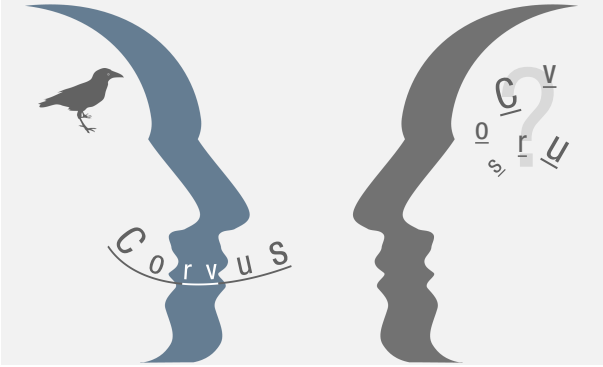
# SPRACHZENTRIERTE ANSÄTZE ZUR STEIGERUNG DER AKZEPTANZ VON GESCHÄFTSPROZESSMODELLEN




DISSERTATION

Von der Universität Bayreuth
zur Erlangung des Grades eines
Doktors der Naturwissenschaften (Dr. rer. nat.)
genehmigte Abhandlung

von

LARS ACKERMANN
aus Dresden

1. Gutachter: Prof. Dr.-Ing. Stefan Jablonski
2. Gutachter: Prof. Dr. rer. pol. Ulrich Frank

Tag der Einreichung: 13.10.2017
Tag des Kolloquiums: 01.03.2018

Für meine Familie.

# ZUSAMMENFASSUNG

Die Modellierung von Geschäftsprozessen hat sich als adäquates Mittel etabliert, um Wertschöpfungsketten in Unternehmen zu dokumentieren, zu analysieren und in gleicher oder verbesserter Qualität zu wiederholen. Dafür steht eine Vielzahl von teils sehr unterschiedlichen Prozessmodellierungssprachen zur Verfügung – eine Tatsache die zumindest zweierlei Schlüsse nahelegt: Einerseits ist keine der Sprachen als „Nonplusultra"-Standard akzeptiert, was andererseits zu Sprachklüften führen kann. Man unterscheidet unter anderem zwischen imperativen und deklarativen Sprachen. Erstere gestatten nur explizit im Modell beschriebene Prozessverläufe, während letztere alle Abläufe erlauben, die nicht gegen die im Modell kodierten Regeln verstoßen. Unabhängig von dieser Klassifikation unterscheiden sich die Sprachen auch hinsichtlich ihrer Fähigkeit, einen Prozess aus verschiedenen Perspektiven betrachten zu können. Vielfalt und Verschiedenartigkeit der Prozessmodellierungssprachen haben in bestimmten Situationen negative Einflüsse auf die Verwendbarkeit der Modelle. Dies betrifft unter anderem ihre Verständlichkeit bei Unkenntnis der verwendeten Modellierungssprache, ihre Interoperabilität mit anderssprachigen Systemen, die damit verbundene Abwärtskompatibilität bei einer Sprachevolution und die Vergleichbarkeit mit Modellen anderer Sprachen.

Die vorliegende Arbeit fokussiert sich vordergründig auf die Konzeption von Werkzeugen, welche die Probleme der Sprachvielfalt abschwächen oder eliminieren. Dies kann zukünftig die Akzeptanz einzelner und andererseits die Kombination mehrerer Sprachen verbessern. Das erste der drei Werkzeuge, eine Simulationstechnik für multi-perspektivische, deklarative Prozessmodelle, ist in der Lage Beispiele valider und nicht-valider Prozessverläufe zu generieren. Dies kann das Verständnis des Modells und auch den Vergleich desselben mit einem zweiten Modell erleichtern. Das zweite Werkzeug, eine Technik zur Generierung natürlichsprachlicher Prozessbeschreibungen aus multi-perspektivischen, deklarativen Prozessmodellen, hebt formalsprachliche Barrieren auf und erfüllt somit denselben Zweck. Das letzte Werkzeug, ein generischer Translationsansatz für Prozessmodelle, kann zusätzlich die Kompatibilität eines Modells mit anderssprachigen Systemen herstellen. Weiterhin kann dieselbe Technik durch Translation in ältere Sprachversionen für Abwärtskompatibilität sorgen.

Vergleichbare Simulationstechniken operieren ausschließlich auf imperativen oder mono-perspektivischen, deklarativen Prozessmodellen und sind oft nicht in der Lage, Beispiele für nicht-valide Prozessverläufe zu erzeugen. Translationstechniken für Prozessmodelle sind rar und zudem sprachspezifisch. Untersuchungen bezüglich der Generierung natürlichsprachlicher Texte konzentrieren sich im Kontext des Prozessmanagements bislang lediglich auf imperative Prozessmodellierungssprachen. Durch Anwendungen der Techniken auf verschiedenartige Beispielmodelle sowie eine qualitative Umfrage werden ihre Funktionalität, Anwendbarkeit und Nützlichkeit belegt.

## ABSTRACT

Business process management has established as a suitable means to document and analyze enterprise value chains as well as to repeat them in a steady or improved quality. For that purpose, a variety of often highly diverse process modeling languages is available. This suggests at least two sorts of conclusions: On the one hand none of the languages is accepted as the "ne-plus-ultra"-standard and, on the other hand, this often leads to language barriers. In general, we distinguish between imperative and declarative process modeling languages. Imperative languages describe valid process execution paths explicitly while models conforming to a declarative language implicitly permits those execution paths that do not violate any of the modeled rules. Irrespective of this classification the languages differ in terms of their capability to consider the process from different perspectives. In certain situations, the variety and diversity of process modeling languages has an adverse effect on the usefulness of process models. This refers to, for instance, the intelligibility of a model in the case that the user is unfamiliar with the modeling language, its interoperability with "foreign-language" process execution systems, the related backward compatibility in the case of language evolution as well as its comparability with models conforming to other languages.

This thesis mainly focuses on the conceptual development of approaches that mitigate the difficulties caused by the diversity of process modeling languages. In future these approaches can improve the acceptance of individual languages and their combinations. The first approach is a simulator for multi-perspective declarative process models that is able to generate artificial logs of valid and invalid process execution paths. These logs can be utilized to improve both the comprehension of a process model and its comparability with a second model. The second approach, a generator for process descriptions in natural language based on multi-perspective declarative process models, removes formal-language barriers and, therefore, produces the same effect. The third approach is a generic translator for process models and, hence, can establish the compatibility between a process model and language-specific process execution systems. Since migration is a special form of translation the same tool can be used to bridge gaps between language versions.

Comparable simulation approaches exclusively operate either on imperative or on mono-perspective declarative process models and are usually unable to generate examples for invalid process execution paths. Translators for process models are rare and mostly language-specific. In the context of business process management, research regarding the generation of natural language texts focuses on imperative process modeling languages. The functionality, feasibility and usefulness of the approaches are evaluated by their application to various exemplary models and through a qualitative survey.

# THEMENBEZOGENE PUBLIKATIONEN

Die in der vorliegenden Arbeit vorgestellten Ansätze wurden anteilig der wissenschaftlichen Gemeinschaft kommuniziert und auf einschlägigen Konferenzen im Bereich der Informationssysteme für Geschäftsprozesse diskutiert. Die Inhalte dieser Dissertation bauen zu Teilen auf den bereits publizierten Ansätzen auf, präsentieren diese jedoch in einem höheren Detailgrad, betrachten den Kontext im größeren Rahmen und werden durch ausführlichere empirische Untersuchungen belegt.

Bisher sind die folgenden thematisch relevanten Arbeiten veröffentlicht:

- Natural Language Generation for Declarative Process Models.
  L. Ackermann, S. Schönig, M. Zeising, S. Jablonski, 11th International Workshop on Enterprise & Organizational Modeling and Simulation, Springer, Schweden, 2015.

- Simulation of Multi-perspective Declarative Process Models.
  L. Ackermann, S. Schönig, S. Jablonski, 12th International Workshop on Business Process Intelligence, Springer, Brasilien, 2016.

- MuDePS: Multi-perspective Declarative Process Simulation.
  L. Ackermann, S. Schönig, BPM Demo Track, CEUR, Brasilien, 2016.

- Towards Simulation- and Mining-based Translation of Process Models.
  L. Ackermann, S. Schönig, S. Jablonski, 12th International Workshop on Enterprise & Organizational Modeling and Simulation, Springer, Slowenien, 2016.

- Towards Simulation- and Mining-based Translation of Resource-aware Process Models
  L. Ackermann, S. Schönig, S. Jablonski, 1st Workshop on Resource Management in Business Processes, Springer, Brasilien, 2016.

- DPIL Navigator 2.0: Multi-Perspective Declarative Process Execution
  S. Schönig, L. Ackermann, S. Jablonski, BPM Demos, CEUR, Spanien, 2017.

In der erstgenannten Publikation werden wesentliche Teile der Technik zur Generierung natürlichsprachlicher Beschreibungen auf Basis von Prozessmodellen beschrieben. Die zwei sich anschließenden Arbeiten stellen den Simulationsansatz für deklarative Prozessmodelle vor. In den beiden letzten Arbeiten werden schließlich das Grundprinzip der Translationstechnik sowie sprachklassenspezifische Aspekte desselben diskutiert. Die letzte Veröffentlichung beschreibt ein Prozessausführungssystem für multi-perspektivische, deklarative Prozessmodelle, welches eine mögliche Zielplattform für die Integration des vorher genannten Simulationsansatz darstellt.

# INHALTSVERZEICHNIS

# ABBILDUNGSVERZEICHNIS

# TABELLENVERZEICHNIS

# CODEAUSSCHNITTE

# ABKÜRZUNGSVERZEICHNIS

| | |
|---|---|
| AECME | European Association of Aerospace Industries |
| API | Application Programming Interface |
| AST | Abstrakter Syntaxbaum |
| BLEU | Bilingual Evaluation Understudy |
| BPDM | Business Process Definition Metamodel |
| BPEL | WS-Business Process Execution Language |
| BPMN | Business Process Model and Notation |
| BTM | Bedeutung-Text-Modell |
| CLIMB | Computational logic for the verification and modeling of business constraints |
| CMMN | Case Management Model and Notation |
| CPN | Coloured Petri Net |
| CPN ML | Coloured Petri Net Standard MetaLanguage |
| CRM | Customer Relationship Management |
| DCR | Dynamic Condition Response graph |
| DPIL | Declarative Process Intermediate Language |
| DRL | Drools Rules Logic |
| DSL | Domänenspezifische Sprache |
| DSyntR | Deep-syntactic Representation |
| ECA | Event Condition Action |
| EMF | Eclipse Modeling Framework |
| EPC | Ereignisgesteuerte Prozesskette |
| ERP | Enterprise Resource Planning |
| EZA | Endlicher Zustandsautomat |
| FOL | Prädikatenlogik erster Stufe |
| FC | Flow Chart |
| KPI | Key Performance Indicator |

| | |
|---|---|
| KpPQ | Kompetenzzentrum für praktisches Prozess- und Qualitätsmanagement |
| LTL | Lineare temporale Logik |
| M2MT | Modell-zu-Modell-Transformation |
| M2T | Modell-zu-Text-Transformation |
| MOF | Meta Object Facility |
| Mof2Text | MOF Model to Text Transformation Language |
| MXML | Mining eXtensible Markup Language |
| NIST | National Institute of Standards and Technology |
| NLG | Natural Language Generation |
| NNF | Negationsnormalform |
| OCL | Object Constraint Language |
| OMG | Object Management Group |
| OOP | Objektorientierte Programmierung |
| PBS | Prozess-bewusstes Informationssystem |
| PoI | Percentage of Instances |
| PoE | Percentage of Events |
| PN | Petri-Netz |
| PT | Prozessbaum |
| PSL | Process Specification Language |
| RAD | Role Activity Diagrams |
| RST | Theorie rhetorischer Strukturen |
| SAT | Erfüllbarkeitsproblem |
| SBVR | Semantics Of Business Vocabulary And Rules |
| TLA+ | Temporal Logic of Actions Specification Language |
| UML | Unified Modeling Language |
| UML AD | UML Aktivitätsdiagramm |
| VPSB | Verfeinerter Prozessstrukturbaum |
| WDF | Wahrscheinlichkeitsdichtefunktion |
| WfMS | Workflow Management System |

XES IEEE Standard for eXtensible Event Stream

XMI XML Metadata Interchange

YAWL Yet Another Workflow Language

ZEL Zukunftsereignisliste mit Ereignisankündigungen

# Teil I

# EINFÜHRUNG, PROBLEMSTELLUNG UND ÜBERBLICK

Im ersten Teil der Arbeit werden verschiedene Aspekte und Aufgaben des Geschäftsprozessmanagements zusammengefasst. Hierbei liegt der Fokus auf den Berührungspunkten mit Prozessmodellierungssprachen. Deren Vielfalt und unterschiedliche Ausrichtung bergen Hürden für die Verständlichkeit, Interoperabilität, Migration und den Vergleich von Modellen zur Beschreibung von Prozessen. Besonders die zwei grundsätzlich verschiedenen Paradigmen der imperativen und der deklarativen Modellierung im Zusammenspiel mit diesen Problemfeldern steht im Fokus der vorliegenden Arbeit und der ebenfalls in diesem Teil skizzierten Lösungsansätze.

# 1

# GESCHÄFTSPROZESSMANAGEMENT UND SPRACHBEZOGENE AKZEPTANZPROBLEME

Menschen und Unternehmen sind seit Jahrzehnten Teil einer weltweit vernetzten IT-Landschaft. Durch die Industrie-4.0-Initiative und vergleichbare Konzepte soll die Produktionstechnik *direkter* von dieser Infrastruktur profitieren, indem physische Produktionssysteme befähigt werden, IT-gestützt mit anderen Systemen oder Menschen zu kommunizieren. Die damit erzielbare Steigerung der Automatisierung erfordert gut strukturierte, dokumentierte Arbeitsabläufe – sowohl im Zusammenspiel mit anderen Systemen als auch bei menschlichen Beteiligten. Die dadurch gesteigerte Effizienz und Nachvollziehbarkeit von Arbeitsabläufen ist heute nicht nur in produzierenden, sondern auch in dienstleistungsorientierten Unternehmen eine Grundvoraussetzung für wirtschaftlichen Erfolg.

*Strukturierte Arbeitsabläufe und wirtschaftlicher Erfolg*

## 1.1 GESCHÄFTSPROZESSMANAGEMENT

Am Übergang vom sogenannten industriellen zum digitalen Zeitalter begann ein Umdenken hinsichtlich der treibenden Erfolgsfaktoren für Unternehmen [113]. Bis Mitte des zwanzigsten Jahrhunderts fokussierte man sich hauptsächlich auf die Optimierung herstellender Tätigkeiten. In den 1960er Jahren gewannen dann auch Technologien, beispielsweise moderne Schreibmaschinen, zunehmend an Bedeutung. So ebnete unter anderem *IBM* mit der Schreibmaschinen-Modellreihe *IBM Selectric* [25] den Weg für die Moderne Textverarbeitung[1]. Im Laufe der Jahre wandelten sich die rein mechanischen zu elektromechanischen Schreibgeräten mit Datenspeichern. Später wurden diese mittels innovativer Hardware- und Softwareschnittstellen mit Rechnersystemen gekoppelt, was den Beginn der digitalen Textverarbeitung einleitete. Mit der steigenden Verbreitung und Verwendbarkeit von Rechnersystemen wurde dieser Trend in den folgenden zwei Jahrzehnten noch verstärkt. Ein treibender Faktor hierbei war die Erleichterung administrativer Abläufe in den Unternehmen. Dokumente konnten auf digitalem Weg gespeichert, korrigiert und ausgetauscht werden. Zusammen mit modernen Kommunikationstechnologien und Softwarelösungen entwickelten sich *Prozesse*, welche nicht nur die Interaktion zwischen menschlichen Beteiligten, sondern auch zwischen menschlichen Beteiligten und unterstützenden Technologien umfassen. Damit verdrängte die Optimierung dieser Arbeitsabläufe die Optimierung herstellender Tätigkeiten aus dem Fokus. In drei Phasen gelangte man so bis zum heutigen Tag von der Prozessverbesserung über das Prozess-Re-Engineering zum sogenannten *Geschäftsprozessmanagement*.

*Administrative Abläufe*

*Prozesse*

*Geschäftsprozesse und Geschäftsprozessmanagement*

Als *Geschäftsprozess* wird eine Reihe von festgelegten Tätigkeiten bezeichnet, die von menschlichen oder maschinellen Akteuren durchgeführt werden, um für ein

1 Einer der bekanntesten heutigen Drucker-Hersteller, *Lexmark International, Inc.*, ist eine Ausgründung der IBM-Schreibmaschinensparte.

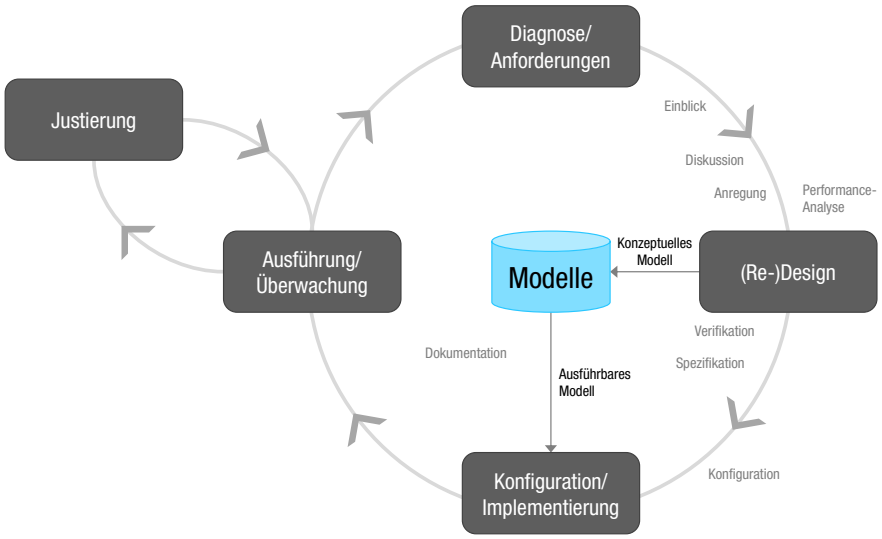



Abbildung 1.1: Prozesslebenszyklus mit Verwendungszwecken für Prozessmodelle

Unternehmen und seine Kunden Werte zu generieren. Das *Geschäftsprozessmanagement*[2] gilt als

> „systematischer Ansatz, um sowohl automatisierte als auch nicht-automatisierte Prozesse zu erfassen, zu gestalten, auszuführen, zu dokumentieren, zu messen, zu überwachen und zu steuern und damit nachhaltig die mit der Unternehmensstrategie abgestimmten Ziele zu erreichen." [21]

Die in dieser Definition enthaltenen Einzelschritte des Prozessmanagements variieren je nach Literaturquelle. In den folgenden Abschnitten werden das in der vorliegenden Arbeit verwendete Konzept des Prozessmanagements erläutert sowie eine grobe Kategorisierung verschiedener Prozessarten und -perspektiven wiedergegeben. Orthogonal dazu werden drei wesentliche Modellierungsparadigmen und ihre Bedeutung hinsichtlich der verschiedenen Prozessarten dargestellt.

### 1.1.1 *Phasen und Prozessmodelle*

*Prozesslebenszyklus*

Die obenstehende Definition des Prozessmanagements unterteilt selbiges in eine Folge von Einzelschritten, die in der Literatur meist in einem sogenannten Prozesslebenszyklus angeordnet werden. Wie für das Prozessmanagement selbst existieren mehrere unterschiedliche Definitionen für das zyklische Modell. Da die in der vorliegenden Arbeit diskutierten Problemstellungen einen starken Bezug zur Rolle der Prozessmodelle haben, bauen die nachfolgenden Kapitel auf der in [6] angegebenen Variante auf. Abbildung 1.1 zeigt eine an die nachfolgenden Erläuterungen angepasste Form des Prozesslebenszyklus.

*Diagnose-/ Anforderungsphase*

Der Lebenszyklus wird durch die *Diagnose- und Anforderungsphase* eingeleitet.

2 Im Folgenden werden Geschäftsprozess und Geschäftsprozessmanagement mit Prozess und Prozessmanagement abgekürzt.

In dieser wird der Prozess aufgenommen. Das geschieht meist über Interviews mit Prozessbeteiligten, das Sichten der Dokumentation, die Auswertung der Nutzungsaufzeichnungen involvierter IT-Systeme und andere Quellen. Das Ergebnis dieser Phase ist eine erste, informale Skizze eines Prozesses, die einen Einblick in und eine Diskussionsgrundlage für den Prozess liefert.

*Design-Phase*

In der *Design-Phase* werden auf Basis der vorherigen Anforderungsanalyse die Ziele des Prozessmanagements sowie die relevanten Prozesse des betreffenden Unternehmens identifiziert und modelliert. Das Ergebnis ist ein in einer *Prozessmodellierungssprache*[3] formuliertes, *konzeptuelles* Modell[4], welches den umzusetzenden realen Prozess häufig auf einer implementierungsfernen aber leichter verständlichen Ebene beschreibt. Eine Prozessmodellierungssprache – und jede andere Modellierungssprache – besteht unter anderem aus einer *Abstrakten Syntax*, also einer Menge von Sprachelementen sowie Regeln für deren mögliche Verknüpfungen [186]. Diese Darstellung dient zur Repräsentation der Semantik des Modells. Zusätzlich verfügt eine Prozessmodellierungssprache über eine oder mehrere *konkrete Syntaxen*, welche jeweils eine textuelle oder grafische Notation beschreiben. Damit ist die konkrete Syntax der Berührungspunkt für die mit der Modellierungsaufgabe betrauten Nutzer und die abstrakte Syntax ist der Berührungspunkt für IT-Systeme, die das Modell in irgendeiner Form verwenden sollen.

*Prozessmodellierungssprachen*

*Konkrete und abstrakte Syntax*

Die Hauptaufgabe der sich anschließenden *Konfigurations- und Implementierungsphase* besteht darin, das Prozessmodell mit Informationen anzureichern, die eine IT-gestützte Ausführung erlaubt. Hierzu können z. B. die Konfiguration des Nachrichtenversandes oder die Automatisierung einer Aufgabe über einen Webservice gehören. Das Ergebnis *dieser* Phase ist ein *ausführbares* Modell.

*Konfigurations-/ Implementierungsphase*

In der *Ausführungs- und Überwachungsphase* werden die Prozesse gemäß des konfigurierten Modells *ausgeführt*. Diese Ausführung übernehmen hierbei meist *Workflow-Management-Systeme (WfMS)*[5], die dazu befähigt sind, ausführbare Prozessmodelle zu interpretieren. Unter Ausführung wird daher sowohl die Abarbeitung automatisierter Aktivitäten durch IT-Systeme als auch die – ebenfalls IT-gestützte – Instruktion von Prozessbeteiligten verstanden. Um diese Aufgaben zu bewältigen, umfasst ein WfMS verschiedene Subkomponenten. Dazu zählen besonders die Routingkomponente zur Regelung des Informationsflusses im Prozess, die Koordinierungskomponente zur kollisionsfreien Ressourcenzuweisung und die Applikationskomponente zur automatisierten Bereitstellung einer Softwareapplikation und zur automatisierten Abarbeitung von Tätigkeiten ohne menschlichen Entscheidungsbedarf [76, 85]. Informationen über den Ausführungsverlauf und dessen Erfolg werden vom WfMS kontinuierlich erfasst und zur Identifikation von Anforderungen für Änderungen weitergereicht.

*Ausführungs-/ Überwachungsphase*

*Workflow-Management-Systeme*

3 Im Folgenden werden, so nicht weiter präzisiert, die Begriffe Geschäftsprozess-Modellierungssprache, Prozessmodellierungssprache und Modellierungssprache zur Vereinfachung synonym verwendet. Dies gilt analog auch für die Begriffe Geschäftsprozessmodell, Prozessmodell und Modell.

4 In der Literatur wird ab und an der Begriff *konzeptionelles* Modell synonym verwendet. *Konzeptuell* bedeutet jedoch *ein Konzept aufweisend*, während *konzeptionell* dagegen *ein Konzept betreffend* bedeutet [74]. Im Folgenden ist, so nicht näher bestimmt, mit Prozessmodellierungssprache oder Prozessmodell jeweils die konzeptuelle Form gemeint.

5 Die eigentliche Ausführung wird von der sogenannten Ausführungskomponente (engl. *workflow engines*) übernommen. Im Folgenden werden die Begriffe Workflow-Management-System, Ausführungssystem und -komponente vereinfacht synonym verwendet.

