for categorical multiplicity ranges arises even in individual objects where the composition is in principle countable. 144
145. Often more than one category is used (e. g., “few” / “some” / “many”). 144
146. Object composition multiplicity may be a mix of quantitative (1, 2, 3, ...) and categorical expressions. 144
147. Whereas a single category “beyond countability” may relatively easily be supported by quantitative data types, multiple categories with more or less well-defined ranges require more complex data structures. 145
148. Expressing multiplicity of object composition through a mixture of instance composition and values of categorical properties requires complex reasoning algorithms, interpreting values of instance properties as well as the instance multiplicity itself differently, depending on whether categorical multiplicity properties are present or not. 145
149. Methods to qualify multiplicity in object compositions as being uncertain or express that multiplicity is unknown are required. 145
150. Multiplicity in object compositions may be expressed in attributes of child objects. These have special semantics and metadata to recognize them are desirable. 147
151. A combination of multiple child objects (if child objects differ) and multiplicity attributes may be desirable. 147
152. Concepts to fix the absolute orientation of physical objects in space are an important means to facilitate object recognition. 147
153. In object compositions the relative orientation of physical objects is an important concept to facilitate object recognition and should be supported in the information model. 148
154. In object compositions, symmetry is an important concept to facilitate object recognition and should be supported in the information model. 149
155. Spatial gradients may interact with object composition, depending on properties or multiplicity of child objects (parts of main object). 150
156. Spatial gradients usually interact with absolute or relative object orientation; the data or terminology model must allow for this. 150
157. Adjacency of object parts in a composition is an important concept that is desirable to be supported by the information model. 152
158. The concept of “object location” is a synthesis of absolute and relative orientation, symmetry, adjacency, and sequences. 152
159. Generalization and composition are distinct forms of relations that have different properties and lead to different conclusions. For physical objects (parts of the described objects) the information model must support both a composition and generalization hierarchy. 155
160. Multiple generalization perspectives exist (e. g., phylogenetic, functional, morphological similarity, or compositional similarity) and must be supported in the model. 156
161. If generalization hierarchies support directed acyclic graphs, a single graph may incorporate the generalization hierarchies for multiple perspectives. However, for the clarity of expressions clearly labeled separate graphs may be preferable. 156

162. The recognition of object parts and the preferred name for these in biology depends in complex ways on character properties, developmental stages, and the taxonomic classification of the organism. In an identification context this information is often available only after identification success ("post-recognition problem"). Support for generalization concepts to overcome this problem is required. 159
163. Object recognition may fail in predictable patterns; support for knowledge about common misinterpretations of object parts is desirable. 160
164. The taxonomic hierarchy itself is a generalization hierarchy placing entire organisms in classes. Generalizations of object parts and entire objects are related. Even composition hierarchies of organisms exist. However, despite the similarities, a special data structure for the taxonomic hierarchy is desirable because of the special role taxa play in evolution itself and in the management of biodiversity knowledge. 162
165. In addition to the common composition (part-of) and generalization (kind-of) relations, relations expressing change over time (ontogenetic, life cycle, evolutionary history) are desirable in descriptive information models designed for biological objects. 163
166. What is considered a property is subject to conventions. Complex properties exist that may also be expressed as a set of more atomic properties. A conversion / mapping functionality is desirable. 165
167. Patterns are especially problematic situations that may be modeled through properties or compositions. Patterns are highly relevant to the description of biological objects and adequate support for them is required. 167
168. Depending on property values, other properties may or may not be applicable. Support for character dependency rules is desirable. This may be in the form of character applicability rules (compare "Character applicability rules", p. 76) or more general property dependency rules (which would be applicable to multiple objects). 169
169. Values in different properties may be comparable or not, depending on other property values. 169
170. In addition to dependency definitions, analogous renaming rules (making labeling dependent on taxonomic scope) may be desirable; this requires further study. 169
171. Even seemingly trivial observations require a definition of an observation method. Support for defining this method – either as a complex method description, or broken down into components such as conditions, instrumentation, operating procedures, and conversion and recording procedures – is highly desirable. 170
172. The separation between experimental conditions, conditions under which material is sampled from the natural environment, and conservation or special sample processing conditions is not always sharp. It may therefore be desirable to support information in a generalized "measurement conditions" category. 172
173. Support for "instrumentation" concepts, a highly reusable part of methodology, is desirable. 173
174. The concept of instrumentation may include basic default operating procedures. 173
175. Generalization and composition hierarchies of instrumentation (such as "forestry field instrumentation") are desirable. 173
176. Complex relations exist between instrumentation; it may be desirable to model these dependency relationships unless this leads to an overly complicated information model. 173
177. In addition to generic concepts of measurement conditions and instrumentation, support for measurement procedures specific to object parts, properties, and interactions with conditions and instrumentation is required. 174