*Justierungsphase*

Änderungen können einerseits durch eine Anpassung der Prozesskonfiguration oder durch die Änderung des Prozessdesigns umgesetzt werden. Die *Justierungsphase* beschäftigt sich ausschließlich mit ersterem.

Die nächste Iteration des Regelkreises wird durch eine weitere Diagnose- und Anforderungsphase eingeleitet. Dabei wird die vorherige Prozessausführung ausgewertet und es werden gegebenenfalls neue Anforderungen identifiziert. Diese können einerseits aus dem Kontext des Prozesses, also Änderungen der prozessexternen Bedingungen, oder aus den Erkenntnissen der erfolgten Prozessausführung hervorgehen. In diesem Fall und im Falle dessen, dass die Prozessausführung Performance-Anforderungen, wie Bearbeitungs- und Durchsatzzeiten nicht oder ungenügend erfüllt, schließt sich eine weitere Iteration an – beginnend mit der *Re-Design-Phase*.

*Verwendungszwecke für Prozessmodelle*

Das Prozessmodell[6] nimmt im gesamten Lebenszyklus eine zentrale Rolle ein. Es gewährt dem Modellierer selbst multiperspektivisch (Abschnitt 1.1.2) Einblick in einen Prozess, bildet eine Diskussionsgrundlage für die Beteiligten sowie eine Basis und Anregung, verschiedene Prozess-Szenarien „durchzuspielen". Zudem bietet es die Möglichkeit, mittels Simulationstechniken Einflussfaktoren auf Performance-Indikatoren, wie z. B. Antwort- und Durchlaufzeiten zu identifizieren und im Rahmen der Verifikation Fehler, z. B. *Deadlocks*, aufzudecken. Gleichzeitig dienen Modelle zur Dokumentation des betrachteten Prozesses, was unter anderem für Zertifizierungs- und Schulungsmaßnahmen von zentraler Bedeutung sein kann. Modelle dienen außerdem als Spezifikation für das zu *implementierende* oder als Konfiguration für ein *existierendes* Ausführungssystem.

*Verständlichkeit von Prozessmodellen*

In allen genannten Verwendungsfällen bildet das Prozessmodell eine Schnittstelle zwischen der Wahrnehmung des Prozesses seitens der menschlichen Beteiligten und der Implementierung eines Informationssystems, welches den fraglichen Prozess unterstützt. Damit bedingt der Kompromiss zwischen Verständlichkeit für verschiedene menschliche Beteiligte und Verständlichkeit seitens des Workflow-Management-Systems sehr stark die Wahl geeigneter Prozessmodellierungssprachen. Die vorliegende Arbeit beschäftigt sich vordergründig mit Möglichkeiten zur Beseitigung der Notwendigkeit dieser Kompromissbildung.

Die Wahl geeigneter Prozessmodellierungssprachen kann einerseits vom verwendeten Workflow-Management-System aber besonders von den Charakteristika des umzusetzenden realen Prozesses beeinflusst werden. Die nachfolgenden Abschnitte betrachten zwei der wesentlichen Einflussdimensionen sowie eine zugehörige Klassifikation existierender Prozessmodellierungssprachen.

### 1.1.2 *Prozessperspektiven*

Prozesse beschreiben das Zusammenspiel von Aufgaben, Ressourcen, Hilfsmitteln und anderen Entitäten. Für die Prozessmodellierung wurde eine Systematik entwickelt, welche eine Prozessbeschreibung in die in Abbildung 1.2 dargestellten

6 In der Implementierungsphase kann das ursprüngliche Prozessmodell in eine andere Modellierungssprache übersetzt werden, welche den Ausführungsaspekt in den Vordergrund stellt. Folgerichtig müsste man hier potentiell von mehreren Modellen sprechen, worauf aber aus Gründen der Verständlichkeit verzichtet wird.

*Perspektiven* aufteilt [47, 86]. Jede davon weist aber immer Berührungspunkte mit anderen Perspektiven auf, was in Abbildung 1.2 exemplarisch dargestellt ist.


Abbildung 1.2: Fünf Prozessperspektiven

*Funktionale Perspektive*

Die *Funktionale Perspektive* betrachtet die Arbeitseinheiten, welche zum Erreichen der Ziele des Prozesses beitragen können. Einzelne Prozessschritte bilden die atomare Grundebene, eine durch gemeinsame Ressourcen, Prozessbeteiligte oder funktional zusammenhängende Menge an Prozessschritten kann als Subprozess modelliert werden. Die dadurch entstehenden Modellhierarchien strukturieren den Prozess zugunsten der Lesbarkeit und ermöglichen die Wiederverwendung dieser Teilmodelle.

*Verhaltensorientierte Perspektive*

Zwischen einzelnen Prozessschritten können zeitliche und/oder kausale Abhängigkeiten bestehen, die in der *Verhaltensorientierte Perspektive* fokussiert werden. Häufig wird die somit definierte Abfolge auch als Kontrollfluss bezeichnet.

*Datenorientierte Perspektive*

Der Informationsfluss in einem Prozess ist Betrachtungsgegenstand der *Datenorientierten Perspektive* und setzt sich aus dem Produzieren, Verändern und Konsumieren von Daten zusammen. Die in der Funktionalen Perspektive identifizierten Prozessschritte dienen entsprechend als Produzent, Modifizierer respektive Konsument. Folgerichtig können die dadurch hinzukommenden Abhängigkeiten die zeitliche Abfolge der Prozessschritte beeinflussen – oder deren inhaltliche Grundlage liefern.

*Operationale Perspektive*

Werkzeuge, egal ob physische oder softwarebasierte, und die konkrete Umsetzung der einzelnen Prozessschritte werden in der *Operationalen Perspektive* behandelt. Damit sind sowohl die prozessbezogene Interaktion zwischen Mensch und IT-Landschaft als auch die Automatisierung von Teilprozessen anhand dieser Perspektive nachvollziehbar.

*Organisatorische Perspektive*

In der *Organisatorischen Perspektive* wird festgelegt, wie zur Prozesslaufzeit die einzelnen Prozessschritte ihren menschlichen Prozessbeteiligten, Akteure genannt, zugewiesen werden. Zuweisungen erfolgen meist an konkrete Personen, die Zuweisungsregeln beschreiben jedoch eher die Voraussetzungen, welche diese Person erfüllen müssen. Diese können in der Praxis vergleichsweise komplex werden, weswegen in der Regel auf ein zusätzliches organisatorisches Modell zurückgegriffen wird. In diesem können sowohl hierarchische Beziehungen zwischen den Akteuren und Akteurgruppen sowie deren Kompetenzen modelliert sein.

*Selektivität der Perspektiven*

Die bisher aufgezählten Perspektiven sind weder für jeden Anwendungsfall vollständig, noch müssen selbst die genannten in jedem Fall betrachtet werden. Beispielsweise kann in einigen Fällen die bloße Abfolge der einzelnen Prozessschritte als Arbeitsplan dienen, um zur Laufzeit noch ausstehende Aufgaben zu identifizieren. Für eine Abschätzung des Personalbedarfs kann eine einfache Erweiterung um die Organisatorische Perspektive ausreichen.

*Erweiterbarkeit der Menge an Perspektiven*

Abhängig von Domäne und konkretem Prozess können weitere Perspektiven hilfreich oder sogar erforderlich sein. Ein Beispiel hierfür ist die *Ortsbezogene Perspektive* [163], welche beispielsweise die Möglichkeit schafft, für die Zuteilung von Prozessen und Prozessschritten zu Akteuren auf Standortinformationen zurückzugreifen. Dies kann beispielsweise für Prozesse in Speditionen oder internationalen Unternehmen von zentraler Bedeutung sein.

*Vollständigkeit der Erfassung*

Der tatsächliche Ablauf eines Prozesses ergibt sich aus der Betrachtung aller Perspektiven. Eine Aufnahme des rein zeitlichen Ablaufs von Prozessschritten, d.h. nur die verhaltensorientierte Perspektive zu erfassen, blendet deren inhaltliche Rechtfertigung aus und reduziert das Potential für Optimierungsmaßnahmen. Das kommt besonders dann zum Tragen, wenn eine zeitliche Abhängigkeit zwischen zwei Schritten modelliert wird, ohne dass im umzusetzenden realen Prozess eine entsprechende inhaltliche oder rechtliche Grundlage dafür besteht. In diesem Beispiel ist eine denkbare Optimierung die Parallelisierung der Abarbeitung dieser beiden Aufgaben. Ohne Modellierung der nicht-kontrollflussorientierten Perspektiven kann dieses Potential jedoch nur schwer erkannt werden. Weiterhin können auch perspektivenübergreifende Abhängigkeiten auftreten. Beispielsweise könnte die Einhaltung einer bestimmten Arbeitsreihenfolge nur für eine Gruppe von Akteuren verpflichtend sein. Durch das Ausblenden der involvierten Perspektiven ließe sich eine derartige Regelung nicht im Modell realisieren. Andererseits kann ein vorläufiges Ausblenden bestimmter Prozessperspektiven auch hilfreich sein, um beispielsweise ein Prozessmodell übersichtlicher zu präsentieren. Die Wahl eines geeigneten Detailgrades für die Modellierung ist also von Zweck und Zielgruppe abhängig.

### 1.1.3 *Prozesstypen: Routine- vs. Entscheidungsintensive Prozesse*

Während sich die im vorangegangenen Abschnitt beschriebenen Prozessperspektiven mit einer Systematik der Prozess*bestandteile* beschäftigen, fokussiert die nachfolgende Diskussion eine Systematik von Prozess*typen* und zu deren Beschreibung geeigneter Modellierungsparadigmen.

Prozessmodelle sollen Prozesse in einem gewünschten Detailgrad verständlich abbilden. Da Prozessmodellierungssprachen zur Klasse der *Domänenspezifischen Sprachen* (engl. *domain specific languages*, kurz: DSLs) gehören, weisen sie eine Spezialisierung [65] auf die Domäne der Prozesse auf. Dies unterstützt die Wahl des Detailgrades und die Verständlichkeit, verringert jedoch gleichzeitig die Flexibilität der Abbildung. Dies kann zu Schwierigkeiten führen, wie die nachfolgende Unterscheidung dreier Prozesstypen zeigt.

In [85] wird belegt, dass sich Prozesse anhand verschiedener Eigenschaften klassifizieren lassen. Jede Klasse, jeder Typ stellt dabei unterschiedliche Anforde-

rungen an die Modellierung [153]. Eines der wichtigsten Klassifikationskriterien ist die *Vielfalt* verschiedener möglicher Abläufe eines Prozesses. Dadurch ergeben sich folgende Prozesstypen:

*Prozesstypen*

- *Routineprozesse* zeichnen sich durch strikte, wiederkehrende Abläufe aus, die zum Modellierungszeitpunkt bekannt sind. Dieser Prozesstyp gilt daher als ablaufzentriert.
- *Entscheidungsintensive Prozesse* beinhalten dynamische Abläufe, die zum Modellierungszeitpunkt nicht alle bekannt sind. Sie beinhalten in der Regel eine größere Zahl wissens- und datenbasierter Entscheidungspunkte, wodurch der Prozessverlauf häufig erst zur Laufzeit bestimmt werden kann [55, 179, 180]. Dieser Typ von Prozessen gilt daher als datenzentriert.
- In der Praxis hat sich herausgestellt, dass sich der Großteil der Prozesse nicht eindeutig in eine der oben genannten Kategorien einteilen lässt. *Hybride Prozesse*, bestehend aus routinehaften und entscheidungsintensiven Teilprozessen, komplementieren daher diese Klassifikation [194].

*Routineprozesse*

*Routineprozesse* werden sehr häufig in gleicher Form durchgeführt und können daher auch leicht *standardisiert* werden. Die Anzahl an unterschiedlichen Ablaufmöglichkeiten ist folgerichtig vergleichsweise gering [175]. Das Prozessmodell nimmt hierbei einer starke, leitende Rolle ein, da vergleichsweise wenige Entscheidungen zu treffen sind. Das sorgt dafür, dass die Tätigkeiten und Pfade zwischen Beginn und Ende des Prozesses bereits zum Modellierungszeitpunkt vorhersehbar sind – was gleichzeitig die Grundlage für die zentrale Rolle des Prozessmodells darstellt [153]. Als Beispiele können hier Fertigungsprozesse an Fließbändern, Bewilligungsprozesse für Kredite in Banken aber auch tägliche Abläufe in Filialen großer Supermarktketten genannt werden. Praktische Studien haben gezeigt, dass sich jedoch nur etwa 20–40% aller Prozesse eindeutig dieser Klasse zuordnen lassen [175].

*Tendenziell wenige Entscheidungspunkte*

*Entscheidungsintensive Prozesse und Personenzentriertheit*

*Entscheidungsintensive Prozesse* zeichnen sich durch eine große Anzahl an Entscheidungen aus, die zur Ausführungszeit und auf Basis von menschlichem Kontext- und Hintergrundwissen und/oder der jeweils aktuellen, häufig sehr differenzierten Datengrundlage getroffen werden müssen. Diese *Personen-* und *Datenzentriertheit* [55, 179, 180] sorgt gleichzeitig für eine weniger restriktive und leitende Rolle des Prozessmodells. Ob und in welcher Reihenfolge bestimmte Prozessschritte durchgeführt werden, ist hier von den Entscheidungen zur Laufzeit und damit von *situativen Informationen* abhängig. Der dadurch vergrößerte Handlungsspielraum führt zu einer größeren Zahl an *Varianten*, nach denen der jeweilige Prozess ablaufen kann [5]. Der konkrete Ablauf ist von Fall zu Fall unterschiedlich. Damit einhergehend wird in der einschlägigen Literatur der ablaufzentrierte Prozessbegriff durch *Fall* (engl. *case*) und der Begriff Geschäftsprozessmanagement durch *Adaptives Fallmanagement* (engl. *adaptive case management*) ersetzt [5]. Für die Modellierung bedeutet diese Fokussierung auf situative Entscheidungen eine Intensivierung der Betrachtung der für die Entscheidungsfindung notwendigen Informationen (*Datenorientierte Perspektive*), der menschlichen Akteure (*Organisatorische Perspektive*) und der Abgrenzung des prozesszielkonformen Rahmens für individuelle Entscheidungen.

*Variantenreichtum*

*Fälle und Adaptives Fallmanagement*

*Hybride Prozesse*

*Routinehafte und entscheidungsintensive Teilprozesse*

Eine „Schwarz-Weiß-Kategorisierung" – routinehaft oder entscheidungsintensiv – deckt weiterhin nur eine Teilmenge der in der Praxis vorkommenden Prozesse ab [194]. Eine weitere Klasse, die *Hybriden Prozesse*, beinhalten sowohl routinehafte als auch entscheidungsintensive Teilprozesse. Besonders im Bereich medizinischer Behandlungsprozesse treffen hierbei gesetzliche Restriktionen auf die Notwendigkeit individueller Diagnosestellung und darauf aufbauender Behandlung. Menschliche Experten, in diesem Fall Ärzte, treffen auf Basis ihres Expertenwissens Entscheidungen für einen möglicherweise routinehaften Behandlungsverlauf (z. B. Impfungen) oder für ein eher dynamisches Vorgehen, je nach Entwicklung des Zustands des Patienten.

### 1.1.4 *Imperative vs. Deklarative vs. Hybride Modellierung*

In diesem Abschnitt werden drei Modellierungsparadigmen erläutert, welche aus den Anforderungen der verschiedenen Prozesstypen (Abschnitt 1.1.3) hervorgegangen sind.

Im Bereich der Programmiersprachen haben sich zwei Strömungen herauskristallisiert, die sich hinsichtlich der Formulierung eines Programmes in dem Punkt unterscheiden, ob sie entweder explizite Zustandsänderungen des Systems festlegen oder eher eine Spezifikation des Berechnungsziels erlauben. Bereits 1995 beschrieb Lloyd den Charakter *deklarativer* Programmiersprachen [111]:

*Deklarative vs. Imperative Programmiersprachen*

> "[...] it involves stating the logic of an algorithm, but not necessarily the control."

Bekannte Beispiele sind für Vertreter dieser Klasse von Programmiersprachen sind *Prolog*, *Haskell* oder auch *SQL*. Sprachen wie *Pascal*, *C* und *Java* (bis Version 1.7) sind dagegen der Klasse der *imperativen* Programmiersprachen zuzuordnen. Mit der Sprache *Scala* wurde auch ein Hybrid geschaffen, mit dem imperativ und deklarativ beschriebene Programmteile gemischt werden können.

*Deklarative vs. Imperative Prozessmodellierungssprachen*

Aufbauend auf der im vorherigen Abschnitt beschriebenen Klassifikation von Prozessen – zu großen Teilen auf Basis des Anteils an Dynamik im Prozessverlauf – hat es in neuerer Zeit eine ähnliche Entwicklung hinsichtlich der Prozessmodellierungssprachen gegeben. Durch unterschiedliche Anforderungen an die Modellierung [61, 62, 133, 146, 153] haben sich zuerst ebenfalls *imperative* und später *deklarative* Prozessmodellierungssprachen entwickelt. Wie in Abbildung 1.3 dargestellt ist, beschreiben erstere einen Prozessablauf explizit, meist mit Hilfe von kontrollflussbasierten, grafischen Darstellungsformen. Jeder erlaubte Ablauf des Prozesses wird explizit im Prozessmodell spezifiziert. Jeder nicht im Modell enthaltene Ablauf ist damit automatisch verboten. Das ist auch gleichzeitig der grundlegende Unterschied zu deklarativen Prozessmodellen, die auf der Grundannahme aufgebaut werden, dass jeder beliebige Ablauf gestattet ist, solange er nicht explizit verboten wird. In der einschlägigen, englischen Literatur ist in diesem Zusammenhang häufig von der sogenannten *open-world assumption* die Rede [133]. Verbote werden in Form von Anforderungs- oder Regelspezifikationen (schraffierter Bereich in Abbildung 1.3) modelliert [62]. Jede darin enthaltene Regel kann die Zahl möglicher Ablaufvarianten des Prozesses einschränken. Die dadurch sicht-

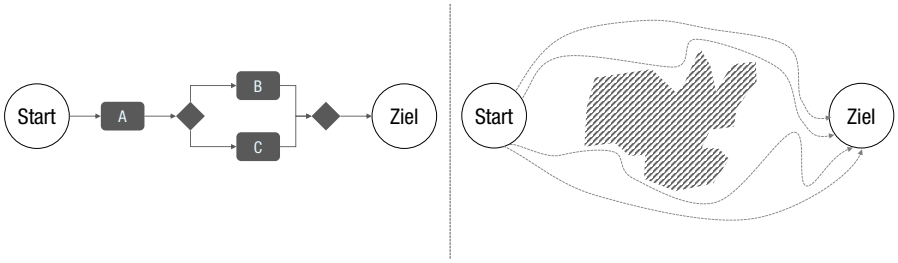



Abbildung 1.3: Imperative (links) vs. deklarative (rechts) Modellierung

bar werdende umgekehrte Proportionalität hinsichtlich der Größe imperativer und deklarativer Modelle zeigt den Bezug zu den zwei genannten Prozesstypen: Je routinehafter ein Prozess ablaufen muss, desto weniger Ablaufvarianten sind möglich und, folgerichtig, desto mehr Regeln müssen eingehalten werden. Umgekehrt gilt: Je stärker ein Prozessverlauf von Entscheidungen abhängig ist, desto mehr Ablaufvarianten sind möglich und, folgerichtig, desto „toleranter" ist das zugehörige regelbasierte Modell. Dieser Argumentation folgend werden Routineprozesse in der Regel imperativ, entscheidungsintensive Prozesse dagegen deklarativ modelliert [143]. In der Literatur wird im Zusammenhang mit deklarativer Prozessmodellierung fälschlicherweise auch von *wissensintensiven* Prozessen gesprochen [75]. Die für die Bewältigung eines Prozesses notwendige Menge an Wissen hat jedoch keinen Einfluss auf die Entscheidung, ob imperativ oder deklarativ modelliert werden sollte. Grund ist, dass sich die Vielfalt möglicher Abläufe nicht aus dem Wissen selbst, sondern seiner *Verwendung* und somit aus der Vielfalt der zu treffenden Entscheidungen ergibt.

Wie im vorangegangenen Abschnitt angedeutet ist die Mehrzahl realer Prozesse nicht eindeutig als ausschließlich routinehaft oder entscheidungsintensiv zu klassifizieren. Dagegen ist es jedoch in den meisten Fällen möglich, routinehafte und entscheidungsintensiv *Anteile* voneinander abzugrenzen [194]. Aus dieser Motivation heraus entwickelten sich *hybride* Prozessmodellierungsansätze [49, 194]. Diese Ansätze fokussieren im Wesentlichen die Identifikation oder Schaffung von Schnittstellen zwischen imperativen und deklarativen Prozessmodellierungssprachen sowie die Gewährleistung ihrer Interoperabilität. Damit bilden die hybriden Modellierungsansätze keine eigene Klasse an Modellierungs*sprachen*, können aber einem eigenen Paradigma zugeordnet werden.

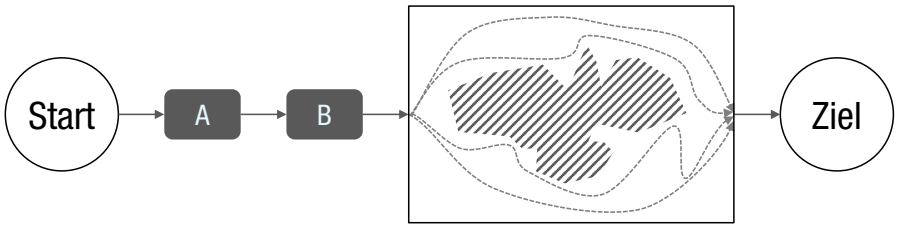



Abbildung 1.4: Hybride Modellierung

*Hybride Prozessmodellierungsansätze*

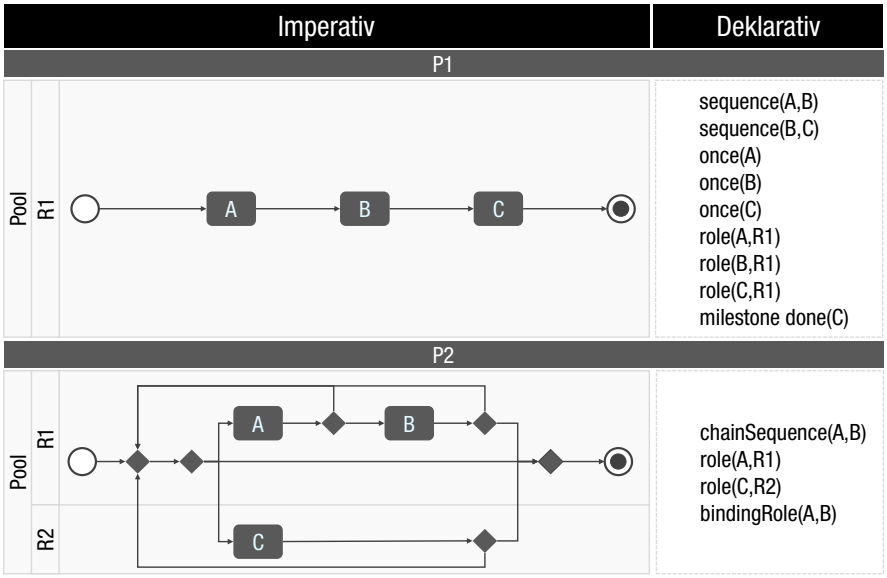



Abbildung 1.5: Imperative (links) vs. deklarative Modellierung (rechts); strikter (oben, P1) und entscheidungsintensiver (unten, P2) Prozess

### 1.1.5 *Imperatives und deklaratives Beispielprozessmodell*

Nach der bisher lediglich auf Typ-Ebene vorgenommenen Einführung imperativer und deklarativer Modellierung werden in diesem Abschnitt zwei künstliche, abstrakte Prozesse mit sehr begrenztem Umfang vorgestellt. Diese werden für den verbleibenden Verlauf des Kapitels zur Demonstration einzelner Teilprobleme verwendet.

*Beispielprozesse*

Die in Abbildung 1.5 modellierten Prozesse P1 und P2 beinhalten jeweils drei Aktivitäten (A, B und C). Alle in der Abbildung dargestellten Modelle umfassen die Funktionale, die Verhaltensorientierte und die Organisatorische Perspektive. Die aus der Aufnahme der Prozesse identifizierten Bedingungen für eine valide Prozessausführung lassen sich wie folgt beschreiben:

P1: Die Aufgaben müssen in der Reihenfolge ABC ausgeführt werden und sind der Rolle R1 zugeordnet. Eine Wiederholung der Aufgaben ist nicht möglich.

P2: Die Aufgaben können in jeder beliebigen Reihenfolge ausgeführt werden, solange unmittelbar vor einer Ausführung von B Aufgabe A abgeschlossen wird. Die Aufgaben A und B sind Rolle R1 und C ist Rolle R2 zugeordnet. Jede der Aufgaben kann beliebig oft wiederholt werden.

*Imperativ: Grafische, flussorientierte Darstellung*

Beide Modellpaare sind an dieser Stelle sprachlich rein qualitativ beschrieben. Die auf der linken Seite dargestellten Modelle basieren auf einer flussorientierten Darstellung, welche Aktivitäten als abgerundete Rechtecke in einem gerichteten Graphen anordnet. Der Prozessstart wird als Kreis und das Prozessende als Kreis mit ausgefüllter Mitte dargestellt. Alternativen im Ablauf werden durch ausgefüllte Rauten mit einer eingehenden und mehreren ausgehenden Kanten eingeleitet, welche es erlauben, einer oder mehrerer der ausgehenden Pfade zu folgen, um die nächste Aktivität zur Bearbeitung auszuwählen. Jede dieser Alternativen muss

schließlich über eine weitere Raute mit den übrigen Zweigen zusammengeführt werden.

*Deklarativ: Textuelle, regelbasierte Darstellung*

Rechtsseitig befindet sich jeweils die deklarative Repräsentation des Prozesses, basierend auf einer textuellen Regelsprache. Die Regelsprache verwendete vorgefertigte Regelschablonen, deren Semantik wie folgt lautet:

- *sequence(x,y):* Vor Ausführung von Aufgabe y muss mindestens einmal Aufgabe x ausgeführt worden sein.
- *once(x):* Aufgabe x darf nicht mehr als einmal ausgeführt werden.
- *chainSequence(x,y):* Unmittelbar vor der Ausführung von y muss immer Aufgabe x ausgeführt werden.
- *role(x,R):* Der Ausführende von Aufgabe x muss die Rolle R bekleiden.
- *binding(x,y):* Die Ausführenden der Aufgaben x und y müssen identisch sein.
- *milestone done(x):* Ein notwendiges Prozessziel ist erreicht, wenn Aufgabe x ausgeführt worden ist.

Vergleicht man nun die beiden Modellpaare *P1* und *P2*, dann wird ersichtlich, dass die imperative Darstellung von P1 kompakter ist, als ihr deklaratives Gegenstück. In P2 verhält es sich hingegen umgekehrt. Gemäß der in Abschnitt 1.1.3 unterschiedenen Prozesstypen ließe sich P1 als Routineprozess, P2 hingegen als entscheidungsintensiver Prozess kategorisieren. Eine Auffälligkeit ist, dass im deklarativen Modell zu Prozess P2 kein Meilenstein modelliert ist. Im imperativen Gegenstück ist es möglich, den Prozess direkt nach dem Beginn zu beenden, ohne eine Aktivität ausgeführt zu haben. Dies kann ein Hinweis auf eine mangelhafte oder fehlende Definition der Prozessziele sein, was hier tendenziell im deklarativen Modell klarer zu erkennen ist.

Mehr Flexibilität hinsichtlich des Ablaufs bedeutet umgekehrt weniger Einschränkungen. Damit ist der entscheidende Faktor, unabhängig von den konkreten Modellierungssprachen, der unterschiedliche Ausgangspunkt. Wie in Abschnitt 1.1.4 beschrieben, gestatten imperative Modelle nur Prozessverläufe, die explizit im Modell spezifiziert ist, während deklarative Modelle jede Reihenfolge erlauben, die nicht durch die im Modell enthaltenen Regeln implizit verboten ist.

Für den verbliebenen Teil dieses Kapitels wird von der oben vorgenommenen, informalen Definition der beiden Prozessmodellierungssprachen ausgegangen. Beide sind zwar stark an existierende Sprachen angelehnt und dienen hier aber lediglich der Veranschaulichung der grundsätzlich verschiedenen Ausgangspunkte imperativer und deklarativer Modellierung.

Vorrangiges Ziel von Abschnitt 1.1 ist es, einen Überblick über Zweck und Aufgaben des Geschäftsprozessmanagements zu verschaffen. Dabei steht besonders die Modellierung der Prozesse im Vordergrund, welche stark vom jeweiligen Prozesstyp und den durch ein Modell zu transportierenden Informationen abhängig ist. Da die einzelnen Phasen des Geschäftsprozessmanagements aufeinander aufbauen, wird beispielsweise auch die Prozessausführung von Entscheidungen zum Modellierungszeitpunkt beeinflusst. Die damit einhergehenden Schwierigkeiten (Abschnitt 1.2) sind die grundlegende Motivation der vorliegenden Arbeit.

## 1.2 SPRACHBEZOGENE AKZEPTANZPROBLEME DER PROZESSMODELLIERUNG

Auf Basis der verschiedenen, voneinander abzugrenzenden Prozesstypen und Modellierungsparadigmen (Abschnitt 1.1.3 und Abschnitt 1.1.4) werden in diesem Abschnitt Probleme identifiziert, welche die Verwendbarkeit von Prozessmodellierungssprachen in bestimmten Situationen erschweren. Grundlegendes Ziel der vorliegenden Arbeit ist es, Hilfsmittel zu entwickeln, welche die Verwendbarkeit und die damit verbundene Akzeptanz dieser Sprachen verbessern können.

Im Verlauf dieses Abschnitts werden zunächst vier Problembereiche identifiziert und auch auf daraus resultierende, für die Arbeit zentrale Anforderungen eingegangen. Aufgrund des großen Umfangs der Literaturgrundlage, fungiert dieser Teil als Zusammenfassung. Eine ausführliche Betrachtung schließt sich in den Teilabschnitten 1.2.1 bis 1.2.4 an.

*Babylonisches Gewirr an Prozessmodellierungssprachen*

Betrachtet man die in Abbildung 1.6 dargestellte Vielfalt an unterschiedlichen Prozessmodellierungssprachen, dann werden Beteiligte des Prozesslebenszyklus im Laufe der Zeit potentiell mit mehreren Prozessmodellierungssprachen konfrontiert. Dabei werden Personen und IT-Systeme gleichermaßen als *Beteiligte* angesehen. Mit der Unterscheidung zwischen imperativen und deklarativen respektive

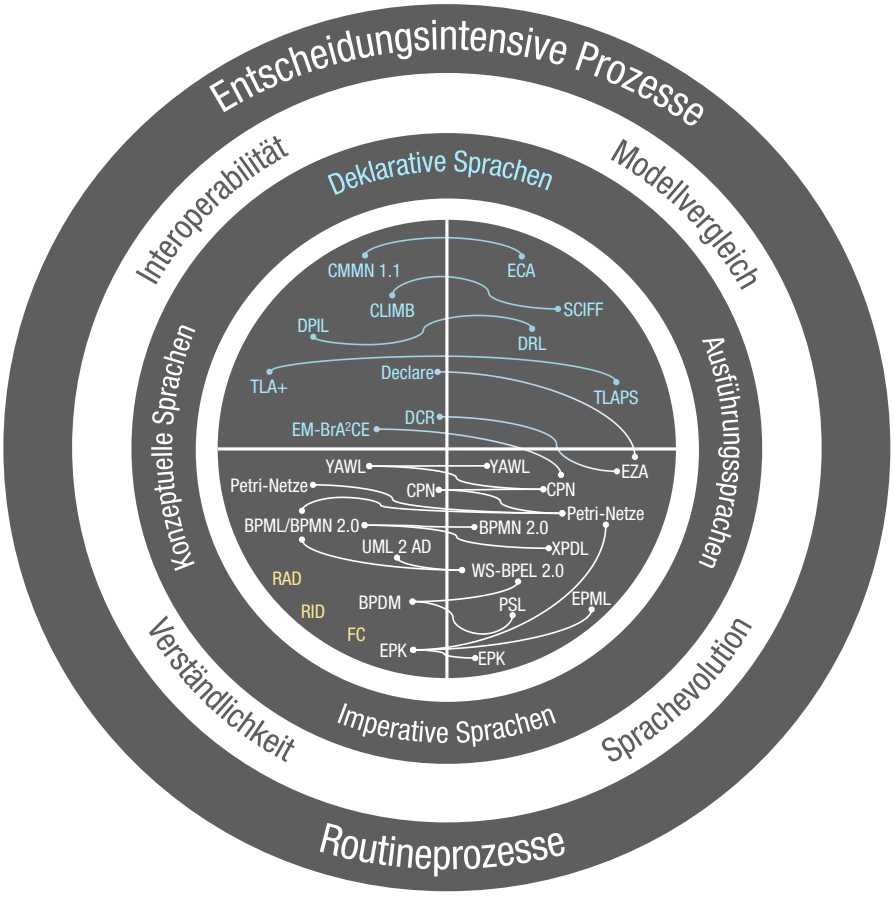



Abbildung 1.6: Beispiele für sprachliche Wechselwirkungen

konzeptuellen (Abschnitt 1.1.4) und Ausführungssprachen (Abbildung 1.1) für die Modellierung von Geschäftsprozessen ergeben sich somit *vier* Sprachklassen.

*Declare* [145] und *DPIL* [200] als Vertreter der deklarativen Sprachen sind konzeptuell und werden zur Ausführung auf jeweils eine Ausführungssprache abgebildet. Sprachen, wie beispielsweise die *Event-Condition-Action-Sprache (ECA)* [57] und das mathematische Modell der *Deterministischen Endlichen Zustandsautomaten (EZA)* [78] werden hingegen rein als Ausführungssprache eingesetzt. Die Sprachen *Role Activity Diagram (RAD)* [77] und *Flow Charts (FC)* [102] sind dagegen rein konzeptuell und nicht ausführbar.

*Konzeptuelle, deklarative Sprachen*

*Imperative Ausführungssprachen*

*Konzeptuelle Sprachen ohne Ausführung*

Wie aus Abbildung 1.6 ersichtlich wird, können konzeptuelle Prozessmodellierungssprachen gleichzeitig auch eine Ausführungssemantik aufweisen. Beispiele hierfür ist die von der *Object Management Group (OMG)* beschriebene *Business Process Model and Notation 2.0 (BPMN)* [141], ein Standard zur konzeptuellen, grafischen Modellierung von Geschäftsprozessen. Da die ersten Versionen der BPMN selbst Lücken hinsichtlich der Ausführungssemantik und des Speicherformats aufwies, wurden Ausführungssprachen, wie beispielsweise die *WS-Business Process Execution Language (BPEL)* [19] entwickelt, die durch ihre mathematische Fundierung diese Lücken schlossen. Mit der Einführung der Version 2.0 des Standards wurde jedoch auch ein Großteil der Ausführungssemantik ergänzt.

*Konzeptuelle, imperative Sprache mit Ausführung*

Grundsätzlich ist es jederzeit möglich, neue konzeptuelle Prozessmodellierungssprachen zu entwickeln. Auch der Abbildung einer existierenden konzeptuellen Sprache auf eine andere, bereits existierende oder sogar neue Ausführungssprache steht nichts im Wege. Aus diesem Grund kann Abbildung 1.6 nicht vollständig sein. Da jede neue Prozessmodellierungssprache und jede neue Abbildung auf eine alternative Ausführungssprache die Komplexität des Problems lediglich verstärkt, bleibt die Grundaussage stets gleich: Das Spektrum an Prozessmodellierungssprachen ist groß, sodass es nur schwer möglich ist, jede der Sprachen im Detail zu verstehen, aus Sicht der Ausführungssysteme zu unterstützen und robuste Verfahren zum Modellvergleich und zur Migration zu entwickeln.

In den nachfolgenden Abschnitten dieses Kapitels wird diskutiert, inwiefern die Vielfalt der Sprachen an sich und die Unterschiede der vier Sprachklassen die Verständlichkeit eines Prozessmodells, die Interoperabilität zwischen Design und Ausführung, die Migration eines Modells zu einer neuen Sprachversion und die Vergleichbarkeit zwischen Prozessmodellen negativ beeinflussen. Obwohl die Existenz weiterer Problemfelder wahrscheinlich ist, konzentriert sich die vorliegende Arbeit auf die genannten vier. Hierfür existieren im Wesentlichen drei Gründe. Sie sind einerseits bis heute Gegenstand zahlreicher Forschungsarbeiten (Abschnitte 1.2.1 bis 1.2.4) und ihre Relevanz ist andererseits auch in aktuellen Forschungsprojekten im akademischen Umfeld und mit Partnern aus Industrie und Wirtschaft zu erkennen. Der dritte – eher pragmatische – Grund ist, dass die drei in dieser Arbeit vorgestellten Werkzeuge zum Teil Synergien erzielen (Abschnitt 2.1), die sich besonders positiv auf die Behandlung eben dieser vier genannten Problembereiche auswirken. Dieses letzte Kriterium geht demnach von der Fragestellung aus, in welchen Problemfeldern die entwickelten Ansätze *in Kombination* eingesetzt werden können.

*Auswahl der Problemfelder*

*Beobachtungen*

Der empirische Teil der Begründung für die Auswahl der vier Problembereiche stützt sich konkret auf die folgenden, in verschiedenen Projekten[7] beobachteten Herausforderungen:

- Nutzer[8] können aus verschiedenen Gründen mit einer unbekannten oder ungewohnten Prozessmodellierungssprache in Kontakt kommen. [*Verständlichkeit*]
- Die Auswahl und Verwendung einer Prozessmodellierungssprache kann sich durch geänderte oder neue Anforderungen ändern. Dadurch können Nutzer mit der erstgenannten Problematik konfrontiert werden und bisher verwendete Werkzeuge und Systeme können meist nicht mehr verwendet werden. [*Verständlichkeit, Interoperabilität*]
- Informationen über die sequentielle Reihenfolge von Aktivitäten können tendenziell besser mit imperativen Prozessmodellierungssprachen umgesetzt und verändert werden. Umgekehrt eigenen sich für die Umsetzung und Veränderung situationsbezogener Informationen eher deklarative Sprachen. Für einen dynamischen Wechsel zwischen einer imperativen und einer deklarativen Sicht auf einen Prozess fehlen bisher die nötigen Werkzeuge. Hybride Sprachen könnten das Problem teilweise lösen, bergen aber das Hindernis, dass beide Modellierungsparadigmen und auch die verschmolzenen Sprachen bekannt sein müssen. Dies ist bisher in der Praxis nicht der Fall. [*Verständlichkeit, Interoperabilität*]
- Die Verständlichkeit von Prozessmodellen kann mittels natürlichsprachlicher Beschreibungen des Prozesses verbessert werden. [*Verständlichkeit*]
- Beispiele für – bezüglich eines Prozessmodells – valide Prozessausführungen und entsprechende Gegenbeispiele verbessern das Modellverständnis. [*Verständlichkeit*]
- Der Wechsel des Prozessausführungssystems aufgrund der geänderten Prozessmodellierungssprache ist unerwünscht. [*Interoperabilität*]
- Die Weiterentwicklung von Prozessmodellierungssprachen kann das Zusammenspiel der Design- und der Implementierungsphase des Prozesslebenszyklus blockieren. [*Sprachevolution*]
- Beim Vergleich zweier Prozessmodelle liegen diese nicht immer in derselben Modellierungssprache vor, was einen direkten Vergleich erschwert. [*Modellvergleich*]

*Anforderungen*

Diese Beobachtungen sind Indizien, die in den Abschnitten 1.2.1 bis 1.2.4 durch Literaturquellen gestützt werden. Sich daraus ergebende Anforderungen werden im Diskurs derselben Abschnitte motiviert und in folgender Listen zusammengefasst:

7 Als zentral ist hier vor allem das Kompetenzzentrum für praktisches Prozess- und Qualitätsmanagement (KpPQ) herauszustellen (`http://www.kppq.de/`, abgerufen am: 03.10.2017)

8 Als *Nutzer* gilt sowohl der Modellierungsexperte als auch jede andere Person, die in den Modellierungsprozess involviert ist.

| | A1 | A2 | A3 | A4 |
|---|---|---|---|---|
| Verständlichkeit | ✓ | ✓ | ✓ | |
| Interoperabilität | ✓ | | | |
| Sprachevolution | ✓ | | | ✓ |
| Modellvergleich | ✓ | ✓ | ✓ | |

Tabelle 1.1: Behandlung der Problembereiche durch identifizierte Anforderungen

A1 Es soll ermöglicht werden, ein Modell flexibel in andere Sprachen zu übersetzen. Besonderer Fokus liegt dabei auf der paradigmenübergreifenden Übersetzung.

A2 Es soll ermöglicht werden, für gegebene Prozessmodelle automatisiert einen – auf Wunsch vollständigen – Satz an Beispielen für *gültige* und für *nicht-gültige* Prozessausführungsspuren abzuleiten.

A3 Dem Nutzer sollte es ermöglicht werden, für gegebene Prozessmodelle automatisiert natürlichsprachliche Beschreibungen abzuleiten.

A4 Der Aufwand, syntaktische, evolutionäre Änderungen an Prozessmodellierungssprachen auf zugehörige, existierende Prozessmodelle anzuwenden, soll durch einen erhöhten Grad an Automatisierung hinsichtlich der Migrationsmechanismen reduziert werden.

*Ziele*

Tabelle 1.1 zeigt, welche der Anforderungen auf die Behandlung welcher der vier Problembereiche abzielen. Durch die Erfüllung der Anforderungen A1 bis A3 werden dem Nutzer zusätzliche, verschiedenartige Sichten auf einen modellierten Prozess zur Verfügung gestellt. Anforderung A1 berücksichtigt zusätzlich auch die Systemsicht. Durch die Möglichkeit einer flexiblen Übersetzung zwischen Prozessmodellierungssprachen kann beispielsweise der Wechsel eines Ausführungssystems unnötig werden. Zusammen mit Anforderung A4 ist sie außerdem die Grundlage für eine flexible, syntaktische Migration bestehender Modelle im Zuge der Evolution von Prozessmodellierungssprachen. Migration kann auch als eine speziellere Form der Translation betrachtet werden, weswegen dieser Problembereich mit beiden Anforderungen assoziiert ist. Ein Vergleich von Modellen, die in unterschiedlichen Prozessmodellierungssprachen formuliert sind, kann auf unterschiedliche Arten durchgeführt werden. Dazu zählen eine Übersetzung in die jeweils andere Prozessmodellierungssprache (A1), ein Vergleich des Ausführungsverhaltens (A2) aber auch der Abgleich natürlichsprachlicher Beschreibungen der modellierten Prozesse (A3).

Die Liste der identifizierten Anforderungen wird im Hauptteil der Arbeit mit bestehenden Lösungen abgeglichen. Lücken werden über entsprechende eigene Entwicklungen geschlossen. Im Evaluationsteil wird diskutiert, inwiefern die identifizierten Anforderungen damit erfüllt werden. Die Erfüllung der Anforderungen gewährt eine höhere sprachliche Flexibilität für die Modellierung von Prozessen, was das grundlegende Ziel der Arbeit darstellt. Zu den Teilzielen gehören unter anderem eine verbesserte Verständlichkeit von Prozessmodellen, eine breitere

Unterstützung der Sprachen durch Ausführungssysteme, eine gemeinsame Basis unterschiedlicher Sprachen für den Vergleich von Modellen sowie eine größere Robustheit gegenüber der Weiterentwicklung der Prozessmodellierungssprachen. Diese vier Teilziele und ihre jeweiligen Problembereiche werden in den Abschnitten 1.2.1 bis 1.2.4 detailliert betrachtet.

### 1.2.1 *Verständlichkeit*

Aus dem Prozesslebenszyklus-Modell (Abbildung 1.1) lassen sich mindestens zwei Phasen identifizieren, in denen ein Prozessmodell von Menschen interpretiert werden muss – die (Re-)Design- und die Konfigurations-/Implementierungsphase. In beiden Fällen sind das Resultat ein oder mehrere Prozessmodelle, deren Verwendung, den Phasen entsprechend, unterschiedlich ist. Bis zum aktuellen Zeitpunkt gibt es jedoch keine allgemein akzeptierte Standardmethode, die als Anleitung für die Bewältigung dieser Modellierungsaufgabe im Geschäftsprozessmanagement verwendet werden kann [148]. Dagegen existiert eine klare Vorstellung, aus welchen Bestandteilen sich eine solche Methode zusammensetzt [84]. Die zwei wesentlichen Bestandteile sind:

*Methoden zur Prozessmodellierung*

- *Sprache:* Die *Sprache* beschreibt einerseits die äußere Form in der die Ergebnisse einer Modellierung visualisiert werden. Andererseits ist für diese Darstellung auch eine Semantik festzulegen.
- *Vorgehensweise:* Die *Vorgehensweise* beschreibt eine Abfolge von Schritten und Aufgaben zur Erzeugung eines Prozessmodells in dieser Sprache.

Damit ergibt sich für die Nutzung einer *Methode zur Prozessmodellierung* die triviale Voraussetzung, dass dem Anwender die jeweilige Prozessmodellierungssprache bekannt ist. Die Frage der *Verständlichkeit* von Prozessmodellen kann jedoch nicht *allein* damit beantwortet werden, ob der Anwender mit der Repräsentationssprache vertraut ist oder nicht. Beispielsweise wird im vorangegangenen Abschnitt bereits angedeutet, dass Routineprozesse verständlicher mittels imperativer und entscheidungsintensive Prozesse zugänglicher mittels deklarativer Modellierungssprachen dargestellt werden können. Verständlichkeit ist daher nicht allein ein Problem der persönlichen Sprachpräferenz, sondern auch ein Problem der Eigenschaften des Modellierungsgegenstandes. Diese Behauptung wird von mehreren Untersuchungen, zum Teil zumindest tendenziell oder implizit, gestützt [26, 61, 62, 73, 126, 143, 145, 146]. Folglich können die Einflüsse auf die Verständlichkeit von Prozessmodellen grundsätzlich in zwei Klassen eingeteilt werden:

*Endogen-subjektive vs. exogen-objektive Einflüsse*

1. *Endogen-subjektiv:* personenbezogene, individuelle Einflüsse und
2. *Exogen-objektiv:* auf den Modellierungsgegenstand bezogene, allgemeine Einflüsse.

*Sequenzbezogene vs. situationsbezogene Informationen*

Die Arbeiten von Pesic et al. [143, 145], Fahland et al. [61, 62] und Pichler et al. [146] diskutieren zwei grundlegende Hypothesen, welche besagen, dass sich sequenzbezogene Informationen über die zeitliche Reihenfolge von Aktivitäten besser

mittels imperativer und situationsbezogene Informationen besser mittels deklarativer Prozessmodellierungssprachen umsetzen und verändern lassen. Als sequenzbezogene Informationen werden hier entsprechend der Unterscheidung imperativer und deklarativer Modelle explizite Vorgaben für die Reihenfolge von Prozessschritten bezeichnet. Konsequenterweise sind situationsbezogene Informationen dadurch charakterisiert, dass sie Regeln beschreiben, die nur dann relevant werden, wenn sich der Prozess in einem bestimmten Zustand, in einer bestimmten Situation befindet. Diese Hypothesen werden auch in einer späteren Studie bestätigt [73]. Diese diskutiert vordergründig den Prozess des Verstehens von deklarativen Prozessmodellen und damit verbundene Hürden am Beispiel von Declare, einer deklarativen Prozessmodellierungssprache. Eine wesentliche Erkenntnis der Untersuchung ist, dass Declare-Modelle mit steigender Größe unverständlicher werden, da sich durch den deklarativen Charakter häufig *versteckte Abhängigkeiten*, also Regeln, die aus der Kombination von zwei oder mehr Regeln, implizit ergeben. Diese Einflüsse sind als exogen-objektiv zu kategorisieren.

*Versteckte Abhängigkeiten*

Mendling et al. [126] schlagen einen Satz von sieben Richtlinien für qualitativ hochwertige, grafische Prozessmodelle vor. Man solle beispielsweise so wenige Prozesselemente wie möglich im Modell verwenden, einen niedrigen Grad der Vernetzung der Entitäten durch Kanten sicherstellen und Modelle, die mehr als 50 Prozesselemente beinhalten, durch Dekomposition aufteilen. Auch wenn diese Richtlinien vor Ihrer Verwendung kritisch hinterfragt werden müssen und meist zu generalisierend sind, sind auch hier wieder die Größe und Komplexität der Modelle und damit die Entscheidung imperativer oder deklarativer Modellierung von zentraler Bedeutung. Auch diese Richtlinien sind aus Erkenntnissen exogen-objektiver Einflüsse entstanden.