178. The form of data returned by a measurement method may differ from the form expected in data recording. Support to define and document conversion and recording procedures is desirable. 175
179. Dependencies between the observation circumstances, conditions, instrumentation, etc. and characters available for comparison and identification is an important aspect of the use of descriptive data. It is essential for branching (e. g., dichotomous) keys, and beneficial for multi-access keys. 176
180. It may be desirable to record "observation circumstances / conditions" as part of archiving identification data. This may point to modeling the method dependency as a character dependency based on special "observation condition" characters. (Alternatively, method dependency may be modeled through the method concept hierarchy / ontology.) 176
181. Abstract properties and methods interact in complex ways that should be addressed in the information model. Whether a "Part-Property-Method decomposition model" or multiple concept hierarchies superimposed on fixed character concepts are preferable needs further analysis and testing. 179
182. The information model should support management and curation of the descriptive terminology independently of the descriptive data itself. 180
183. It is desirable to enable curation of different parts of the terminology by different organizations, in different systems. 180
184. Supporting managed federations is desirable. This may require some data items supporting management procedures. These are, however, difficult to specify because they strongly depend on local management practices. 181
185. It is desirable to support a combination of locally defined and multiple externally defined (standardized) terminology modules. 182
186. It is desirable to distinguish between locally developed terminology modules proposed for external use, and terminology modules that are considered to be too instable or poorly developed for such use. 182
187. It is desirable to support extending external standard terminology modules with local information. The essential definition should not be changed, but it may be extended through labels or definition text in the local language. Furthermore other information affecting presentation or assumptions for analysis purposes may be desirable to extend or change locally. 183
188. It is desirable to express the scope of terminology modules relative to the taxonomic hierarchy. Application may use this information to manage availability of terminology items for different taxa. 185
189. Terms in the terminology modules should be identified by GUIDs. 188
190. It is desirable that the relation between locally defined terms and external standard terminology

- modules can be expressed through GUIDs. The relation may be of several kinds: e. g., “copied from template”, believed to be “similar” or “essentially identical” 188
191. For biological objects, the primary classification of organisms by taxonomic name needs to be supplemented by additional concepts to describe the scope of a descriptive data set. Lacking a better concept, these may be called “secondary classifiers”. Secondary classifiers summarize a special form of correlated variation that is independent of the taxonomic classification. 227
192. Handling secondary classifiers as an extension of the primary taxonomic hierarchy (i. e., below infraspecific ranks) is theoretically possible, but leads to severe artifacts and is not recommended. 228
193. Secondary classifier concepts are not limited to sex and life cycle, seasonal, or developmental stages. Many further general concepts (castes of social insects) or highly desirable “custom” classifiers (like spore states of rusts) exists. A generalized concept is required. 228
194. Secondary classifiers are required in the following contexts of an information model:
- When defining the scope of a coded or natural language description.
 - When defining the scope of an identification key.
 - When defining the scope of an identification key result (“keyed-out taxon”).
 - As part of a specimen identification information in observation or collection databases. 228
195. Secondary classifiers are distinct from characters. Classifier-related characters exist, but:
- classifier-related characters can usually not be calculated based on classifier values;
 - classifier-related characters require no special handling in identification. 228
196. Dependency relations between secondary classifiers and characters exist:
- Classifier-related characters may control the valid values for classifiers (see heterostyly example);
 - Classifiers may control characters (e. g., only part of the life cycle stages may have sexual differentiation). 228
197. The presence of some secondary classifier concepts (especially developmental stages) depends on the taxonomic group (i. e., the “primary classifier”)..... 228
198. Secondary classifier values of a given description may be unknown. A description may either be general (e. g., apply to both sexes) or the scope may be unknown (e. g., it is unknown whether the description applies to one or both sexes). Support for coding status values or a similar concept is desirable..... 228
199. Secondary classifiers require representations for multiple audiences / languages..... 228
200. Although the analysis was limited to biological objects, some secondary classifiers (e. g., geographical scope) may occur outside of biology as well. Thus, although the priority may be lower for non-biological applications, secondary classifiers are required for all descriptive information models..... 228
201. The information model needs special data structures for branching keys. These are *required* for authored (or “designed”) keys. They may also be *desirable* to store (cache) algorithmically generated branching keys for fast retrieval. 258
202. Metadata supporting such a distinction between authored and algorithmically generated keys may be desirable, including information about last update for algorithmically generated keys..... 258
203. Authored branching keys need certain metadata like name, title, description, expected experience level of user (untrained, generally trained, specialist), and information on available languages. The metadata for a branching and multi-access key are generally identical with those of a set of coded descriptive data. For key operation, however, additional metadata may be required (compare requirement 229, p. 276). 258
204. Dichotomous keys are a special case of polytomous keys and may be stored in a model for the latter. The author of a key may desire to indicate that a key should remain strictly dichotomous; this may be stored as metadata item specific to branching keys. Key builder applications (“editors”) may recognize and either warn or prevent the user from adding more than two leads. The distinction is, however, not considered central enough to warrant enforcement by the information model itself. 258