*Möglichst geringe Modellgröße*

In [148] argumentieren die Autoren auf empirischer Basis, dass der Mangel an klar definierten Modellierungszielen in der Praxis oft ein Problem darstellt. Die Modellierung solle zielgerichtet erfolgen. Zusammen mit der von Mendling et al. vorgeschlagenen möglichst geringen Modellgröße, lässt sich damit schlussfolgern, dass ein Modell nur das, was zum Erreichen des Modellierungsziel relevant ist, enthalten soll[9]. Gleichzeitig darf es jedoch auch nicht weniger enthalten [26]. Ein Modell kann beispielsweise alle der in Abbildung 1.2 dargestellten Perspektiven abdecken. Ist jedoch für das Modellierungsziel lediglich die zeitliche Reihenfolge der Prozessschritte relevant, dann kann alternativ auch eine kontrollflussorientierte Darstellung gewählt werden. Eine Möglichkeit, diesen Fokuswechsel zu realisieren, ist die Übersetzung des bestehenden Modells in eine andere Prozessmodellierungssprache, die sich auf die gewünschten Perspektiven fokussiert (Anforderung A1). Die Reduktion oder die Erweiterung auf den gewünschten Fokus ist aufgrund des starken Bezugs zum Modellierungsgegenstand ebenfalls den exogen-objektiven Einflüssen zuzuordnen.

*Modellierungsziele*

Georges et al. [136] und Recker et al. [152] abstrahieren vom konkreten Modellierungsparadigma und betrachten den Einfluss bestimmter kognitiver Fähigkeiten auf das Verständnis von Prozessmodellen. Eine der grundlegenden Aussagen ist hierbei, dass sich verschiedene kognitive Fähigkeiten – beispielsweise das Selektions- und das Abstraktionsvermögen – positiv oder negativ auf das Verständnis von Pro-

*Kognition und Modellierungverständnis*

9 Siehe auch *Reduktionsmerkmal* nach Stachowiak [170]

zessmodellen auswirken. Darauf aufbauend wird die Schlussfolgerung gezogen, dass die *individuellen* kognitiven Charakteristika für das Verständnis von Prozessmodellen von zentraler Bedeutung sind und daher die Verständlichkeitsproblematik nicht losgelöst von den konkreten involvierten Nutzern einer Prozessmodellierungssprache betrachtet werden kann. Das verstärkt die identifizierten Anforderungen und macht gleichzeitig eine allgemeingültige Aussage über die Bevorzugung eines oder mehrerer der im Hauptteil der Arbeit vorgestellten Ansätze unmöglich. Diese Problematik ist als endogen-subjektiver Einfluss zu kategorisieren.

*Initiale Auswahl der Modellierungssprache*

Damit konkurrieren endogen-subjektive und exogen-objektive Einflüsse auf die Verständlichkeit von Prozessmodellen. Folglich darf die initiale Auswahl der im konkreten Fall *zu verwendenden* Modellierungssprache nicht allein auf Basis der persönlichen Präferenzen der Anwender getroffen werden.

*Neuauswahl der Modellierungssprache*

Der in Abbildung 1.1 dargestellte Prozesslebenszyklus ist iterativ und erlaubt die Berücksichtigung neuer oder geänderter Anforderungen, die sich aus der Analyse früherer Ausführungen des Prozesses oder durch veränderte Rahmenbedingungen ergeben haben. Potentiell können sich diese Veränderungen auch auf die Wahl der geeigneten Prozessmodellierungssprache auswirken [100]. Ein Prozess, der anfänglich sehr stark strukturiert – beinahe routinehaft – anmutet, könnte in einer weiteren Re-Design-Iteration als tendenziell entscheidungsintensiv bewertet werden, wodurch sich dann eine deklarative Repräsentation empfiehlt. Auch der umgekehrte Fall ist denkbar. Ein Prozess, der initial sehr entscheidungsintensiv erschien, kann sukzessive weiter eingeschränkt werden, sodass letztlich eine eher routinehafte Abfolge von Prozessschritten beschrieben wird. Damit kann der Modellierungsgegenstand einen Sprachwechsel veranlassen. Folglich ist die Möglichkeit für den flexiblen Wechsel zwischen verschiedenen Prozessmodellierungssprachen der beiden Paradigmen erneut von großer Wichtigkeit (Anforderung A1). Hinsichtlich eines Wechsels vom imperativen zum deklarativen Paradigma hat sich gezeigt, dass die Kenntnisse über die Prozessmodellierung in imperativen Sprachen nicht auf die deklarative Modellierung übertragen werden können und stattdessen sogar hinderlich sein können [73]. Eine Untersuchung, ob eine simultane Präsentation eines imperativen und eines deklarativen Modells, die beide denselben Prozess beschreiben, Vorteile beim Umstieg auf das deklarative Paradigma bieten, fehlt. Um eine solche Untersuchung zu unterstützen, könnte es hilfreich sein, beide Modelle in die jeweils andere Sprache zu übersetzen, was wiederum Anforderung A1 untermauert.

*Prozessmodelle als Kommunikationsmittel*

Im Verlauf dieses Abschnitts wird lediglich die Perspektive des im Prozesslebenszyklus beteiligten Modellierungsexperten betrachtet. Becker et al. [26] argumentieren dagegen, dass Vertreter verschiedener Geschäfts- und Technikabteilungen immer stärker in die Modellierungsaufgabe eingebunden werden – wobei diese Vertreter keine Modellierungs-, sondern Domänenexperten sind. Prozessmodelle dienen demnach bereits zum Modellierungszeitpunkt auch als Kommunikationsmittel [112] zwischen den beiden Expertengruppen. Ein häufig auftretendes Problem hierbei ist, dass dem Domänenexperten die Modellierungssprache unbekannt ist, was Diskussionen über die Modellentwürfe erschwert [26, 83, 152] (endogen-subjektiver Einfluss). Diese Problematik ist auch aus dem Bereich relationaler Datenbanken in Form der *Entity-Relationship-Modellierung* bekannt [66].

Vielfach wird daher auch gefordert, Prozessmodelle mittels einer natürlichsprachlichen Beschreibung zu komplementieren [108] (Anforderung A3).

Das Erlernen allgemeiner Prinzipien auf Basis von Beispielen ist ein beliebtes didaktisches Mittel. Grund hierfür ist die stark kategorisierende menschliche Denkweise [103]. Auf Basis von Gemeinsamkeiten werden einzelne Objekte in eine Kategorie eingeordnet. Auf Basis von Unterschieden werden andere Objekte von dieser Kategorie getrennt. Aus diesem Grund ist eine weitere Möglichkeit, die betroffenen Personen zu unterstützen, das automatische Ableiten von repräsentativen Instanzen, also typischen Beispielen für *valide* Prozessausführungen, aus Modellen. Auch Gegenbeispiele, also Beispiele für, gegenüber des Prozessmodells, *nicht-valide* Prozessinstanzen, können zur Unterstützung verwendet werden (Anforderung A2). Das ist besonders in deklarativen Sprachen anzunehmen, da diese meist regelbasiert sind und man somit jede einzelne Regel durch Positiv- und Negativbeispiele deduktiv beschreiben kann. Diese Annahmen basieren auf psychologischen Untersuchungen in unterschiedlichen Bildungssegmenten, die sich auf das Erlernen eines allgemeinen Prinzips, beispielsweise einer Sprache, auf Basis von Beispielen und Gegenbeispielen beziehen [20, 29, 198]. Ein Indiz für die Nützlichkeit der Kombination aus Positiv- und Negativbeispielen für das Verständnis *deklarativer* Prozessmodelle ist, dass in einer Studie zur Untersuchung der Verständlichkeit derartiger Modelle die verwendete Beispielsprache anhand von validen und nicht-validen Ausführungsspuren erklärt wird [73].

*Positiv- und Negativbeispiele*

Im Wesentlichen unterscheidet man – wie der aktuelle Abschnitt diskutiert – zwischen endogen-subjektiven und exogen-objektiven Einflüssen auf die Verständlichkeit von Prozessmodellen. Die Verständlichkeit von Prozessmodellen muss in vielen Fällen verbessert werden. Dazu ist die Entwicklung von Verfahren zur Translation in eine alternative Prozessmodellierungssprache oder in natürliche Sprache zu empfehlen. Zusätzlich besteht auch ein Bedarf an einer Technik zur Generierung beispielhafter oder vom Modell abweichender Prozessinstanzen.

### 1.2.2 *Interoperabilität*

Die große Vielfalt an Prozessmodellierungssprachen birgt nicht nur Herausforderungen für die Verständlichkeit von Modellen in ihrer Rolle als Kommunikationsmittel. Im diesem Abschnitt werden die Herausforderungen aus der Perspektive der *Interoperabilität*, also der Kompatibilität von Modellierungssprache und Werkzeugen des Geschäftsprozesslebenszyklus betrachtet. Prozessausführungssysteme und Technologien des *Process Mining* sind zwei Gruppen von Vertretern dieser Werkzeuge, die in der Regel die Semantik *einer* Ausführungssprache interpretieren können. Aus unterschiedlichen Gründen kann es notwendig sein, die konzeptuelle Prozessmodellierungssprache zu wechseln, was in den meisten Fällen auch einen Wechsel der Ausführungssprache nach sich ziehen würde. Schließlich wird für die vorliegende Arbeit davon ausgegangen, dass der Wechsel des Ausführungssystems und auch der Process-Mining-Werkzeuge unerwünscht ist.

*Unerwünschter Wechsel der Systemunterstützung*

Prozessausführungssysteme interpretieren zur Laufzeit des Prozesses Ereignisse, wie beispielsweise den Eingang eines Lieferauftrags per E-Mail, und reagieren darauf mit verschiedenen Aktionen. Diese könnten beispielsweise das Ablegen des

*Interoperabilität: Ausführungssysteme*

| | UML 2 AD | BPDM | BPMN 2.0 | EPC | Petri-Netze | CPN | YAWL | CMMN 1.1 | CLIMB | Declare | DPIL | TLA+ | DCR | EM-BrA²CE |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
| UML 2 AD | ✓ | ✓ | ✓ | – | – | – | – | – | – | – | – | – | – | – |
| BPDM | ✓ | ✓ | ✓ | – | – | – | – | – | – | – | – | – | – | – |
| BPMN 2.0 | ✓ | ✓ | ✓ | – | ✓ | ✓ | ✓ | – | – | – | – | – | – | ✓ |
| EPC | – | – | – | ✓ | ✓ | – | – | – | – | – | – | – | – | – |
| Petri-Netze | – | – | ✓ | – | ✓ | ✓ | ✓ | – | – | – | – | – | – | ✓ |
| CPN | – | – | ✓ | – | ✓ | ✓ | ✓ | – | – | – | – | – | – | ✓ |
| YAWL | – | – | ✓ | – | ✓ | ✓ | ✓ | – | – | – | – | – | – | ✓ |
| CMMN 1.1 | – | – | – | – | – | – | – | ✓ | – | – | – | – | – | – |
| CLIMB | – | – | – | – | – | – | – | – | ✓ | – | – | – | – | – |
| Declare | – | – | – | – | – | – | – | – | – | ✓ | – | – | ✓ | – |
| DPIL | – | – | – | – | – | – | – | – | – | – | ✓ | – | – | – |
| TLA+ | – | – | – | – | – | – | – | – | – | – | – | ✓ | – | – |
| DCR | – | – | – | – | – | – | – | – | – | ✓ | – | – | ✓ | – |
| EM-BrA²CE | – | – | ✓ | – | ✓ | ✓ | ✓ | – | – | – | – | – | – | ✓ |

Tabelle 1.2: Ausführungskompatibilität konzeptueller Prozessmodellierungssprachen

Lieferauftrags in einer Datenbank und die Koordination von Prozessbeteiligten durch Handlungsanweisungen beinhalten. Die notwendigen Informationen für die Auswahl der durchzuführenden Aktionen erhält das System durch ein ausführbares Prozessmodell, welches in der Regel von einem konzeptuellen Modell abgeleitet wird (Abbildung 1.1).

*Konzeptuelle Prozessmodellierungssprache vs. Ausführungssprache*

Vanderhaeghen et al. [183] bezeichnen die Fähigkeit der Interoperabilität von Prozessmodellen als fundamentalen Startpunkt für die Entwicklung unternehmensübergreifender Geschäftsprozesse. Im Kontext der langjährigen Diskussionen um das Thema *Borderless Enterprise* wird bereits die Wichtigkeit einer IT-gestützten, dynamischen Interaktion zwischen Organisationen diskutiert. Dieser Gedanke wird weiterverfolgt und hat sich unter dem Schlagwort *Kollaboratives Geschäftsprozessmanagement* etabliert. Die Fähigkeit zur kollaborativen Allokation von Ressourcen, zur Steuerung von Finanzen und Material sowie zum gezielten Austausch von Informationen und Daten wird als zentral angesehen. Dies setzt nicht nur eine technologische, sondern auch eine konzeptuelle Grundlage zur Verschränkung der Geschäftsprozesse voraus. Vor dem Hintergrund, dass jedes der interagierenden Unternehmen seine eigenen Methoden und Werkzeuge etabliert hat, ist die Interaktion auf Basis existierender Prozessmodelle zwar gewünscht, aber kaum unterstützt. Vanderhaeghen et al. argumentieren, dass diese Unterstützung unter anderem in der Fähigkeit liegt, Konzepten einer Prozessmodellierungssprache in eine andere übersetzen zu können.

*Interoperabilität: Kollaboratives Geschäftsprozessmanagement*

Im Prozesslebenszyklus sind weitere Szenarien denkbar, in denen der Wechsel der Prozessmodellierungssprache notwendig wird (Abschnitt 1.2.1). Weiterhin gibt es Szenarien, in denen ein hybrider Modellierungsansatz, also eine Modellierung mit imperativen und deklarativen Elementen gewünscht ist (Abschnitt 1.1.4). Aus Abbildung 1.6 wird ersichtlich, dass nur wenige konzeptuelle Prozessmodellierungssprachen auf dieselbe von Prozessausführungssystemen verwendete Ausführungssprache abgebildet werden. Die sich daraus ergebende Kompatibilität konzeptueller Prozessmodellierungssprachen ist in Tabelle 1.2 dargestellt und basiert auf Abbildung 1.6. Wie aus der Tabelle ebenfalls abzulesen ist, sind lediglich 64% der bekanntesten Prozessmodellierungssprachen überhaupt mit anderen Sprachen ausführungskompatibel. Betrachtet man die Interoperabilität aller Sprachpaare, so werden nur etwa 15% dieser Paare auf eine gemeinsame Ausführungssprache abgebildet oder können zumindest transitiv (z.B. *YAWL*) auf diese abgebildet werden.

*Sprachliche Limitierung der Ausführungssysteme*

Außerdem wird deutlich, dass ein größerer Teil der imperativen Sprachen untereinander kompatibel ist als die deklarativen oder die imperativen mit den deklarativen.

Zwar unterstützen einige Ausführungssysteme mehrere Ausführungssprachen, eine generelle Kompatibilität ist damit jedoch nicht gewährleistet. Folglich müsste zum Wiederherstellen der Operabilität des konzeptuellen Prozessmodells unter anderem der Wechsel des Ausführungssystems vorgenommen werden. Letzteres beinhaltet jedoch meist die visuelle Nutzerschnittstelle, also das Arbeitsinstrument der Prozessbeteiligten. Ein Wechsel dieses Instruments kann dadurch einen erhöhten Umschulungsaufwand nach sich ziehen. Zusätzlich kann die Einführung des neuen Ausführungssystems selbst problematisch sein, wenn das ursprüngliche System stark mit weiteren IT-Systemen verwoben ist und damit beispielsweise Kommunikationsprotokolle umsetzt, die vom neuen System nicht unterstützt werden. Die daraus resultierende, verbleibende Alternative ist die Abbildung der Teile des Modells, deren Ausführungssprache nicht vom Ausführungssystem unterstützt wird, auf eine kompatible Repräsentation (Anforderung A1) [101]. Der Begriff der *Interoperabilität* bezeichnet im Kontext der vorliegenden Arbeit demnach vor allem die Möglichkeit, mehrsprachig zu modellieren und dabei die IT-gestützte Ausführbarkeit der Prozessmodelle zu erhalten.

*Abbildung auf kompatible Repräsentation*

Ein vollkommen anderer Aspekt der Interoperabilität ist das Zusammenspiel zwischen den konzeptuellen Prozessmodellierungssprachen und sogenannten *Process-Mining*-Techniken [6]. Letztere ermitteln mittels Techniken des maschinellen Lernens aus Aufzeichnungen erfolgter Prozessdurchführungen ein zugrundeliegendes Prozessmodell in einer festgelegten Zielsprache. Die Anwendungen dieser Klasse von Techniken sind vielfältig. Einerseits werden auf diesem Weg *reale* Prozessverläufe modelliert und erlauben dadurch einen Vergleich mit den ursprünglich modellierten *möglichen* Verläufen. Andererseits gibt es Fälle, in denen eine erste Prozessmodellierung mittels dieser Techniken sinnvoll erscheint, anstatt den Prozess manuell zu modellieren. Die meisten dieser Techniken basieren auf statistischen Maßen. Gemäß der Natur statistischer Maße ermittelt man somit kein *garantiert* die Realität widerspiegelndes, sondern ein mit einer *gewissen Wahrscheinlichkeit* den Verlaufsaufzeichnungen entsprechendes Prozessmodell [6, S. 269ff]. Hinzu kommt, dass die grundlegende Idee von Process Mining die *Induktion* ist, also das abstrahierende Schließen vom Beobachteten auf eine allgemeine Erkenntnis. Im Kontext setzte das demnach voraus, dass die Beobachtung des Prozesses bis zu einem für das Verfahren notwendigen Grad vollständig ist. Beinhalten die bisherigen Aufzeichnungen der Ausführungen eines Prozesses eine bestimmte Kette von Aktivitäten nicht oder das Mining-Verfahren betrachtet sie als Ausnahme, dann kann es passieren, dass auch das erzeugte Modell diese Kette nicht beschreibt bzw. es stattdessen sogar verbietet, obwohl der tatsächliche Prozess sie gestattet. Demnach kann also eine Divergenz zwischen Realität, Ausführungsaufzeichnung und Modellierung entstehen. Das kann auch gewünscht sein, zum Beispiel wenn aus den Aufzeichnungen das *Standardvorgehen* in einem Prozess ermittelt werden soll. Dies setzt aber voraus, dass das jeweilige Process-Mining-Verfahren Beobachtungen *gezielt* ausschließt. Die Entwicklung eines solchen Verfahrens erfordert folglich eine gründliche Evaluation [6, p. 269ff]. Diese wird üblicherweise durch einen Abgleich von Prozessausführungsaufzeichnungen mit dem generierten

*Process Mining*

*Divergenz zwischen Realität, Aufzeichnung und Modellierung*

*Evaluation von Process-Mining-Techniken*

Modell realisiert. Reale Aufzeichnungen dieser Art beinhalten häufig unternehmenskritische Daten und werden, wenn überhaupt, modifiziert und anonymisiert öffentlich gemacht [41]. Außerdem ist die Qualität und Vollständigkeit der Aufzeichnungen ebenso häufig unklar. Um eine bessere Kontrolle über Inhalt und Qualität der Aufzeichnungen zu haben, wird zunehmend auf *künstlich* erzeugte Prozessaufzeichnungen zurückgegriffen, die beispielsweise mittels Prozesssimulation generiert werden. Derartige Techniken sind jedoch besonders für multiperspektivische, deklarative Prozessmodellierungssprachen rar (Anforderung A2).

*Generierung künstlicher Prozessausführungsaufzeichnungen*

Eine wesentliche Erkenntnis des aktuellen Abschnitts ist, dass die Interoperabilität verschiedener Prozessmodellierungssprachen mit Prozessausführungssystemen und Process-Mining-Technologien problematisch ist. Die starke Abhängigkeit dieser Werkzeuge von einer konkreten Prozessmodellierungssprache erschwert deren Wechsel und steht damit im Konflikt zu der in Abschnitt 1.2.1 identifizierten potentiellen Notwendigkeit für derartige Wechsel. Eine Möglichkeit der Abbildung zwischen verschiedenen Prozessmodellierungssprachen ist demnach obligatorisch. Um die Qualität von Process-Mining-Techniken für die gewünschte Sprache sicherzustellen, werden zusätzlich Simulationswerkzeuge benötigt, die künstliche Aufzeichnungen valider Prozessinstanzen generieren.

### 1.2.3 *Sprachevolution*

Domänenspezifische Modellierungssprachen – beispielsweise Prozessmodellierungssprachen – können sich, wie natürliche Sprachen auch, in einem Evolutionsprozess verändern. Einer der wesentlichen Unterschiede ist, dass im Falle der DSLs Evolutionsschritte in diskreten Zeitschritten gemessen werden können, während die Veränderung natürlicher Sprachen kontinuierlich erfolgt. Grundlage dafür ist, dass DSLs durch geänderte oder neue Anforderungen weiterentwickelt werden [186]. Damit ist es generell möglich, von verschiedenen Versionen einer DSL, beispielsweise einer Prozessmodellierungssprache zu reden. Die Art der Weiterentwicklung der Sprache kann in zwei Klassen unterteilt werden:

*Syntaktische vs. Semantische Evolution*

- *Syntaktisch:* Hinzufügen, Entfernen oder Verändern von Elementen der abstrakten Syntax *ohne* Veränderung der Ausdrucksmächtigkeit und
- *Semantisch:* Hinzufügen, Entfernen oder Verändern von Elementen der abstrakten Syntax *mit* Veränderung der Ausdrucksmächtigkeit.

Im aktuellen Abschnitt wird erläutert, inwiefern es möglich ist, für die Klasse der rein *syntaktischen* Evolutionsschritte, die Operabilität der gewünschten Prozessmodellierungssprache wiederherzustellen. Da zwei verschiedene Versionen *derselben* Sprache auch als zwei verschiedene Sprachen aufgefasst werden können [48], besteht ein starker Bezug zur Problematik der Interoperabilität und folglich auch zur Bewältigung dieses Problems durch Translationstechniken.

Die Weiterentwicklung einer Modellierungssprache zieht Konsequenzen hinsichtlich der bis dahin mit dieser Sprache modellierten Instanzen nach sich. Meyers und Vangheluwe schreiben in diesem Zusammenhang [129]:

> "When the [abstract] syntax of a modelling language evolves (i.e., the meta-model evolves), the most prominent side effect is that its in-

> stance models may no longer conform to the new meta-model. Therefore, the co-evolution (with evolution of their meta-model) of models has become an important research topic."

*Ungültigkeit nach Sprachevolutionsschritt*

Eine der wichtigsten Konsequenzen ist, dass bereits existierende Modellinstanzen nach einem Evolutionsschritt gegenüber der neuen Sprachversion ungültig sein können. Ein intuitiver Ansatz zur Behebung dieses Problems ist die manuelle Neumodellierung des Modellierungsgegenstands, was jedoch selten gewünscht ist. Levendoszky et al. formulieren den Grund dafür wie folgt [110]:

> "With the industrial applications of domain-specific modeling environments, models are valuable investments. If the modeling language evolves, these models must be seamlessly migrated to the evolved DSML. [...], migrating existing models to a new language is still a challenging task."

*Modellmigration*

Modelle sind wertvolle Investitionen der jeweiligen Unternehmen. Damit ist das zu präferierende Prinzip, um das Problem der Inkompatibilität zwischen Modell und neuer Sprachversion zu lösen nicht revolutionär, sondern ebenfalls evolutionär und wird *Migration* genannt. Sowohl die Spezifikation der Migration als auch die für die operative Nutzung notwendige Werkzeugunterstützung sind aktuelle Forschungsgegenstände und folgerichtig immer noch eine Herausforderung [129]. Ein wesentlicher Problempunkt stellt hierbei die Notwendigkeit da, für jeden Evolutionsschritt eine *Modelltransformation* vom veralteten Modell zum aktualisierten Modell angeben zu müssen [129]. Diese werden meist in Form von *Operationen* angegeben, welche das ursprüngliche Modellelement als Argument akzeptieren und als Resultat ein Modellelement in der Zielversion der Sprache liefern. Häufig wird hier auch von *Transformationsregeln* gesprochen.

*Optionale Operationen*

*Auflösbare Operationen*

*Nicht-auflösbare Operationen*

Gruschko et al. [69] unterscheiden zwischen optionalen, auflösbaren und nicht-auflösbaren Migrationsoperationen. Als *optional* gilt eine solche Operation, wenn existierende Modelle durch die zugrundeliegende Änderung an der Modellierungssprache ihre Gültigkeit nicht verlieren. Dazu zählen zum Beispiel die Einführung optionaler Beziehungen zwischen Modellelementen. Existierende Modelle müssen folglich nicht zwangsläufig migriert werden, um mit der neuen Version der Modellierungssprache konform zu sein. Als *auflösbar* gelten Operationen dann, wenn der Evolutionsschritt der Modellierungssprache automatisiert auf alle Instanzen übertragen werden kann. Ein Beispiel ist die Namensänderung eines Sprachelements, wobei diese Namensänderung für alle Instanzen, also Verwendungen des jeweiligen Sprachelements, übertragen werden können. *Nicht-auflösbare* Operationen behandeln Evolutionsschritte der Modellierungssprache, bei denen zusätzliche Informationen für ihre Ausführung notwendig sind. Ein Beispiel ist das Hinzufügen eines Attributs zu einem Sprachelement. Für existierende Modelle ist der Wert für die initiale Belegung unbekannt – es sei denn, man gibt in der entsprechenden Migrationsoperation einen Standardwert an, der dann in allen existierenden Instanzen als Wertebelegung für die neue Eigenschaft verwendet wird. Da das die Migration jedoch auf ein Standardszenario reduziert, ist diese Möglichkeit häufig zu ungenau. Stattdessen ist es in vielen Fällen der einzig vertretbare Weg, existierende Modelle an die weiterentwickelte Modellierungssprache manuell anzupassen. Gruschko et al. [69] schreiben hierzu jedoch:

> "Though the only correct solution in some cases, it can be a tedious job to perform manual co-evolutions on each model separately."

Für alle drei genannten Operationstypen besteht die Notwendigkeit, für jedes konsekutive Paar von Versionen derselben Sprache entweder die Änderungen von einer Version zur anderen formal zu spezifizieren und daraus automatisch Migrationsregeln abzuleiten oder letztere manuell zu erstellen. Betrachtet man die Anzahl verschiedener Prozessmodellierungssprachen (Abschnitt 1.2), dann ist ein erhöhter Grad an Automatisierung der Migration wünschenswert (Anforderung A4).

*Abwärtskompatibilität*

Wie im vorangegangenen Abschnitt angedeutet, kann sich die Evolution von konzeptuellen Prozessmodellierungssprachen auch auf die Interoperabilität auswirken. Das ist dann der Fall, wenn durch den jeweiligen Evolutionsschritt auch eine Anpassung der zugehörigen Prozessausführungssprache notwendig wird. Fest etablierte und in die IT-Infrastruktur des Unternehmens eingebettete Ausführungssysteme lassen sich besonders in größeren Unternehmen nicht immer ad-hoc aktualisieren. Zudem kann es auch hilfreich sein, das Erlernen syntaktischer Erweiterungen durch die Abbildung auf die Vorgängerversion der Sprache zu unterstützen. Damit ergibt sich die Anforderung der *Abwärtskompatibilität*. Diese gilt jedoch nur in speziellen Fällen der Sprachevolution. Ein beispielhafter Fall bezieht sich auf die Klasse der deklarativen Prozessmodellierungssprachen. Angenommen, eine solche Sprache beinhaltet in Version 1 folgende Regelschablonen[10]:

*Syntaktische Erweiterung deklarativer Sprachen*

- $\mathtt{precedence}(A, B)$: Vor der Ausführung von Aufgabe B muss mindestens einmal Aufgabe A ausgeführt worden sein.
- $\mathtt{response}(A, B)$: Nach der Ausführung von Aufgabe A muss vor dem Prozessende irgendwann Aufgabe B ausgeführt werden.

In Version 2 der Sprache kommt lediglich die Schablone $\mathtt{succession}(A, B)$ hinzu, welche eine Konjunktion der Schablonen $\mathtt{precedence}$ und $\mathtt{response}$ darstellt. Damit wird zwar die *Ausdrucksmächtigkeit* der Sprache nicht erweitert, jedoch die Menge ihrer Ausdrucksmittel – ihre Syntax. Folglich ist ein Modell in Sprachversion 2 auf Version 1 zurückzuführen, ohne die Ausführungssemantik des Modells zu verändern. Für diese Klasse an Sprachänderungen ist ein generisches, vollautomatisiertes Prinzip zur Rückführung hilfreich. Um ältere Modelle bereits mit der neuen Sprachversion und damit mit komfortableren Sprachelementen bearbeiten zu können, ist hier auch eine Konvertierung in umgekehrter Richtung wünschenswert (Anforderung A4).

*Syntaktische Erweiterung imperativer Sprachen*

Auch für imperative Sprachen besteht die Möglichkeit der Weiterentwicklung ohne Erweiterung der Ausdrucksmächtigkeit. Als Beispiel wird von einem minimalistischem Beispielprozess ausgegangen, der aus den Aktivitäten A und B besteht, von denen mindestens eine ausgeführt werden muss, um das Prozessziel zu erreichen. Weiterhin ist es auch erlaubt, beide auszuführen. Diese Semantik kann auf zumindest zwei Arten in boolescher Logik beschrieben werden:

a) $A \vee B$

b) $A \wedge B \oplus A \wedge \neg B \oplus \neg A \wedge B$

10 Ausschnitt aus der Declare-Syntax [145]

Variante *a* beschreibt beispielsweise die Logik des in der BPMN beschriebenen *Inclusive Gateways*, während Variante *b* eine ausführungssemantisch gleichwertige Konstruktion aus mehreren *Exclusive Gateways* und einem *Parallel Gateway* angibt. Ordnet man die beiden Varianten zwei unterschiedlichen Sprachversionen zu, dann bildet das Inklusiv-Oder eine syntaktische Erweiterung, ohne die Ausdrucksmächtigkeit der Logik zu erweitern. Folglich ist Anforderung A4 nicht nur für die Klasse der deklarativen Prozessmodellierungssprachen relevant.

Während Interoperabilität die Beziehung zwischen konzeptuellen und ausführbaren Prozessmodellierungssprachen thematisiert, fokussiert das Problem der syntaktischen Sprachevolution hingegen jeweils *eine* konzeptuelle Sprache. Die Nachvollziehbarkeit der Entwicklungsschritte und die Flexibilität der Migrationstechniken können sich direkt auf die Akzeptanz der Sprache selbst auswirken. Da die Evolutionsproblematik als Spezialfall der Übersetzungsproblematik aufgefasst werden kann, können potentielle Übersetzungswerkzeuge auch zur Migration von Prozessmodellen eingesetzt werden (Anforderung A1).

### 1.2.4 *Modellvergleich*

Der Vergleichbarkeit von Prozessmodellen ist ein weiterer Einflussfaktor für die Akzeptanz von Prozessmodellierungssprachen. Im aktuellen und letzten Abschnitt dieses Kapitels wird diskutiert, dass der Vergleich von Prozessmodellen an sich problematisch ist und zusätzlich erschwert wird, wenn die zu vergleichenden Modelle in unterschiedlichen Sprachen vorliegen. Eine Überführung eines der zu vergleichende Modelle in die Sprache des anderen ist eine Möglichkeit, ihre Vergleichbarkeit zu verbessern. Eine andere ist, anstatt die Modelle direkt zu vergleichen, die Ähnlichkeit des *Verhaltens* beider Modelle während der Ausführung oder Simulation zu bewerten.

Durch die mittlerweile große Verbreitung des Konzepts der Prozessmodellierung als probates Mittel zur Analyse und strukturierten (Neu-)Gestaltung unternehmensinterner Prozesse verfügen besonders größere Unternehmen häufig über *Prozessrepositorien*, die oft tausende Modelle beinhalten [27]. Um für diese Menge an Modellen adäquate Verwaltungswerkzeuge bereitzustellen, werden Ansätze im Bereich des *Modellvergleichs* entwickelt. Ausgangspunkt hierbei ist, dass Suchanfragen an diese Repositorien mittels meist partieller Prozessmodelle gestellt werden. Aus diesem Grund werden Maße benötigt, welche die Ähnlichkeit zweier Prozessmodelle quantifizieren. Becker und Laue [27] geben einen Überblick über gängige Maße und unterscheiden implizit zwischen zwei Kategorien:

*Prozessrepositorien*

a) Vergleich mittels Element- und Strukturanalyse der Modelle,

b) Vergleich auf Basis des beobachteten Verhaltens beider Modelle während der Ausführung.

*Vergleich der Modelle vs. Vergleich ihres Ausführungsverhaltens*

Maße der Kategorie *a* gehen meist von zwei wesentlichen Schritten aus. Der erste Schritt ist dabei die Identifikation der Korrespondenzen der Aktivitäten beider Prozessmodelle. Der zweite Schritt ist dann die eigentliche Berechnung der Modellähnlichkeit auf Basis struktureller Charakteristika. Dabei wird meist davon

*Imperative, kontrollflussorientierte Verfahren*

ausgegangen, dass beide Modelle in der gleichen Modellierungssprache vorliegen. Zahlreiche Verfahren nehmen zudem an, dass bereits alle Korrespondenzen der Aktivitäten vorliegen. Die Beschränkung des ersten Schrittes auf Aktivitäten suggeriert, dass die meisten Verfahren lediglich die Funktionale und die Verhaltensorientierte Perspektive berücksichtigen. Zudem gibt es ein deutliches Übergewicht hinsichtlich der Menge der Ansätze für imperative Prozessmodelle gegenüber ihren deklarativen Gegenstücken [24].

*Annahme: Zwei Modelle, eine Modellierungssprache*

Die oft verwendete Annahme, dass die zu vergleichenden Modelle in derselben Modellierungssprache vorliegen, kann ebenfalls hinderlich sein. Für den Anwendungsfall der Verwaltung von großen Prozessrepositorien bedeutet das beispielsweise, dass die Anfragen in derselben Sprache gestellt werden müssen, in der auch die Prozesse selbst modelliert sind. Zusätzlich bringt das die Forderung mit sich, dass alle im Repositorium enthaltenen Modelle in derselben Sprache abgebildet sind. Wie jedoch in Abschnitt 1.1.4, Abschnitt 1.2.1 und Abschnitt 1.2.2 erläutert, kann es aus verschiedenen Gründen zu einem Wechsel der Modellierungssprache kommen. Dieser Wechsel kann demselben Modellierungsparadigma folgen, also weiterhin imperativ oder weiterhin deklarativ sein. Im Kontrast dazu kann es jedoch auch zu einem Paradigmenwechsel oder gar einem hybriden Modellierungsansatz kommen.

*Vergleich imperativer mit deklarativen Prozessmodellen*

Hinsichtlich der Übertragbarkeit existierender Vergleichsverfahren aus Kategorie *a* auf den Vergleich imperativer mit deklarativen Prozessmodellen gibt es starke Vorbehalte [24]. Beim Vergleich zwischen imperativen und deklarativen Modellen müsste man Korrespondenzen zwischen „Erlaubnissen" und „Verboten" berechnen. Das ist in den seltensten Fällen möglich. Ein Beispiel dafür bieten die deklarativen Prozessmodelle aus Abbildung 1.5. Die textuelle Repräsentation der Modelle erschwert mithin die Verwendung der existierenden Verfahren zusätzlich, die auf dem direkten Vergleich der Modelle beruhen. Das gilt jedoch nicht für Vergleichsverfahren aus Kategorie *b*, die statt der Modelle beobachtete Ausführungspfade miteinander vergleichen und damit eine Aussage über die sogenannte *Spurähnlichkeit* (engl. *trace similarity*) treffen. Um jedoch die Ähnlichkeit zweier Modelle auf diesem Weg zu vergleichen, besteht die Notwendigkeit, *alle* möglichen Verläufe vorliegen zu haben. Aufgrund von potentiellen, uneingeschränkten Zyklen in Prozessmodellen ist dies im allgemeinen Fall jedoch nicht möglich. Hinzu kommt, dass durch die Beobachtung realer Prozessausführungen, die meist in Form von Log-Daten erfasst werden, nicht sichergestellt werden kann, dass alle möglichen Ausführungsverläufe erfasst werden. Die Wahrscheinlichkeit, auf diesem Wege nur die Standardpfade des Prozesses zu erfassen, ist sehr hoch [27]. Das Resultat wäre zwar eine Repräsentation des prototypischen Prozessverhaltens, nicht aber eine, die repräsentativ für das ursprüngliche Modell ist. Aus diesem Grund wird häufig auf die Funktionalität von Prozesssimulationswerkzeugen zurückgegriffen, die jedoch meistens nur auf Basis von Wahrscheinlichkeitsverteilungen Verzweigungen im Prozess abdecken – womit wiederum keine Garantie für die Erfassung aller möglichen Abläufe gegeben ist. Aus diesem Grund ist für den Vergleich von Prozessmodellen auf Basis der Spur-Ähnlichkeit ein Simulationswerkzeug vonnöten, welches für eine gegebene maximale Pfadlänge *alle* möglichen Pfade identifiziert

*Spurähnlichkeit*

(Anforderung A2). Damit ist für die zu vergleichenden Modelle und für die angegebene Pfadlänge eine verlässliche Aussage über die Spurähnlichkeit möglich.

*Vergleich mittels Graph-Editierabstand*

Eine weitere Möglichkeit, zwei Prozessmodelle zu vergleichen, bieten verschiedene Maße des *Editierabstands* zwischen diesen [27]. Je weniger Operationen notwendig sind, um einen Graphen in den anderen zu überführen, desto ähnlicher sind sie. Auch hier gehen die meisten Verfahren davon aus, dass die zu vergleichenden Modelle mittels derselben Sprache repräsentiert werden. Dennoch ist das Prinzip der Umformung eines Modells bis zur Gleichheit mit dem zweiten Modell ein bereits bekannter Weg, Modelle vergleichbar zu machen. Mit den schon in Abschnitt 1.2.3 erwähnten Modelltransformationen stehen Mittel zur Verfügung, um prinzipiell das eine Modell in die Sprache des Vergleichspartners zu übersetzen. Die Anwendbarkeit dieses Ansatzes kann hier jedoch nur für den konkreten Fall, für das konkrete Sprachpaar entschieden werden. Besonders beim Vergleich eines imperativen und eines deklarativen Prozessmodells wird die Entwicklung des Transformationssystems vor die große Herausforderung gestellt, die semantische Kluft zwischen der expliziten, graphorientierten Modellierung und der impliziten, regel- und logikorientierten Repräsentation zu überwinden (Anforderung A1).

*Vergleichbarkeit durch Modelltransformation*

Wie der aktuelle Abschnitt zeigt, ist der Vergleich von verschiedensprachigen Prozessmodellen problematisch und die gängigen Vergleichsprinzipien sind limitiert. Um den Vergleich auf Basis der Verhaltensähnlichkeit von Prozessmodellen zu unterstützen, können Simulationswerkzeuge als Generator für valide Prozessausführungsspuren verwendet werden (Anforderung A2). Um dagegen den Vergleich auf Modellebene zu verbessern, wird ein generisches Übersetzungsprinzip zwischen den jeweiligen Prozessmodellierungssprachen benötigt (Anforderung A1).

Abschnitt 1.2 beschreibt grundlegende Problembereiche, welche durch das Aufeinandertreffen unterschiedlicher Prozessmodellierungssprachen verursacht werden. Diese beinhalten Hindernisse sowohl in den Bereichen der Verständlichkeit und Interoperabilität als auch beim Modellvergleich und der Evolution verwendeter Prozessmodellierungssprachen. Eine gezielte Auswahl oder Kombination von Werkzeugen hat das Potential, sowohl menschlichen Nutzern als auch Systemen den Zugang zu unterschiedlichen Modellierungssprachen zu erleichtern. Diese Werkzeuge sind ein Prozesssimulator für die Erzeugung von Prozessausführungsspuren, ein Translationsprinzip für Prozessmodelle und ein System zur Generierung natürlichsprachlicher Prozessbeschreibungen.

# 2

# LÖSUNGSANSATZ UND FORSCHUNGSMETHODIK IM ÜBERBLICK

In diesem Kapitel wird bezüglich der identifizierten Probleme und Anforderungen (Abschnitt 1.2) ein Überblick über die im Rahmen der vorliegenden Arbeit entstandenen Lösungsansätze gegeben. Besonders das Zusammenspiel dieser Ansätze steht dabei im Fokus. Ihre Entwicklung orientiert sich grob an den Richtlinien der *Design-Science*-Methode, was ebenfalls in diesem Kapitel erläutert wird.

## 2.1 FOKUS UND LÖSUNGSANSATZ

Die in Abschnitt 1.2 zusammengefassten Anforderungen deuten bereits an, dass der Fokus der vorliegenden Arbeit auf Aspekten der *flexiblen Verwendung* von Prozessmodellierungssprachen liegt. Es wird argumentiert, dass der flexible Gebrauch verschiedener konzeptueller Prozessmodellierungssprachen durch Verständlichkeits-, Interoperabilitäts-, Evolutions- und Vergleichsprobleme erschwert oder sogar verhindert wird. Ziel ist es, Ansätze zu entwickeln, welche diese Probleme lösen. Der aktuelle Abschnitt gibt einen Überblick über diese Ansätze und ordnet ihnen die identifizierten, grundlegenden Anforderungen zu.

*Drei Ansätze: Simulation, Translation und Generierung natürlichsprachlicher Texte*

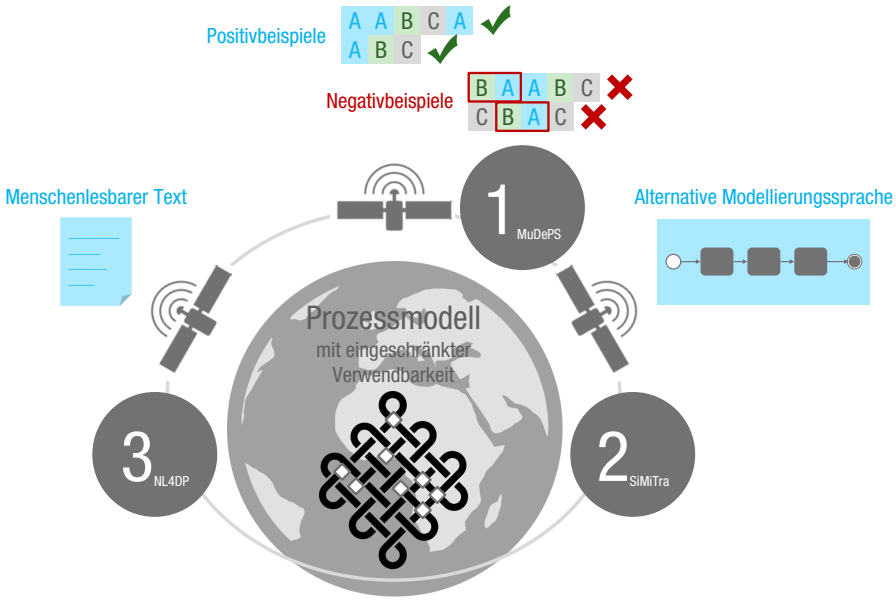



Abbildung 2.1: Lösungsansätze: Simulation, Translation, Natural Language Generation

In Abschnitt 1.2 werden vier sprachbezogene Problemfelder identifiziert, welche im weitesten Sinne die Verwendbarkeit eines Prozessmodells einschränken können. Die in Abbildung 2.1 dargestellten drei Ansätze *MuDePS*, *SiMiTra* und *NL4DP* verbessern die flexible Verwendung von Prozessmodellierungssprachen. Sie komplementieren bestehende Techniken und tragen somit zur Lösung der Probleme aus

| Ansatz | Funktion | Erfüllt Anforderung | Wiederverwendung in |
|---|---|---|---|
| MuDePS | Multi-perspektivische, deklarative Prozesssimulation | A2 | SiMiTra, NL4DP |
| SiMiTra | Translation von Prozessmodellen via Simulation und Process Mining | A1, A4 | NL4DP |
| NL4DP | Generierung natürlichsprachlicher Prozessbeschreibungen | A3 | - |

Tabelle 2.1: Erfüllung der Anforderungen durch MuDePS, SiMiTra und NL4DP

den genannten vier Bereichen bei. Das wird im Evaluationsteil der vorliegenden Arbeit geprüft und diskutiert. In Abschnitt 4.1, Abschnitt 5.1 und Abschnitt 6.1 wird mit einer problemorientierten Betrachtung existierender Ansätze eine Grundlage und Einordnung der drei Verfahren geschaffen.

*MuDePS*

Wie in Tabelle 2.1 beschrieben, ist *MuDePS* eine Technik für die Simulation multi-perspektivischer, deklarativer Prozessmodelle. Die hauptsächliche Neuerung gegenüber existierenden Ansätzen ist die Garantie, bei Bedarf *alle* möglichen Prozessausführungsspuren für eine gegebene maximale Länge der Spur zu erzeugen. Der zugrundeliegende Kern des Ansatzes greift dabei auf drei Logikformalismen zurück und überträgt das Simulationsproblem auf ein Erfüllbarkeitsproblem, welche dann mit generischen Verfahren gelöst werden.

*SiMiTra*

Mit *SiMiTra* wird ein Ansatz beschrieben, um Prozessmodelle von einer Prozessmodellierungssprache in eine andere zu übersetzen. Der wesentliche Unterschied zu bestehenden Ansätzen ist, dass es sich um ein *induktives* Verfahren handelt, wodurch es einerseits generisch ist und andererseits keinerlei Spezifikationen hinsichtlich notwendiger Modelltransformationen erfordert. Der Kern dieser Lösung ist das Zusammenspiel bekannter Techniken zur Simulation von Prozessmodellen und Process-Mining-Techniken zu deren automatisierter Erschließung auf Basis von in Log-Daten vorliegenden Abbildern historischer Prozess-Spuren.

*NL4DP*

Das verbleibende Verfahren, *NL4DP*, transformiert multi-perspektivische, deklarative Prozessmodelle in natürlichsprachlichen, menschenlesbaren Text. Grundsätzlich ist es das erste Verfahren für deklarative Prozessmodellierungssprachen, wobei die grundlegende Neuerung eine auf Mustersuche basierende Technik zur Selektion und Extraktion der Informationen aus dem Prozessmodell ist. Das NL4DP-Verfahren liegt nur begrenzt im Fokus der vorliegenden Arbeit, da es auf den bereits vorher genannten Techniken MuDePS und SiMiTra aufbaut und für diese einen weiteren Anwendungsraum exemplarisch untersucht, welcher im direkten Zusammenhang mit den in Abschnitt 1.2 identifizierten Anforderungen steht.

In Tabelle 2.1 ist überblicksweise dargestellt, welche der drei genannten Ansätze einen Beitrag zu welchen der identifizierten Anforderungen liefert. Die Erfüllung derselben wird im Evaluationsteil der vorliegenden Arbeit detailliert betrachtet. Aus diesem Grund werden die Anforderungen an dieser Stelle lediglich dazu verwendet, das Zusammenspiel der drei Techniken zu beschreiben.

*Zusammenspiel von MuDePS, SiMiTra und NL4DP*

Der NL4DP-Ansatz erfüllt Anforderung A3 insofern, als er in der Lage ist, für eine bestimmte multi-perspektivische, deklarative Prozessmodellierungssprache, ähnlich der in Abbildung 1.5 verwendeten, automatisiert eine natürlichsprachliche Beschreibung zu generieren. Die Beschränkung auf lediglich eine einzelne Prozessmodellierungssprache würde das Einsatzspektrum der Technik jedoch signifikant einschränken (Abbildung 1.6). Da die unterstützte Sprache aber eine größere Ausdrucksstärke als alle anderen bisher entwickelten deklarativen Prozessmodel-

lierungssprachen besitzt, können Modelle, die mittels ausdrucksschwächerer Sprachen modelliert sind, potentiell in die von NL4DP unterstützte übersetzt werden. Hierfür existieren jedoch derzeit keine Verfahren. Auch für imperative Prozessmodellierungssprachen existieren nur sehr wenige Ansätze wie NL4DP. Wäre es möglich, diese imperativen Modelle auf die NL4DP-kompatible deklarative Sprache abzubilden, dann wäre es auch möglich, die Ausführungssemantik des imperativen Modells mindestens anteilig mit natürlichsprachlichen Mitteln zu beschreiben. An dieser Stelle kommt mit SiMiTra ein generisches Prinzip zur Übersetzung von Prozessmodellen ins Spiel und unterstützt somit die Anwendbarkeit von NL4DP.