205. Combinations of broad branching keys and groups of browsable descriptions / illustrations – commonly found in field guides – can be modeled as a special case of polytomous keys, with a high number of lead choices at the end. However, the information model must support empty lead text and perhaps a metadata item requesting “embedding” of descriptions or illustrations instead of the usual linking. 258
206. Leads in branching keys may lead to other couplets, entire keys (“subkeys”), or taxon names / descriptions. The choice is a strict alternative in most cases (but see requirements 207 below and 220, p. 262). 258
207. The “result-and-continue pattern” (compare p. 235) implies requirements that may lead to either supporting both a taxon group and a following couplet at the end of a lead, or presentation metadata indicating that a subkey shall be embedded in a higher-level key. Exactly how to support this situation needs further study. 258
208. Branching keys may provide for redirection (“cross-linking”, “reticulation”). As a consequence, the information model cannot be limited to a tree, but must support directed acyclic graphs (DAGs). 258
209. It is desirable that branching keys may support the question / answer and the lead style. The question / answer style requires an additional “question label” at each couplet. 258
210. The question / answer style may also occur in multi-access keys, requiring an additional “question label” for each character. This item is separate from the question in branching keys, because a couplet in branching keys may involve multiple characters. 258
211. The text in branching keys (question / answer or lead style) should be free-form text and provide for minimal inline text formatting such as italics, bold, sub- and superscript. 258
212. Multiple keys for a single set of taxa should be supported (for a plant family, for example, separate keys based on flowers, fruits, and vegetative organs might be desirable). 258
213. Multilingual support for branching keys is desirable, especially for complex keys intended for ongoing revisions. Additional metadata on default language, fall-back language etc. may be desirable. 258

214. Media resources (images, etc.) are required at all nodes in an identification key, not only on terminal nodes. At higher nodes they may either illustrate diagnostic features, or support the concepts of “looks like” / “promorphs”. 258
215. Media resources require context-dependent captions. The same resource may be used at different points in the key, illustrating different character concepts or the entire resulting taxon. ... 258
216. Presentation styles of branching keys may be supported as stored preferences in the metadata, but are not an issue of the required data structures. 259
217. Tabular keys may have additional requirements (e. g., a fixed set of few characters and sequence of characters determining the sort sequence) that cannot be represented with the polytomous branching key model. Whether this warrants an independent model, or whether it can be included in a model for character guidance needs further study..... 259
218. The confirmation phase (computer-aided choice of similar taxa) may be based on algorithmically determined similarity, or it may be based on manually entered lists of “easily confused taxa”. In the latter case, data structures for this must be presented in the information model..... 259
219. Both branching and multi-access keys may point to another key (subkey) rather than to a taxon. Subkeys often are associated with taxonomic ranks (order family, genus keys) but may be dominated by other scopes (e. g., “shrubs in winter”, “broad-leafed trees”)..... 262
220. More than one subkey may be desirable at a single result-lead. 262
221. Descriptive data for multiple algorithmically created keys (branching or multi-access) may be kept in a unified matrix; in addition support for pointers from one key to independently developed related keys is required. 262
222. Support for transferring information between branching and multi-access keys is desirable. 264
223. Optional support for coded data reflecting the proposition made in the lead of a branching key is desirable. This may have the form of markup of the natural language lead text or of a coded description associated with a lead. 264
224. Support for Boolean operators and nesting is required for both alternatives (markup of natural language lead text or associated coded descriptions). 264
225. Character-ranking metadata, expressing various optimality criteria for identification or analysis, are important data elements..... 270
226. Character-ranking metadata should be flexible to support various ranking categories (or “topics”)... 270
227. Both interoperable and application-specific ranking metadata may be required. 270
228. Support for missing data in general, and specifically coding status “not to be coded”, is desirable to support guidance in character selection. This supports requirement 50, p. 76. ... 270
229. It is desirable to store parameters of character guidance algorithms (like DELTA VaryWt, Rbase, Reuse) as key-metadata. These data are not descriptive data, but specific to a given algorithm and key (authored branching key or multi-access key data set)..... 276
230. It remains inconclusive whether the information model must support metadata on taxa resulting in a preferential treatment in a key. Such data could express the intent that some taxa are keyed out faster than others. If supported, the model should allow different values for different audiences and key algorithms. 276
231. Support for metadata on the level of data sets (or “projects”) is desirable. This may include intellectual property rights (IPR; including authorship, ownership, copyright, and licenses), coverage and scope (taxonomic group, geographical range, perhaps seasonal applicability), version numbers, initiation and modification dates, etc..... 281
232. Some data-set-level metadata may be calculable from individual objects (contributors, last modification, coverage) others not (editors, scope, etc.). Flagging data as having been, or to be automatically updated may be desirable. 281
233. To improve communication about identification processes, the detailed descriptive data created during identification processes may be permanently stored in “IdentificationBanks” and made citable by issuing globally unique “identification accession numbers”..... 296