*NL4DP + SiMiTra*

SiMiTra erweitert nicht nur das Einsatzspektrum von NL4DP, sondern behandelt selbst die Anforderungen A1 und A4. Aufgabe der Technik ist die Übersetzung von Prozessmodellen. Ihre zentrale Eigenschaft ist die Unabhängigkeit davon, dass die verwendeten Prozessmodellierungssprachen identisch oder zumindest demselben Modellierungsparadigma zuzuordnen sind. Grund dafür ist, dass das Verfahren nicht auf den sonst üblichen Techniken aus dem Bereich der Modelltransformationen basiert, sondern Prozessmodelle übersetzt, indem es für das Quellmodell mittels Simulation Ausführungsspuren künstlich generiert und diese dann mit einem induktiven Verfahren zur Rekonstruktion eines Prozessmodells in der Zielsprache verwendet. Damit ist SiMiTra als induktives Verfahren zu klassifizieren. Anforderung A4 betrachtet eigentlich die Evaluation *einer* Prozessmodellierungssprache. Jedoch können zwei verschiedene Versionen derselben Sprache auch als zwei unterschiedliche Sprachen betrachtet werden. Grundsätzlich ist damit die Anwendung einer Translationstechnik wie SiMiTra nicht ausgeschlossen. Die Vorteile des geringeren manuellen Aufwands und die Nachteile des induktiven Verfahrens müssen im konkreten Fall sorgfältig abgewogen werden. Weiterhin müssen für das konkrete Sprachpaar jeweils ein Simulations- und ein Process-Mining-Verfahren zur Verfügung stehen. MuDePS ist ein Simulationsverfahren für multi-perspektivische, deklarative Prozessmodelle, welches demnach den SiMiTra-Ansatz unterstützt.

*SiMiTra + MuDePS*

MuDePS selbst erlaubt es, für ein gegebenes multi-perspektivisches Prozessmodell gezielt gültige oder ungültige Ausführungsspuren zu generieren (A2). Sollte zum Vergleich von Prozessmodellen ein Maß für die Verhaltensähnlichkeit gewählt werden, kann mittels MuDePS ein, bezüglich der gewählten maximalen Länge der Ausführungsspuren, vollständiger Satz dieser Spuren erzeugt werden.

Damit ergibt sich ein Abhängigkeitsgraph zwischen den drei Ansätzen, der mit MuDePS als unabhängiger Ansatz seine Basis hat, mit SiMiTra einen Mediator zwischen MuDePS und NL4DP beinhaltet und schließlich in eben letzterem seinen letzten Knoten findet.

Ein Beispiel für einen integrierten Arbeitsprozess ist das folgende Szenario: Dem Nutzer ist ein Prozessmodell gegeben, welches er aus Gründen der Kompatibilität mit einem unternehmensweit etablierten Prozessausführungssystem in eine bestimmte imperative Sprache übersetzen soll. Das Modell ist in einer multi-perspektivischen, deklarativen Modellierungssprache formuliert, welche dem Nutzer unbekannt ist. Mittels NL4DP generiert er zunächst eine natürlichsprachliche Beschreibung des Prozesses und erschließt sich so dessen Semantik. Da ihm für die gewünschte Zielsprache kein direktes Translationssystem zur Verfügung steht, greift der Nutzer auf SiMiTra zurück. Er verwendet zunächst MuDePS,

*Beispiel für integrierten Arbeitsprozess*

um eine größere Menge an Prozessausführungsspuren zu generieren. Mittels einer verfügbaren Process-Mining-Technologie erzeugt er daraus eine Rohfassung des Zielmodells. Anhand der natürlichsprachlichen Beschreibung des deklarativen Quellmodells in Kombination mit exemplarischen Ausführungsspuren überarbeitet er schließlich eventuelle Unzulänglichkeiten. Somit ist es dem Nutzer durch eine Kombination der drei Techniken MuDePS, SiMiTra und NL4DP trotz völliger Unkenntnis der Quellsprache gelungen, ein Prozessmodell in eine bestimmte Zielsprache zu überführen.

Sowohl aus den Anforderungen als auch aus den Erläuterungen dieses Abschnittes wird deutlich, dass die vorliegende Arbeit hauptsächlich Werkzeuge liefern soll, mit denen potentiell die in Abschnitt 1.2 diskutierten Ziele erreicht werden können. Die Analyse zu welchem Grad jedes der Werkzeuge einen Beitrag dazu liefert, ist nicht im Fokus dieser Arbeit, da dies stark vom konkreten Anwendungsfall abhängt. Jedoch bilden die drei Verfahren eine Grundlage für eine weitere Arbeit, die sich eben mit dieser Thematik auseinandersetzt.

## 2.2 FORSCHUNGSMETHODIK

*Design Science*

Das Vorgehen der vorliegenden Arbeit deckt sich weitestgehend mit der allgemeinen, auf Forschungen im Bereich der Informationssysteme bezogene Forschungsmethodik der *Design Science* [187]. Die Methodik setzt sich aus sieben Richtlinien zusammen, die im aktuellen Abschnitt kurz beschrieben werden. Weiterhin werden die Bezüge zwischen diesen und Abschnitten der vorliegenden Arbeit in Tabelle 2.2 zusammengefasst. Dies dient der Grobprüfung der Vollständigkeit dieser Arbeit in Bezug auf die Einhaltung der Richtlinien einer angemessenen und bewährten Methodik.

Die Richtlinien geben einen groben Rahmen vor, die zur zufriedenstellenden Beantwortung einer zentralen Forschungsfrage führen sollen. Für die vorliegende Arbeit wird diese wie folgt formuliert:

*Forschungsfrage*

> Welche Techniken können eine Erhöhung der Flexibilität hinsichtlich der Auswahl und Verwendung einer geeigneten Prozessmodellierungssprache bewirken und damit die Akzeptanz letzterer steigern?

Diese Frage beinhaltet zumindest zwei Untersuchungsrichtungen: *(i)* Inwiefern stehen solche Hilfsmittel zur Verfügung oder können solche Hilfsmittel bereitgestellt werden und *(ii)* zu welchem Grad können diese zur Lösung des Flexibilitätsproblems beitragen. Die vorliegende Arbeit fokussiert zum Großteil die erstgenannte Forschungsrichtung, tangiert aber auch Aspekte letzterer, wie die nachfolgende Ausrichtung an der Methode der Design Science zeigt.

*Design als Artefakt*

Die sieben Richtlinien der Design Science sind in Tabelle 2.2 im Überblick dargestellt. Das Ziel der gesamten Methodik ist die Erzeugung zweckmäßiger Artefakte (Richtlinie 2). Im Rahmen der vorliegenden Arbeit werden Artefakte – in Form eines Konzepts und einer Implementierung – zur logikbasierten Generierung von Prozessausführungsspuren für multi-perspektivische, deklarative Prozessmodelle (Abschnitt 4.2), für die Prozessmodell-Translation mittels Prozesssimulation und Process Mining (Abschnitt 5.2) sowie zur Generierung natürlichsprachlicher Beschreibungen für multi-perspektivische, deklarative Prozessmodelle (Abschnitt 6.2)

| Nr. | Richtlinie | Kapitel/Abschnitt |
|---|---|---|
| 1 | Problemrelevanz (engl. *problem relevance*) | Abschnitt 1.2 |
| 2 | Design als Artefakt (engl. *design as an artifact*) | Abschnitt 4.2, Abschnitt 5.2, Abschnitt 6.2, Kapitel 7 |
| 3 | Design als Suchprozess (engl. *design as a search process*) | Kapitel 3, Abschnitt 4.1, Abschnitt 5.1, Abschnitt 6.1 |
| 4 | Forschungsbeiträge (engl. *research contributions*) | Abschnitt 9.1 |
| 5 | Veröffentlichung der Forschung (engl. *communication of research*) | Auflistung der Themenbezogene Publikationen |
| 6 | Wissenschaftliche Sorgfalt (engl. *research rigor*) | Kapitel 4 bis Kapitel 8 |
| 7 | Bewertung des Designs (engl. *design evaluation*) | Kapitel 8 |

Tabelle 2.2: Einordnung der Arbeit in die Forschungsmethodik *Design Science*

entwickelt. Jedes Konzept ist implementiert – also in ein Softwareartefakt oder eine Instanziierung existierender Artefakte überführt (Kapitel 7). Damit werden die Fähigkeiten von Organisationen durch eine flexiblere Wahl der Prozessmodellierungssprache erweitert.

Mit der Akzeptanz von Prozessmodellierungssprachen wird ein relevanter Problembereich betrachtet, welcher mehrere noch ungelöste respektive nur zum Teil gelöste Probleme beinhaltet (Richtlinie 1). *Problemrelevanz*

Der Eigenentwicklung soll immer eine umfassende und sorgfältige Sichtung existierender Lösungsansätze vorausgehen (Richtlinie 3), um die Wiederverwendung vorhandener, adäquater Mittel sicherzustellen. Auch die Identifikation weiterer existierender Mittel zur Bewältigung der Problemstellung ist erwünscht. Aus diesem Grund werden in Kapitel 3 unter anderem Erkenntnisse aus verschiedenen Domänen zusammengefasst. *Design als Suchprozess*

Die Zweckmäßigkeit und der Beitrag der eigens erzeugten Artefakte richtet sich nach der *Behandlung* ungelöster, nicht vollständig oder nicht effizient gelöster Probleme (Richtlinie 4). Aus den Schlussfolgerungen der Problemanalyse in Abschnitt 1.2 gehen vier grundlegende Anforderungen hervor, deren Bewältigung einen Beitrag zur Forschung in der Domäne der Prozessmodellierungssprachen darstellt. Dies wird in Abschnitt 9.1 diskutiert. *Forschungsbeiträge*

Die Erkenntnisse der Forschungsarbeit müssen zudem veröffentlicht werden, was eine Publikation sowohl innerhalb der wissenschaftlichen Gemeinschaft (Auflistung der Themenbezogene Publikationen) als auch Anwendungen in der wirtschaftlichen Gemeinschaft einschließt (Richtlinie 5). *Veröffentlichung der Forschung*

Die produzierten Artefakte müssen nachvollziehbar und korrekt definiert, formal beschrieben und kohärent sowie in sich konsistent sein (Richtlinie 6). Bei der Anwendung von Methode zu Konstruktion und Evaluation der Artefakte ist auf größte Sorgfalt zu achten. In den einzelnen Publikationen, wie auch in der vorliegenden Arbeit ist die Wahl jeder Methode begründet und die Ausführungen erfolgte gründlich. Die Erfüllung dieser Richtlinie ist keinen einzelnen Kapitel der Arbeit zuzuordnen. Vielmehr ist sie anhand der Kombination aus Analyse des Ist-Standes, der Entwicklung fehlender oder der Verbesserung existierender Lösungen sowie ihre gründliche Evaluation ersichtlich (Kapitel 4 bis Kapitel 8). *Wissenschaftliche Sorgfalt*

Eine detaillierte und aussagekräftige Einschätzung der Qualität und Leitungsfähigkeit muss mittels anerkannter Methoden geprüft und diskutiert werden (Richtlinie 7). Die in dieser Arbeit verwendeten Methoden sind statische Komplexitätsanalysen, dynamische Leistungstests, kontrollierte Experimente, Simulationen, funktio- *Bewertung des Designs*

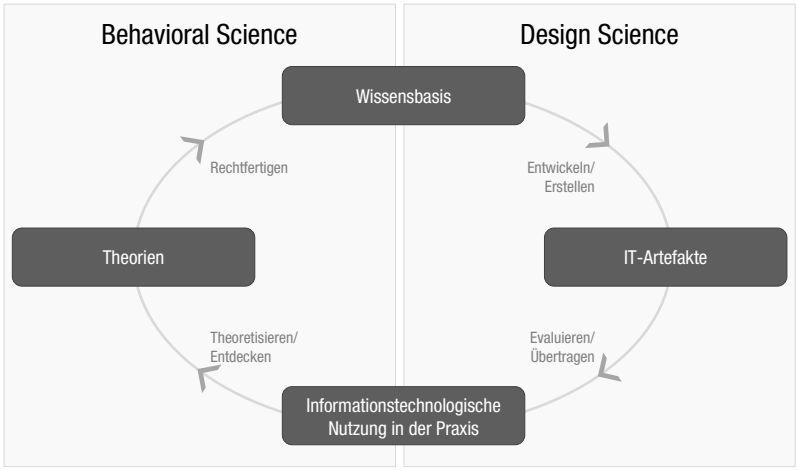



Abbildung 2.2: Der Forschungszyklus für Informationssysteme (basierend auf: [137])

nelle Tests und deskriptive Argumentationen und Szenariobeschreibungen [187] für die Anwendbarkeit und Nützlichkeit der Ansätze.

*Komplementär zur Verhaltensforschung*

Die Methodik der Design Science überschneidet sich teilweise mit der ansonsten komplementären Methodik der Behavioral Science, weswegen die Distanzierung von den Zielen letzterer den Abschluss dieses Kapitels bildet. Die Verhaltensforschung ist auf die Beantwortung der Frage „was ist wahr" ausgerichtet, während die Design Science eine eher pragmatische Stellung einnimmt und prüft, „was effektiv ist". Der grundsätzliche Unterschied ist folglich, dass die Erkenntnisse der Verhaltensforschung theoretischer Natur sind und diese als Grundlage für die Anwendung der Design-Science-Methode dienen können. Gleichzeitig kann der Einsatz von IT-Artefakten in der Praxis neue Fragestellungen und Theorien hervorbringen, deren Rechtfertigung wiederum im Bereich der Verhaltensforschung liegt. Diese wechselseitige Beziehung ist in Abbildung 2.2 dargestellt und wird als Leitwerk für die vorliegende Arbeit verwendet, da sich die im ersten Kapitel beschriebene Problemstellung auf die Schnittstelle zwischen Prozessmodell und den Beteiligten des Prozesslebenszyklus fokussiert. Letzterer definiert gleichzeitig die praktischen Einsatzszenarien der im Rahmen der vorliegenden Arbeit entwickelten Techniken.

# Teil II

# LÖSUNGSKONZEPTION

Nach der Identifikation von vier Problembereichen für die Akzeptanz konzeptueller Prozessmodellierungssprachen im ersten Teil der Arbeit werden in den hier folgenden Kapiteln drei Lösungskonzepte beschrieben. Diese beinhalten einen Ansatz für die Simulation deklarativer, multi-perspektivischer Prozessmodelle, ein generisches Prinzip zur Translation von Prozessmodellen in alternative Prozessmodellierungssprachen sowie einen Ansatz zur Generierung natürlichsprachlicher Prozessbeschreibungen auf Basis deklarativer, multi-perspektivischer Prozessmodelle. Die in den Konzeptkapiteln vorausgesetzten Grundlagen werden in einem einführenden Kapitel zusammengefasst.

# 3 GRUNDLAGEN

In diesem Kapitel werden einige Grundlagen zusammengefasst, welche den Zugang zu den Beiträgen der vorliegenden Arbeit erleichtern sollen. Die Selektion und Darstellung dieser Grundlagen ist naturgemäß unvollständig und der Autor verweist für weitere Informationen auf die angegebene Literatur. Außerdem sind die einzelnen Hauptabschnitte dieses Kapitels voneinander unabhängig. Leser mit dem entsprechenden Hintergrundwissen oder sehr differenziertem Interesse an den in dieser Arbeit vorgestellten Beiträgen können somit beliebig viele Abschnitte überspringen. Um diese Entscheidung zu erleichtern, wird am Beginn jedes Hauptabschnitts der inhaltliche Bezug zwischen Grundlage und einem oder mehreren der drei in Abschnitt 2.1 skizzierten Lösungskonzepte hergestellt. Tabelle 3.1 fasst diese Bezüge zusammen.

| Ansatz | Abschnitte für Beiträge | Abschnitte für Verwandte Arbeiten |
|---|---|---|
| MuDePS | 3.1.5, 3.2, 3.3 | 3.1.1, 3.1.2, 3.1.3, 3.1.4, 3.3 |
| SiMiTra | 3.2.5, 3.3, 3.4 | 3.1.1, 3.1.2, 3.1.3, 3.1.4, 3.5 |
| NL4DP | 3.1.5, 3.6 | 3.1.3, 3.6 |

Tabelle 3.1: Relevanz der Abschnitte dieses Kapitels für die verschiedenen Teile der Arbeit

Tabelle 3.1 zeigt, welche der in diesem Kapitel zusammengefassten Grundlagen für welche der drei Lösungskonzepte relevant sind (mittlere Spalte). Zusätzlich werden auch Hinweise gegeben welche der Grundlagen für das Verständnis verwandter Arbeiten benötigt werden (rechte Spalte).

## 3.1 AUSWAHL EXEMPLARISCHER PROZESSMODELLIERUNGSSPRACHEN

Die vorliegende Arbeit beschäftigt sich hauptsächlich mit der Simulation, der Translation und der automatisiert-natürlichsprachlichen Beschreibung von Prozessmodellen. Um diese Konzepte anhand von beispielhaften Prozessmodellen illustrieren zu können, werden im Verlauf der Arbeit sowohl imperative als auch deklarative Prozessmodellierungssprachen benötigt. Im nachfolgenden Abschnitt werden die ausgewählten imperativen und danach die deklarativen Prozessmodellierungssprachen vorgestellt.

Als erstes werden im nächsten Abschnitt die *Petri-Netze* vorgestellt, da diese die grundlegende Semantik der meisten imperativen Prozessmodellierungssprachen beschreiben. Zudem basieren einige der zur Evaluation (Abschnitt 8.4.2) verwendeten Techniken zur Translation auf dieser Notation. Im weiteren Verlauf wird unter anderem auch die Sprache BPMN vorgestellt, welche im Evaluationsteil als Beispielsprache fungiert. Gleiches gilt auch für die beiden deklarativen Sprachen Declare und DPIL. Zusätzlich nutzen der im Rahmen dieser Arbeit entwickelte

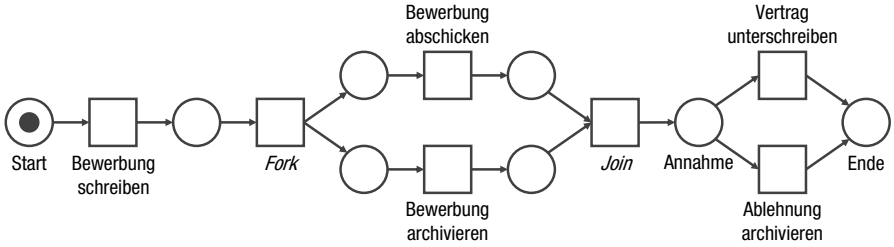



Abbildung 3.1: Petri-Netz zur Modellierung eines vereinfachten Bewerbungsprozesses

Simulationsansatz und die NLG-Technik exemplarisch DPIL als Sprache zur Repräsentation der Eingabemodelle. Weiterhin werden alle nachfolgend aufgeführten Sprachen in mindestens einem der später vorgestellten, bereits existierenden Simulationsansätze verwendet. Da diese in der ebenfalls später beschriebenen Translationstechnik für Prozessmodelle Anwendung finden können, ist ein grundlegendes Verständnis dieser Sprachen notwendig.

### 3.1.1 *Petri-Netze*

*Petri-Netze*

*Petri-Netze* sind ursprünglich aus den endlichen Zustandsautomaten zur Beschreibung von Schaltvorgängen in der Elektrotechnik verwendet worden. Der Entwickler, Carl Adam Petri, war dabei besonders darauf fokussiert, eine Beschreibungsmöglichkeit für *nebenläufige* Schaltvorgänge zu finden. Das solide mathematische Fundament sorgte in Kombination mit einer generischen, domänenunabhängigen Notation für eine weite Verbreitung der Petri-Netze, weit über die Schaltungstechnik hinaus. Die Modellierung von Geschäftsprozessen ist dabei nur eines der zahlreichen Anwendungsgebiete und auch die Notation selbst ist mittlerweile in zahlreichen Ausprägungen verfügbar. In der vorliegenden Arbeit wird von der gebräuchlichsten und in [181] anschaulich zusammengefassten Notation ausgegangen.

*Plätze und Transitionen*

Petri-Netze sind gerichtete Graphen mit zwei verschiedenen Knotentypen:

- *Plätze* (engl. *places*) symbolisieren Zustände und werden als Ellipsen oder Kreise dargestellt.
- *Transitionen* (engl. *transitions*) symbolisieren Zustandsübergänge und im Prozesskontext Aktivitäten. Sie werden als Quadrate oder Rechtecke dargestellt.

Plätze und Transitionen – und nur diese – werden über gerichtete Kanten verbunden. Es ist nicht erlaubt, Platz mit Platz oder Transition mit Transition zu verbinden. Ein Platz $p$ ist ein *Eingabeplatz* für Transition $t$, wenn eine ausgehende Kante von $p$ zu $t$ führt. Verläuft die Kante entgegengesetzt, ist $p$ gegenüber $t$ ein *Ausgabeplatz*. Plätze können sogenannte *Markierungen* (engl. *tokens*) enthalten. Markierungen symbolisieren den Zustand des modellierten Prozesses und werden als schwarze Punkte innerhalb der Plätze dargestellt. Abbildung 3.1 zeigt das vereinfachte Beispiel eines Bewerbungsprozesses.

*Eingabeplatz*

*Ausgabeplatz*

*Markierungen*

Gemäß der ursprünglichen Idee aus der Abbildung von Schaltvorgängen sind Petri-Netze in der Lage, Abläufe zu parallelisieren. Das ist in der Abbildung mit

der Transition *Fork* dargestellt, auf die statt eines zwei Plätze folgen. Entscheidungen werden durch eine umgekehrte Konstellation, bestehend aus einem Platz, auf den zwei Transitionen folgen, dargestellt. In der Abbildung ist diese Konstellation um den Platz *Annahme* zu sehen. Zu beiden Verzweigungsarten gehört jeweils eine symmetrisch gespiegelte Zusammenführung der Zweige, die als *Join* bezeichnet werden.

*Workflow Nets*

In der Domäne der Geschäftsprozesse wird in der Regel eine spezielle Form der Petri-Netze, die sogenannten *Workflow Nets* verwendet. Diese unterscheiden sich in drei Eigenschaften von allgemeinen Petri-Netzen. Demnach ist ein Petri-Netz ein Workflow Net, wenn:

*Definierter Start- und Ende-Platz*

- Das Petri-Netz die zwei speziellen Plätze *Start* und *Ende* beinhaltet, die dadurch gekennzeichnet sind, dass *Start* keine eingehenden und *Ende* keine ausgehenden Kanten aufweist,

*Free-Choice-Eigenschaft*

- Die *free-choice*-Eigenschaft erfüllt ist, d.h. Entscheidungen nur von *einem* Platz ausgehen und nicht den Markierungszustand anderer Plätze berücksichtigen und

*Stark Zusammenhängender Graph*

- Es *stark zusammenhängend* ist, also alle Plätze und Transitionen durch gerichtete Kanten beginnend mit *Start* und endend mit *Ende* zu erreichen sind.

Damit ist das in Abbildung 3.1 dargestellte Petri-Netz auch gleichzeitig ein Workflow Net.

*Ausführung durch Markierungszustand und Feuerregel*

Die Ausführungssemantik von Petri-Netzen ist durch Markierungszustände und die sogenannten *Feuerregeln* für Transitionen definiert. Transitionen können in folgender Form aktiv werden:

- Eine Transition gilt als aktiviert, wenn jeder Eingabeknoten mindestens eine Markierung aufweist.
- Eine Transition kann *feuern*, wobei sie von jedem Eingabeknoten eine Markierung konsumiert und für jeden Ausgabeknoten eine neue generiert.

Folglich muss für ein Workflow Net der *Start*-Platz mit einer Markierung initialisiert werden, damit das Netz überhaupt ausgeführt werden kann. Jeder Zustand, in dem eine Markierung in einem anderen Platz als *Start* vorhanden ist, ist ein Zustand mit dargestelltem Ausführungsfortschritt.

*Nur Kontrollfluss*

Diese klassische Form der Petri-Netze unterstützt somit lediglich die Funktionale und die Verhaltensorientierte Perspektive (Abschnitt 1.1.2) des Prozesses und eignen sich daher nur für die Darstellung des Kontrollflusses. Viele höhere Formen der Petri-Netze können prinzipiell auch zwischen Markierungstypen unterscheiden, die als *Markierungsfarben* bezeichnet werden. Transitionen werden so ausschließlich durch die Verfügbarkeit der korrekten Anzahl der Markierungen eines bestimmten Typs aktiviert werden. Damit ist dann auch eine Repräsentation der Datenperspektive möglich. Einige erweiterte Notationen können auch die Zeitperspektive berücksichtigen und somit eine Dauer für Aktivitäten angeben. Auch die Dekomposition komplexer Netze mittels hierarchischer Netz-Subnetz-Beziehungen

*Höhere Petri-Netze*
*Markierungsfarben*

ist möglich. Die meisten der in der vorliegenden Arbeit diskutierten Ansätze, die auf der Petri-Netz-Notation basieren, beschränken sich jedoch auf die klassische Darstellung. Zudem definieren viele der existierenden imperativen Prozessmodellierungssprachen ihre Ausführungssemantik durch eine Abbildung auf klassische Petri-Netze. Für größere Ausdrucksmächtigkeit wird meist auf eine der anderen, nachfolgenden Prozessmodellierungssprachen zurückgegriffen. Aus diesem Grund beschränkt sich dieser Abschnitt auf eine Beschreibung der klassischen Notation.

### 3.1.2 *Coloured Petri Nets*

*CPN*

Die Modellierungssprache *Coloured Petri Nets* (kurz: *CPN)* stellt eine rückwärtskompatible Erweiterung der allgemeinen Petri-Netz-Notation dar [90]. Das bedeutet, dass auch ein CPN-Modell aus Plätzen und Transitionen besteht, die durch gerichtete Kanten verbunden sind. Die erste Erweiterung ist die Möglichkeit, an

*Beschriftungen*

alle genannten Elemente eine *Beschriftung* anzufügen. Diese werden in einer Erweiterung der funktionalen Programmiersprache *StandardML*, der sogenannten *CPN ML* formuliert. Wie in einem allgemeinen Petri-Netz wird auch in einem CPN der Zustand des modellierten Systems über Markierungen angegeben. Als nächste Erweiterung erlaubt die CPN-Notation das Anfügen eines Datenwerts pro Markierung, welcher, wie bereits erwähnt, als Markierungsfarbe bezeichnet wird. Damit wird der Zustand des Systems aus der Kombination der Anzahl von Markierungen pro Platz und ihrer jeweiligen Farbe repräsentiert. Mithilfe der CPN ML kann für jeden Platz der Satz gültiger Markierungsfarben festgelegt werden. Das ist semantisch gleichbedeutend mit der Zuweisung gültiger Datentypen. In Abbildung 3.2 ist ein Kommunikationsprotokoll mittels CPN modelliert. Die gültigen Markierungen der Plätze C und D sind dabei auf Sequenznummern der vom Protokoll zu verarbeitenden Pakete beschränkt (NO). Die Plätze A und B beschränken die Menge gültiger Datenwerte hingegen auf eine Kombination aus Sequenznummer und Datenpaket (NOxDATA), wobei sich ein Datenpaket aus allen möglichen Zeichenketten zusammensetzen darf. Die zugehörigen Definitionen in CPN ML sind in Codeausschnitt 3.1 dargestellt.

```
colset NO = int;
colset DATA = string;
colset NOxDATA = product NO * DATA;
```

Codeausschnitt 3.1: Zuweisung von Datentypen in der CPN ML

*Initiale Markierung*

Auch die Festlegung von initialen Markierungen für jeden Platz erfolgt mittels der CPN ML. Die Beschriftung 1`1 in der oberen rechten Ecke des Platzes NextSend bedeutet beispielsweise, dass dieser Platz initial mit *einer* Markierung des Typs NO und mit dem Wert 1 versehen ist. Damit wird sichergestellt, dass das erste gesendete Datenpaket auch die Sequenznummer 1 aufweist. Der Platz DataReceived verfügt dagegen initial über eine Markierung des Typs DATA und mit einer leeren Zeichenkette als Wert, was bedeutet, dass der Empfänger zu Anfang keine Daten empfangen hat. Mit AllPackets wird eine Konstante als Multimenge definiert, welche insgesamt sechs initiale Markierungen beinhaltet. Weist ein Platz keinerlei derartige Beschriftungen auf – wie beispielsweise die Plätze A bis D, dann sind initial auch keine Markierungen vorhanden. Die klei-

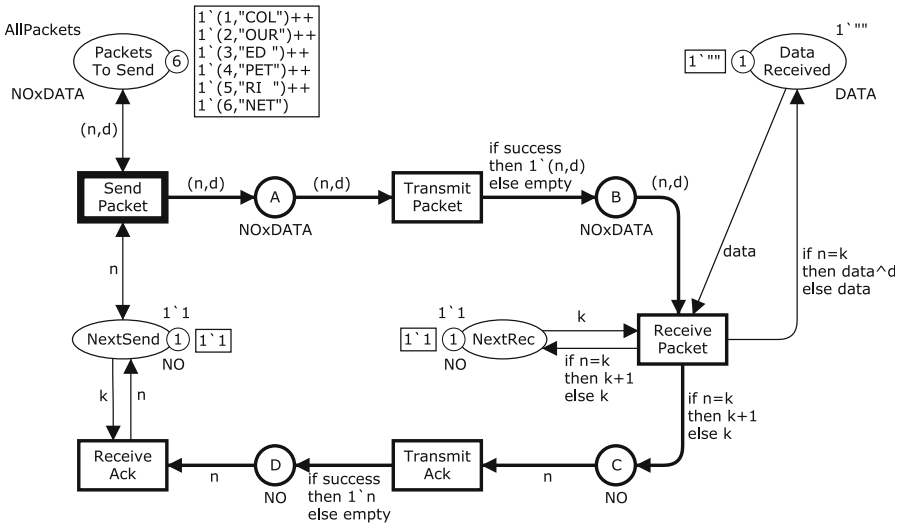



Abbildung 3.2: CPN-Beispielmodell für ein Kommunikationsprotokoll (Quelle: [90])

nen Kreise mit den Zahlen und die danebenstehenden Rechtecke symbolisieren den *aktuellen* Markierungszustand.

*Aktuelle Markierung*

Kantenbeschriftungen definieren, welche Markierungen des Ausgangsplatzes bei welcher Aktivierung einer Transition zum Zielplatz verschoben werden. Diese Entscheidung basiert auf der Farbe der Markierung, welche durch getypte Variablen wie beispielsweise `n` und `d` in Abbildung 3.2 angegeben wird. Die Variablen `n` und `d` sind in Codeausschnitt 3.2 definiert.

```
var n : NO;
var d : DATA;
```

Codeausschnitt 3.2: Getypte Variablen in der CPN ML

Die Kante zwischen `SendPacket` und Platz `A` ist mit `(n, d)` festgelegt, dass `SendPacket` jeweils genau *eine* Markierung vom Typ des Tupels `(NO, DATA)` produziert. Allerdings ist die Anzahl der Markierungen durch die verwendete Kurzschreibweise ausgeblendet worden. Die für andere Kardinalitäten verpflichtende Schreibweise ist hier demzufolge `(1`n, 1`d)`. Somit kann `SendPacket` Markierungen mit Daten wie beispielsweise `1`(7,„Hallo“)` an Platz `A` liefern. Die nachfolgende Kante weist die gleiche Beschriftung auf, was bedeutet, dass Markierungen derselben Menge und desselben Typs für die Aktivierung von `TransmitPacket` notwendig ist. Die Differenzierung zwischen verschiedenen Datentypen und -werten erlaubt somit gegenüber den klassischen Petri-Netzen die Modellierung der Datenorientierten Perspektive. Ein Sonderfall stellt dabei der Zeitstempel dar, den jede Markierung – zusätzlich zum eigentlichen Datenwert – beinhalten kann und der auf Basis eines globalen Modelltaktes während der Ausführung berechnet wird.

*Differenzierung zwischen Datentypen*

Die Ausführungssemantik ist wie bei klassischen Petri-Netzen über die Aktivierung und das Feuern von Transitionen definiert. Hier ist jedoch, wie bereits zuvor erläutert, nicht nur die *Anzahl* der Markierungen in Eingabeplätzen, sondern auch deren *Typ* entscheidend. Basierend auf der initialen Markierung von `NextSend` wird `n` an den Wert `1` gebunden. `PacketsToSend` verfügt zwar

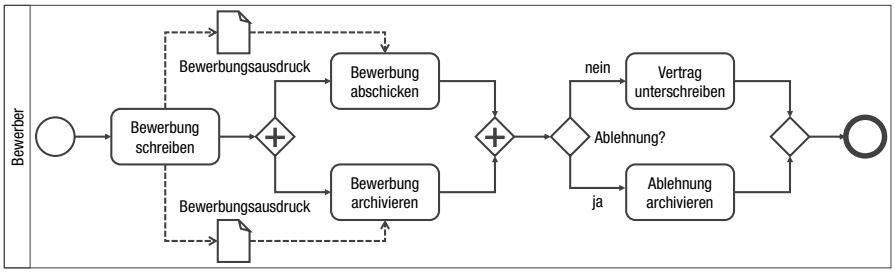



Abbildung 3.3: BPMN-Diagramm eines vereinfachten Bewerbungsprozesses

über sechs Markierungen, kann aber an d nur den Wert „COL“ binden, da n durch NextSend bedingt nur auf 1 gesetzt werden kann. Durch das Feuern der so aktivierten Transition SendPacket wird gemäß Petri-Netz-Semantik die Markierung mit dem Wert 1 von NextSend und die Markierung mit dem Wert (1,„COL“) aus PacketsToSend entfernt. Durch die bidirektionale, gerichtete Kante zwischen letzterem und SendPacket wird dieselbe Markierung aber auch wieder an PacketsToSend verschoben. Gleiches gilt für die Beziehung zwischen SendPacket und NextSend. Das bedeutet, dass sich in dem Bereich des Modells der Markierungszustand nicht verändert. Jedoch ist nun durch die Verbindung mit Platz A auch dieser mit der Markierung (1,„COL“) versehen, was wiederum die Transition TransmitPacket aktiviert und letztlich feuern lässt. Die darauf folgende Kante enthält eine zusätzliche CPN-ML-Aktivierungsbedingung in Form eines Booleschen Ausdrucks. Die Markierung wird hier nur an Platz B weitergeleitet, wenn die Boolesche Variable success auf true gesetzt ist. Dies geschieht jedoch nicht über das Modell, sondern muss durch externe Parametrierung zum Ausführungs- oder Simulationszeitpunkt dynamisch festgelegt werden. Modellierung, Ausführung und Simulation können mit dem etablierten Werkzeug *CPN Tools* [193] durchgeführt werden.

### 3.1.3 *Business Process Model and Notation: BPMN*

*BPMN*

Die imperative Prozessmodellierungssprache *BPMN* ist ein Standard der *Object Management Group (OMG)* und befindet sich zum Zeitpunkt der Anfertigung der vorliegenden Arbeit in der Version 2.0.2 [142]. Eigentlich umfasst der Standard mehrere grafische Notationen, welche jeweils einen unterschiedlichen Fokus haben. In der für die vorliegende Arbeit relevanten Literatur wird fast ausschließlich die Notation für *Prozessdiagramme* betrachtet. Diese beschreibt den Prozess innerhalb einer Organisation, fokussiert den Ablauf der Aktivitäten und blendet beispielsweise die Kommunikation mit anderen Organisationen aus. Die wesentlichen Notationselemente, ein Einblick in die Ausdrucksmächtigkeit der Prozessdiagramme und ihre Ausführungssemantik werden nachfolgend auszugsweise beschrieben. Eine vollständige Betrachtung ist hier nicht sinnvoll, da BPMN eine Vielzahl an Konstrukten liefert, die nur in speziellen Anwendungsfällen Verwendung finden. Im Folgenden werden die Begriffe „BPMN-Standard“ und Prozessdiagramm-Sprache zur Vereinfachung synonym verwendet.

BPMN kann einerseits als domänenspezifische Abstraktionsschicht für Petri-Netze und andererseits als Erweiterung derselben aufgefasst werden. Die wesentlichen Elemente werden anhand von Abbildung 3.3 beschrieben, welche eine BPMN-Repräsentation des bereits in Abbildung 3.1 als Petri-Netz dargestellten vereinfachten Bewerbungsprozesses zeigt. Der Beginn des Prozesses wird mittels eines oder mehrerer *Start Events* modelliert. Das Ende eines Prozesses wird durch ein oder mehrere *End Events* dargestellt. Das Kontrollflusskonzept der Petri-Netze wird zum Teil übernommen. Transitionen werden hier *Tasks* genannt und beschreiben atomare Aktivitäten. Plätze werden hingegen nicht explizit modelliert. Die Aufgaben werden mittels gerichteter Kanten verbunden, die dann einen gerichteten Graphen formen. Verzweigungen im Kontrollfluss wird, wie in Abbildung 3.3 dargestellt, mit sogenannten *Gateways* modelliert. Im Beispiel drückt die Verzweigung nach der ersten Aktivität eine Parallelisierung (*Parallel Gateway*) aus. Die zweite Verzweigung fordert hingegen die exklusive Wahl zwischen zwei möglichen Fortsetzungen des Prozessverlaufs (*Exclusive Gateway*). Zu jeder dieser Verzweigungen gehört ein *Join*-Element gleichen Typs, hier also ein *Parallel Join* und ein *Exclusive Join*. BPMN beinhaltet weitere Gateway-Typen, wie beispielsweise *Inclusive*, *Complex* oder *Event-Based* Gateways. Zusätzlich existieren verschiedene Subtypen für Tasks, beispielsweise *Send Tasks* für das Versenden einer Nachricht oder *Service Tasks* für Aufgaben, die beispielsweise durch den Aufruf eines Webservices bewältigt werden können. Im Beispiel wird lediglich der generische Task-Typ verwendet. Start und End Events sind nur zwei aus einer Vielzahl verschiedener Ereignis-Typen, wobei einige auch innerhalb des Prozessverlaufs auftreten können. Datenobjekte (*Data Objects*) können ebenfalls modelliert werden und beschreiben Dateneingaben und -ausgaben von Aktivitäten. Mit *Pools* und *(Swim-)Lanes* lassen sich Teile des Prozesses verschiedenen organisatorischen Einheiten zuordnen. Ein Pool kann dabei entweder mehrere Lanes enthalten oder selbst als Lane fungieren. Letzteres ist im Beispielprozess dargestellt. Die Unterteilung in verschiedene Lanes erlaubt eine Unterteilung einer größeren in mehrere kleine Organisationseinheiten.

*Start und End Events*

*Tasks*

*Gateways*

*Data Objects*

*Pools und Swimlanes*

Die Ausführungssemantik von Prozessmodellen basiert im Wesentlichen auf zwei verschiedenen Konzepten – den Petri-Netzen und einem generischen *Expression*-Mechanismus. Für den Kontrollfluss wird auf die markierungsbasierte Ausführung der Petri-Netze zurückgegriffen. Ein Start Event generiert dabei genaue eine Markierung. Wenn das sich anschließende Kontrollflusselement ein Task ist, wird die Aufgabe ausgeführt und danach wird die Markierung wieder freigegeben und geht zum nächsten Element über. Ist dieses Element kein Task, sondern ein Gateway, muss zwischen verschiedenen Fällen unterschieden werden. Jedes Join-Element *konsumiert* für jede eingehende Kante eine Markierung. Jedes Join-Element gehört zu einem entsprechenden Verzweigungselement. Eine Verzweigung, die mit einem Parallel Gateway beschrieben ist, *produziert* für jede ausgehende Kante exakt eine Markierung. Ein Exclusive Gateway hingegen produziert nur eine einzige Markierung und reicht diese an eine der ausgehenden Kanten. Welche das ist, kann entweder informal mittels einer Expression oder formal mittels einer sogenannten *Formal Expression* beschrieben werden. Ersteres wird in Abbildung 3.3 verwendet. Für die Verwendung der Formal Expression muss eine beliebige Spra-

*Petri-Netz- und Expression-Semantik*

che gewählt werden, die vom Ausführungssystem unterstützt wird. Ein wichtiger Unterschied zu Petri-Netzen und auch zu den CPNs ist, dass Markierungen in BPMN keine Daten repräsentieren, sondern *ausschließlich* die Aktivierungsreihenfolge von Aktivitäten steuern. Weiterhin ist anzumerken, dass BPMN seine Ausführungssemantik nicht vollständig definiert, sondern diese an einigen Stellen durch Erweiterungsmechanismen offen lässt. Ein Beispiel hierfür ist die eben erwähnte Verwendung von Formal Expressions für Gateways. Auch für Datenobjekte ist eine Definition der Ausführungssemantik gegeben. Dabei wird zwischen Ein- und Ausgabedaten unterschieden. Der Bewerbungsausdruck in Abbildung 3.3 ist Ausgabedatum des Tasks *Bewerbung schreiben* und Eingabedatum der beiden Tasks *Bewerbung abschicken* und *Bewerbung archivieren*. Für jede Aufgabe wird zunächst nacheinander geprüft, ob alle Eingabedaten verfügbar sind. Ist das der Fall, dann wird die Aktivität gestartet – vorausgesetzt, auch der Kontrollfluss hat bereits diese Aktivität erreicht.

*„Ausführungssemantik" für Daten*

Für die Bewertung der Abdeckung der Prozessperspektiven (Abschnitt 1.1.2) muss demnach strikt zwischen den Ebenen der konzeptuellen und der Ausführungssemantik unterschieden werden. Der Kontrollfluss, also die Verhaltensorientierte Perspektive, wird auf beiden Ebenen vollständig beschrieben. Für jedes grafische Element zur Modellierung des Kontrollflusses existiert eine eindeutige Beschreibung der Ausführungssemantik. Das gilt für die operationale Perspektive nur teilweise. Mit beispielsweise Service Tasks können Aufgaben automatisiert werden, was sowohl konzeptuell als auch aus Sicht der Ausführungssemantik modelliert werden kann. Andere Aspekte der Perspektive, wie beispielsweise das für das Bewältigen einer Aufgabe zu verwendende Software-Werkzeug, können lediglich als Freitext an Aufgaben angefügt werden. Eine Ausführungssemantik kann hier nicht hinterlegt werden. Gleiches gilt auch für die Organisatorische Perspektive. Die Modellierung von Pools und Lanes bieten eine Möglichkeit der visuellen Strukturierung, verfügen aber über keinerlei Ausführungssemantik. Stattdessen wird für die „echte" Zuweisung von Ressourcen zu Aktivitäten auf den Expression-Mechanismus verwiesen. Somit beschreibt BPMN die Organisatorische Perspektive nur unvollständig. Die Datenorientierte Perspektive wird hingegen auf beiden Ebenen beschrieben.

*Verhaltensorientierte Perspektive*

*Operationale Perspektive*

*Organisatorische Perspektive*

*Datenorientierte Perspektive*

Diese eingeschränkte Beschreibung der BPMN-Notation verdeutlicht einerseits den signifikanten Unterschied zu Petri-Netzen hinsichtlich der konkreteren Ausrichtung auf die Prozessdomäne und andererseits die Komplexität der Spezifikation selbst. Die BPMN ist einer der Notationen, die für die Evaluation (Abschnitt 8.4.2) des in dieser Arbeit vorgestellten Translationsansatzes (Abschnitt 5.2) für Prozessmodelle verwendet werden.

### 3.1.4 *Declare*

*Declare* [143] ist die in der einschlägigen Literatur am häufigsten anzutreffende deklarative Prozessmodellierungssprache. Diese umfasst im Wesentlichen einen Satz von *Regelschablonen*, von denen in Tabelle 3.2 einige Beispiel dargestellt sind. Mit *ConDec* [144] und *DecSerFlow* [1] existieren eine grafische und eine textuelle Syntax zur Beschreibung von Declare-Modellen. Da diese jedoch an

*Regelschablonen*

die Ausdrucksmächtigkeit und Semantik des Declare-Formalismus gebunden sind, beschränken sich dieser Abschnitt auf die Erläuterung von Declare selbst.

| Regelschablone | Semantik in $\text{LTL}_\text{f}$ |
|---|---|
| $\texttt{existence}(A)$ | $\top \rightarrow \mathbf{F}(e(A) \wedge \varphi_a(\vec{x})) \vee \mathbf{O}(e(A) \wedge \varphi_a(\vec{x}))$ |
| $\texttt{once}(A)$ | $\texttt{existence}(A) \wedge \mathbf{G}((A \wedge \varphi_a(\vec{x})) \rightarrow \neg\mathbf{O}(A \wedge \varphi_c(\vec{x}, \vec{y}) \wedge \varphi_t(\vec{y})))$ |
| $\texttt{init}(A)$ | $A \wedge \varphi_a(\vec{x})$ |
| $\texttt{responded existence}(A, B)$ | $\mathbf{G}((A \wedge \varphi_a(\vec{x})) \rightarrow (\mathbf{O}(B \wedge \varphi_c(\vec{x}, \vec{y}) \wedge \varphi_t(\vec{y})) \vee \mathbf{F}(B \wedge \varphi_c(\vec{x}, \vec{y}) \wedge \varphi_t(\vec{y}))))$ |
| $\texttt{co-existence}(A, B)$ | $\texttt{existence}(A) \leftrightarrow \texttt{existence}(B)$ |
| $\texttt{response}(A, B)$ | $\mathbf{G}((A \wedge \varphi_a(\vec{x})) \rightarrow \mathbf{F}(B \wedge \varphi_c(\vec{x}, \vec{y}) \wedge \varphi_t(\vec{y})))$ |
| $\texttt{alternate response}(A, B)$ | $\mathbf{G}((A \wedge \varphi_a(\vec{x})) \rightarrow \mathbf{X}(\neg(A \wedge \varphi_a(\vec{x}))\mathbf{U}(B \wedge \varphi_c(\vec{x}, \vec{y}) \wedge \varphi_t(\vec{y})))$ |
| $\texttt{chain response}(A, B)$ | $\mathbf{G}((A \wedge \varphi_a(\vec{x})) \rightarrow \mathbf{X}(B \wedge \varphi_c(\vec{x}, \vec{y}) \wedge \varphi_t(\vec{y})))$ |
| $\texttt{precedence}(A, B)$ | $\mathbf{G}((B \wedge \varphi_a(\vec{x})) \rightarrow \mathbf{O}(A \wedge \varphi_c(\vec{x}, \vec{y}) \wedge \varphi_t(\vec{y})))$ |
| $\texttt{alternate precedence}(A, B)$ | $\mathbf{G}((B \wedge \varphi_a(\vec{x})) \rightarrow \mathbf{Y}(\neg(B \wedge \varphi_a(\vec{x}))\mathbf{S}(A \wedge \varphi_c(\vec{x}, \vec{y}) \wedge \varphi_t(\vec{y})))$ |
| $\texttt{chain precedence}(A, B)$ | $\mathbf{G}((B \wedge \varphi_a(\vec{x})) \rightarrow \mathbf{Y}(A \wedge \varphi_c(\vec{x}, \vec{y}) \wedge \varphi_t(\vec{y})))$ |
| $\texttt{succession}(A, B)$ | $\texttt{precedence}(A, B) \wedge \texttt{response}(A, B)$ |
| $\texttt{chain succession}(A, B)$ | $\texttt{chain precedence}(A, B) \wedge \texttt{chain response}(A, B)$ |
| $\texttt{not responded existence}(A, B)$ | $\mathbf{G}((A \wedge \varphi_a(\vec{x})) \rightarrow \neg(\mathbf{O}(B \wedge \varphi_c(\vec{x}, \vec{y}) \wedge \varphi_t(\vec{y})) \vee \mathbf{F}(B \wedge \varphi_c(\vec{x}, \vec{y}) \wedge \varphi_t(\vec{y}))))$ |
| $\texttt{not response}(A, B)$ | $\mathbf{G}((A \wedge \varphi_a(\vec{x})) \rightarrow \neg\mathbf{F}(B \wedge \varphi_c(\vec{x}, \vec{y}) \wedge \varphi_t(\vec{y})))$ |
| $\texttt{not precedence}(A, B)$ | $\mathbf{G}((B \wedge \varphi_a(\vec{x})) \rightarrow \neg\mathbf{O}(A \wedge \varphi_c(\vec{x}, \vec{y}) \wedge \varphi_t(\vec{y})))$ |
| $\texttt{not succession}(A, B)$ | $\texttt{not precedence}(A, B) \wedge \texttt{not response}(A, B)$ |
| $\texttt{not chain response}(A, B)$ | $\mathbf{G}((A \wedge \varphi_a(\vec{x})) \rightarrow \neg\mathbf{X}(B \wedge \varphi_c(\vec{x}, \vec{y}) \wedge \varphi_t(\vec{y})))$ |
| $\texttt{not chain precedence}(A, B)$ | $\mathbf{G}((B \wedge \varphi_a(\vec{x})) \rightarrow \neg\mathbf{Y}(A \wedge \varphi_c(\vec{x}, \vec{y}) \wedge \varphi_t(\vec{y})))$ |
| $\texttt{not chain succession}(A, B)$ | $\texttt{not chain precedence}(A, B) \wedge \texttt{not chain response}(A, B)$ |

Tabelle 3.2: Beispiele für Declare-Regelschablonen

In Declare ist es möglich, Aktivitäten zu modellieren und diese mittels Instanziierung der angebotenen Regelschablonen zu verknüpfen. Diese Verknüpfungen haben allerdings die Semantik von *Einschränkungen* möglicher Prozessverläufe, anstatt, wie bei imperativen Sprachen, die Semantik *erlaubter* Prozessverläufe. Die Regelschablonen lassen sich im Wesentlichen in zwei Gruppen einteilen: *(i)* Existenzbezogene und *(ii)* Reihenfolgebezogene Schablonen. Die Schablone $\texttt{precedence}(A, B)$ ist ein Beispiel für zweiteres und legt fest, dass irgendwann vor der Ausführung von Aktivität $B$ Aktivität $A$ ausgeführt werden muss. Dagegen besagt beispielsweise $\texttt{co-existence}(A, B)$, dass, falls eine der beiden Aktivitäten $A$ respektive $B$ ausgeführt wird, auch die jeweils andere ausgeführt werden muss, bevor der Prozess endet. Diese ursprüngliche Form von Declare erlaubt lediglich die Spezifikation von Kontrollflussabhängigkeiten und unterstützt somit lediglich die Funktionale und die Verhaltensorientierte Perspektive. Eine spätere Erweiterung [35], im Folgenden *MP-Declare* genannt, führt die Möglichkeit ein, zusätzlich Datenabhängigkeiten zu formulieren. Ein Beispiel in natürlichsprachlicher Form ist: „Wenn die Kosten höher als 1000 sind, muss vor der Ausführung von Aktivität *B* irgendwann *A* ausgeführt worden sein." Damit ist es auch möglich, *perspektivenübergreifende* Regelschablonen zu definieren, was in keiner der vorab beschriebenen imperativen Prozessmodellierungssprachen möglich ist.

*Ausführungssemantik*

Die Ausführungssemantik von Declare ist durch eine statische Abbildung der Regelschablonen auf geeignete Logikkalküle definiert. Im Fall der ursprünglichen Declare-Variante ohne Datenperspektive handelt es sich dabei meist um die *Lineare Temporale Logik (LTL)* und für die Modellierung von Datenabhängigkeiten wird

*LTL*

*MFOTL*

auf die sogenannte *Metric First Order Temporal Logic (MFOTL)* zurückgegriffen. Declare ist damit zunächst auf eine statische Grundmenge an Regelschablonen beschränkt, kann aber durch die Definition neuer Schablonen und unter Angabe der Logik-Entsprechung erweitert werden. Die LTL-Entsprechung der oben genannten $\mathtt{precedence}(A, B)$-Regel lautet beispielsweise $\mathbf{G}\,(B \rightarrow \mathbf{O}A)$ (Tabelle 3.2), wobei der Sonderoperator $\mathbf{G}$ die Gültigkeitsbedingung für die Regel beinhaltet. Der $\mathbf{G}$-Operator legt fest, dass der Klammer-Ausdruck zu jedem Zeitpunkt Gültigkeit haben muss. Der $\mathbf{O}$-Operator sagt, dass das davon rechter Hand stehende Symbol zu einem früheren Zeitpunkt mindestens einmal aufgetreten sein muss. Zusammen mit dem Implikationsoperator ist die Regel von links nach rechts wie folgt zu lesen: Es muss zu jedem Zeitpunkt gelten, dass, wenn $B$ ausgeführt werden soll, zu einem früheren Zeitpunkt mindestens einmal $A$ ausgeführt worden sein. Die drei verbleibenden Operatoren $\mathbf{Y}$, $\mathbf{X}$ und $\mathbf{F}$ schränken die Gültigkeit der eingeschlossenen Bedingung – der Reihe nach – auf den unmittelbar vorherigen, den unmittelbar folgenden oder einen beliebigen zukünftigen Zustand ein. Jede Logikformel der beispielhaft aufgeführten Declare-Regelschablonen (Tabelle 3.2) setzt sich aus einem solche Grundgerüst (schwarz) zusammen. Da dieses jedoch nur die Funktionale und die Verhaltensorientierte Perspektive Unterstützt, wird es mittels MFOTL-Elementen um Zusatzbedingungen (graublau) erweitert. Beispielsweise ist die MFOTL-Regelschablone $\mathbf{G}((B \wedge \varphi_a(\vec{x})) \rightarrow \mathbf{O}(A \wedge \varphi_c(\vec{x}, \vec{y}) \wedge \varphi_t(\vec{y})))$ eine Erweiterung der $\mathtt{precedence}(A, B)$-LTL-Regel. Im Wesentlichen wird letztere zu beiden Seiten des Implikationsoperators um Datenbedingungen erweitert.

*Aktivierungs-, Korrelations- und Zielbedingungen*

Die zusätzliche Datenbedingung $\varphi_a(\vec{x})$ schränkt die Aktivierung der Regel auf die Fälle ein, in denen $B$ ausgeführt werden soll und die Datenbedingung selbst wahr ist. Diese zusätzliche Einschränkung wird als *Aktivierungsbedingung* bezeichnet. Der rechte Teil der Implikation wird um $\varphi_c(\vec{x}, \vec{y})$ erweitert und legt somit fest, dass nicht nur $A$ zu einem früheren Zeitpunkt auftreten muss, sondern dass zu diesem Zeitpunkt eine bestimmte Beziehung zwischen den Datenwerten $\vec{x}$ und $\vec{y}$ bestehen muss. Als letzte Zusatzbedingung schränkt $\varphi_t(\vec{y})$ den Datenwert zum Zeitpunkt der Ausführung von $A$ ein. Statt der LTL- bzw. MFOTL-Schreibweisen werden nachfolgend die in Tabelle 3.2 angegebenen Regelschablonen verwendet. Die beispielhaften Logikausdrücke in diesem Abschnitt dienen jedoch zur Verdeutlichung des Unterschieds zwischen der prinzipiellen Ausdrucksmächtigkeit der deklarativen Prozessmodellierungssprachen. Diesbezüglich ist Declare also auf Beschreibungen des Kontrollflusses beschränkt, da der zugrundeliegende Logikformalismus keine Datenabhängigkeiten erlaubt. Eine Erweiterung von LTL auf MFOTL hebt diese Beschränkung auf.

Declare wird im Evaluationsteil der Arbeit zur Modellierung einiger Beispielprozessmodelle verwendet (Abschnitt 8.4.2).

### 3.1.5 *Declarative Process Intermediate Language: DPIL*

*DPIL*

Die multi-perspektivische *Declarative Process Intermediate Language (DPIL)* [199, 200] ist eine textuelle, regelbasierte, konzeptuelle Prozessmodellierungssprache,

*Multi-Perspektivität*

welche alle der in Abschnitt 1.1.2 beschriebenen fünf Perspektiven abdeckt. Darüber hinaus zeichnet sich DPIL gegenüber anderen Sprachen besonders durch

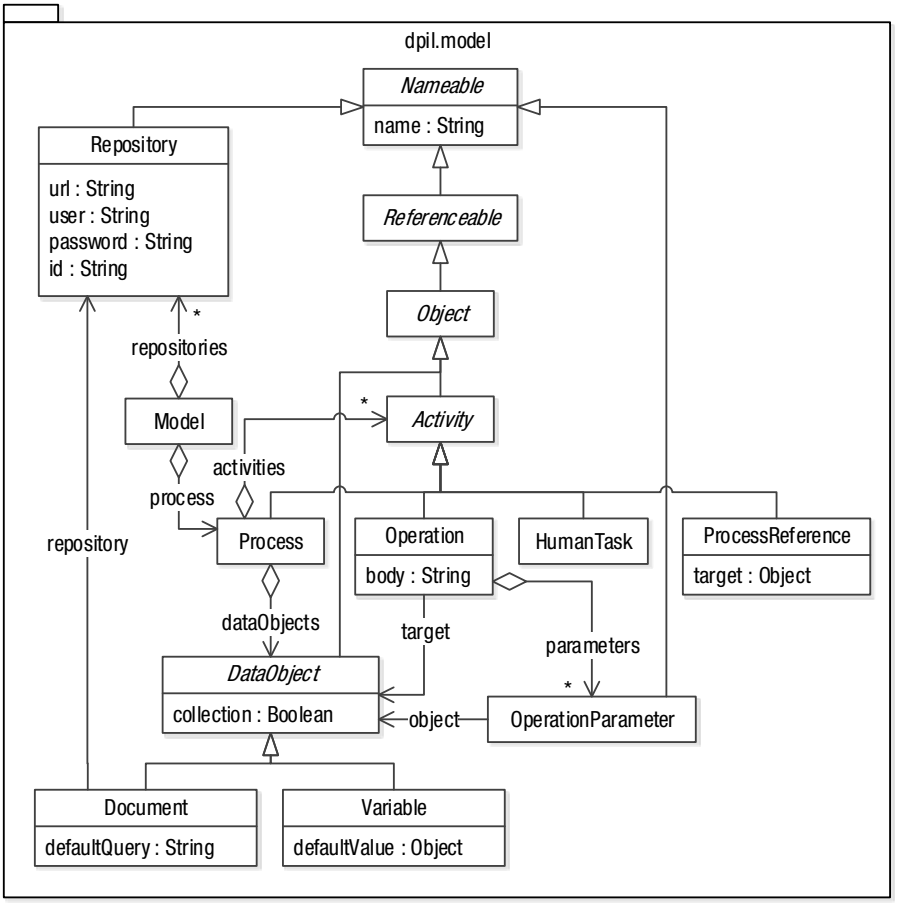



Abbildung 3.4: Abstrakte Syntax der DPIL-Modelle (Quelle: [199])

flexible Möglichkeiten zur Definition der Prozessregeln aus. Ziel dieses Abschnitts ist es daher, einerseits die Sprache selbst und andererseits ihre Alleinstellungsmerkmale überblicksweise zu beschreiben. Dazu wird primär die *abstrakte* Syntax der Sprache betrachtet. Grund ist, dass diese die Grundlage für die Abbildung auf die zugehörige Ausführungssprache und die in Kapitel 4 verwendete Simulationssprache ist. Dagegen beschränkt sich die Beschreibung der *konkreten* Syntax auf wenige Auszüge in einer an die *Erweiterte Backus-Naur-Form*[1] angelehnten Darstellung. Auch von der abstrakten Syntax werden nur die Teile im Detail beschrieben, welche für die vorliegende Arbeit relevant sind. In den letzten beiden Teilen dieses Abschnitts wird die Ausführungssemantik und eine Liste gängiger Regelschablonen beschrieben.

*Modellieren mit DPIL Abstrakte und Konkrete Syntax von DPIL*

Modelle (Abbildung 3.4 und Codeausschnitt 3.3) beinhalten unter anderem die Entitäten der einzelnen Perspektiven. Daher können im Kopf der DPIL-Modelle Definitionen der zu verwendenden organisatorischen Ressourcen (*Identity*) und Ressourcengruppen (*Group*) sowie die Beziehungstypen (*RelationType*) angegeben werden. Letzteres schafft die Möglichkeit, flexibel zwischen verschiedenen Arten der Ressource-Ressource- respektive Ressource-Ressourcengruppe-Beziehungen zu unterscheiden. Weiterhin können Repositorien (*Repository*) angegeben werden,

1 Konkret handelt es sich um die Syntax sogenannter *ANTRL*-Grammatiken (`http://www.antlr2.org/doc/metalang.html`, zuletzt aufgerufen am 28.12.2016)

*Makros als Regelschablonen*

um Dateien und Prozessmodelle aus verschiedenen Quellen beziehen oder in selbigen ablegen zu können. Dazu müssen Verbindungsinformationen für *CMIS-kompatible ECM-Systeme*[2] angegeben werden. Makros (*Macro*) sind ein zentraler Gegenstand für die Lesbarkeit von Prozessmodellen und werden zur Formulierung von Regelschablonen genutzt. Einige Regeln gelten global für alle Prozesse, sodass man sie nicht in einem Prozessmodell, sondern auf höherer Ebene definieren sollte. Dazu können *GlobalRule*s angegeben werden. Schließlich enthalten Modelle auch das eigentliche Prozessmodell (*Process*).

```
Model: Identity* Group* RelationType* Repository* Macro* GlobalRule* Process?;
Identity: 'identity' ID;
Group: 'group' ID;
RelationType: 'relationtype' ID;
Repository: 'repository' ID '{'
'url' STRING
'user' STRING
'password' STRING
'id' STRING '}';
```

Codeausschnitt 3.3: Oberste Ebene der DPIL-Grammatik

*DPIL-Modellstruktur*

Ein Prozessmodell setzt sich aus Aktivitäten (*Activity*), Datenobjekten (*DataObject*) mit den Ausprägungen *Variable* und *Document* sowie den Prozessregeln (*ProcessRule*) zusammen (Abbildung 3.4, Abbildung 3.5 und Codeausschnitt 3.4). Aktivitäten können dabei einerseits eigene Prozesse sein, wodurch hierarchische Modelle formuliert werden können. Andererseits kann es sich auch um eine von menschlichen Beteiligten zu erledigende Aufgabe (*Task*) oder einen automatisierten Arbeitsschritt (*Operation*) handeln. Letztere können parametriert werden (*Parameter*) und beispielsweise Webservice-Aufrufe kapseln. Variablen beinhalten Werte, die nur innerhalb einer Prozessinstanz Gültigkeit haben, Dokumente repräsentieren Werte, die auch nach Beendigung der Instanz noch verfügbar sind. Dokumente werden in den im Modellkopf definierten Repositorien abgelegt. Beide können mehrwertig, also Kollektionen von Variablen oder Dokumenten sein.

```
Process: 'process' ID STRING? '{'
    Activity*
    DataObject*
    ProcessRule* '}';
Activity: Process | Task | Operation;
Task: 'task' ID STRING?;
Operation: 'operation' ID STRING
    ('{' Parameter (',' Parameter)* '}')?
    ('to' ID);
Parameter: ID ID;
DataObject: Variable | Document;
Variable: 'variable' 'collection'? ID STRING?;
Document: 'document' 'collection'? ID STRING?
    ('default' STRING)? 'at' ID;
```

Codeausschnitt 3.4: Prozessmodellebene der DPIL-Grammatik

*DPIL-Regelstruktur*

Prozessregeln bilden den Kern des Prozessmodells. Ihre Bestandteile sind in Abbildung 3.5 und Codeausschnitt 3.5 dargestellt. Die zugehörige Ausführungssemantik wird detailliert im späteren Verlauf dieses Abschnitts betrachtet. Prozessregeln müssen entweder obligatorisch befolgt werden (*ensure*), können lediglich

*Zwei Modalitäten*

2 CMIS = Content Management Interoperability Services, ECM = Enterprise Content Management

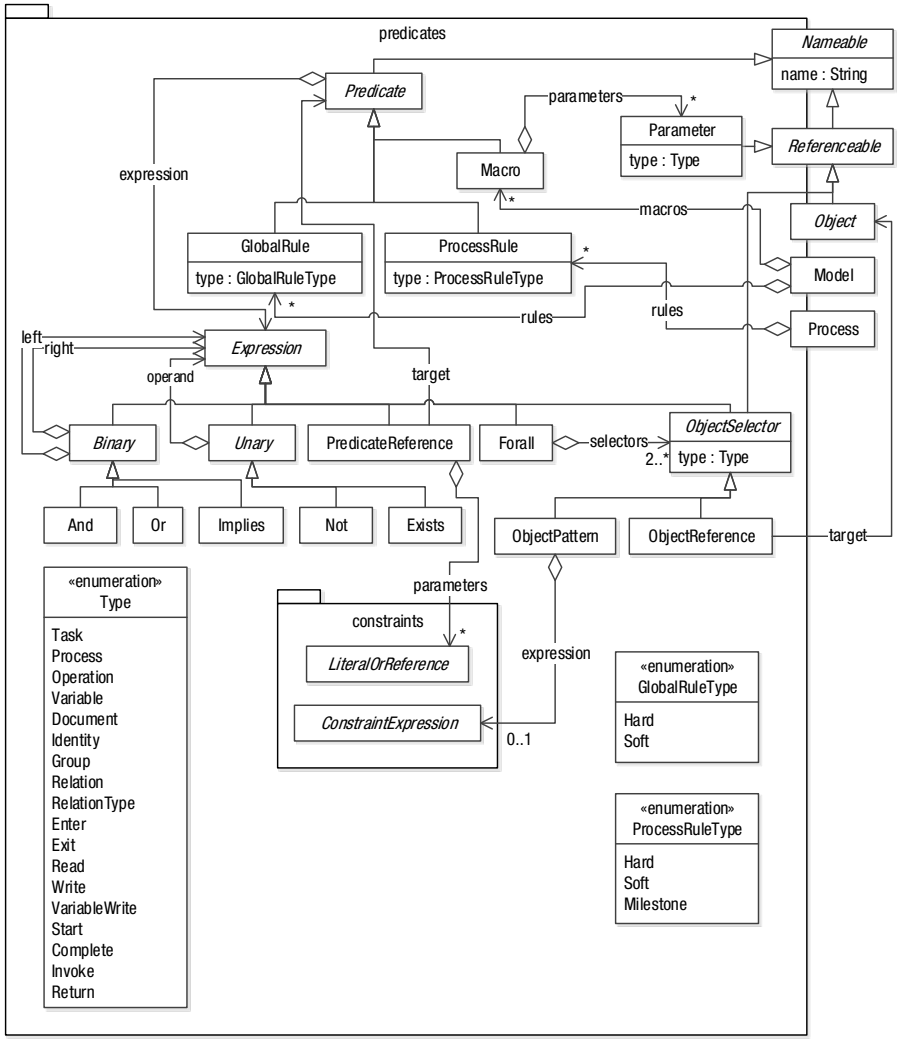



Abbildung 3.5: Abstrakte Syntax der DPIL-Regeln (Quelle: [199])

Empfehlungen beinhalten (*advise*) oder Prozessziele formulieren (*milestone*).

Die Regel selbst wird als Kombination der unterschiedlichen Ausprägungen eines Ausdrucks (*Expression*) formuliert, wie es in Abbildung 3.5 und Codeausschnitt 3.6 dargestellt ist. Ausdrücke bilden in DPIL eine Baumstruktur, wobei jeder Knoten entweder ein unärer oder ein binärer Ausdruck ist. Unäre Ausdrücke setzen sich aus einem Existenz- oder Negationsoperator (*Exists* bzw. *Not*) und einem weiteren beliebigen Ausdruck zusammen. Binäre Ausdrücke setzen sich aus einem Konjunktions-, Disjunktions- oder Implikationsoperator (*And*, *Or* bzw. *Implies*) und zwei weiteren beliebigen Ausdrücken zusammen. Die zugehörige Grammatik setzt die Operatorpräzedenz um und priorisiert so beispielsweise Negations- und Klammeroperatoren höher als den Implikationsoperator. Zudem wird eine Infix-Schreibweise binärer Ausdrücke vorgegeben, sodass der binäre Operator immer zwischen seinen beiden Argumenten steht.

*Verschachtelte Ausdrücke als Regel-Grundstruktur*

Der Universalquantor (*ForAll*) legt auf Basis der Objekte, die mit dem ersten Muster (*ObjectSelector*) erfasst werden alle weiteren Objekte fest. Ausdrücke können auch eine Referenz auf ein Prädikat (*PredicateReference*), also eine Referenz

```
ProcessRule: ProcessRuleType (ID? STRING? ':')? Expression;
ProcessRuleType: 'ensure' | 'advise' | 'milestone';
Macro: ID ('(' ID (',' ID)* ')')? 'iff' Expression;
```

Codeausschnitt 3.5: DPIL-Grammatik für Prozessregeln

auf ein Makro oder eine andere Regel beinhalten. Makros beinhalten Platzhalter – sogenannte Formalparameter – für die jeweils Aktualparameter angegeben werden müssen. Diese können in der Prädikatreferenz nach dem Regel- oder Makronamen spezifiziert werden. Muster können Objekte entweder mittels ihres eindeutigen Bezeichners (*ObjectReference*) oder anhand anderer Eigenschaften (*ObjectConstraint*) auswählen. Die Ergebnisse der musterbasierten Selektion kann zur späteren Weiterverwendung an Variablen gebunden werden. Ein ObjectConstraint setzt sich wieder aus unären respektive binären Ausdrücken zusammen, die den Vergleich der gewünschten Eigenschaften auf Basis von Musterwerten ermöglichen.

```
Expression: Implies;
Implies: Or ('implies' Or)*;
Or: And ('or' And)*;
And: Unary ('and' Unary)*;
Unary: '(' Expression ')'
| 'not' Unary
| 'exists' Unary
| Forall
| PredicateReference
| ObjectSelector;
Forall: 'forall' '(' ObjectSelector ObjectSelector+ ')';
PredicateReference: ID ('(' LiteralOrReference (',' LiteralOrReference)* ')')?;
ObjectSelector: ObjectConstraint | ObjectReference;
ObjectReference: Type ID (':' ID)?;
ObjectConstraint: Type ('(' ConstraintExpression ')')?
(':' ID)?;
Type: 'task' | 'operation' | 'process' | 'start' | ...;
```

Codeausschnitt 3.6: DPIL-Grammatik für Ausdrücke

*Organisatorisches Metamodell*

Die organisatorischen Modellanteile müssen gegenüber des in Abbildung 3.6 dargestellten und in [38] erstmals vorgestellten Metamodells konform sein. Der untere Teil der Abbildung zeigt eine mögliche Implementierung des Metamodells als *LDAP*-Netzwerkprotokoll.

*Ausführung von DPIL-Modellen DRL*

*Lebenszyklus*

Die *Ausführung* von DPIL-Modellen wird mittels einer Abbildung auf eine prädikatenlogikbasierte Ausführungssprache, die *Drools Rule Language (DRL)* [177] realisiert. Ein wichtiger Aspekt für die Ausführung ist, dass bestimmte Modellelemente innerhalb einer Prozessinstanz einem *Lebenszyklus* folgen. Daten können so beispielsweise geschrieben (*Write*) oder gelesen (*Read*) werden. Aufgaben für menschliche Beteiligte können sich in den Zuständen gestartet (*Start*) und abgeschlossen (*Complete*) befinden. Das Erreichen jedes dieser einzelnen Zustände wird als Ereignis behandelt. Die Prozessausführung erfolgt also ereignisbasiert und unterscheidet, im Kontext von DPIL, zwischen jeweils zwei Ereignistypen für Datenzugriffe und Aufgaben mit menschlicher Beteiligung. Der aktuelle Zustand des Prozesses setzt sich aus der Kette aller vorangegangenen Ereignisse zusammen. Basierend auf diesem Zustand werden alle möglichen Folgeereignisse für jedes Modellelement generiert und hinsichtlich ihrer Konformität gegenüber den

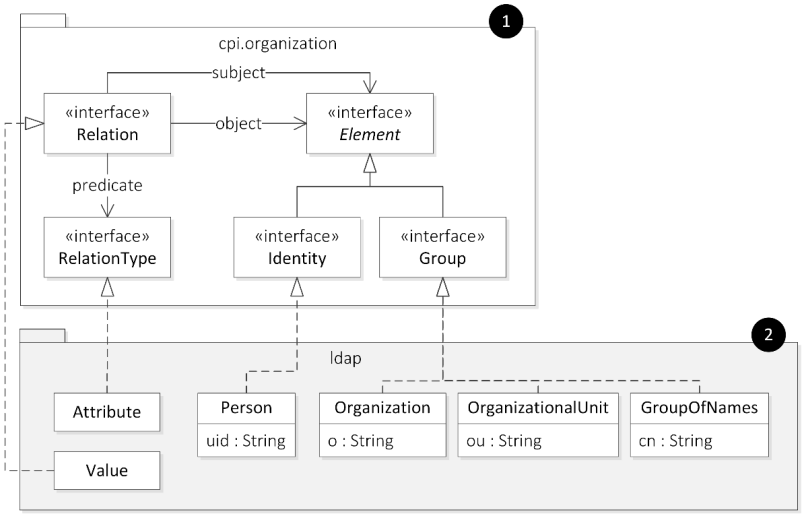



Abbildung 3.6: Organisatorisches Metamodell (Quelle: [199])

Regeln des DPIL-Modells geprüft. Verletzt ein Ereignis eine obligatorisch zu befolgende Regel (ensure), dann wird es verworfen. Bei Verletzung einer Empfehlung (advise) wird das Ereignis als nicht empfohlen gekennzeichnet, wird aber *nicht* verworfen. Jedes der übrig gebliebenen Ereignisse wird schließlich dem Nutzer als mögliche Prozessfortsetzung angeboten.

*Unterscheidung in verbotene, nicht empfohlene, neutrale und empfohlene Ereignisse*

Für die Überprüfung der Konformität der Ereignisse mit den Prozessregeln eines DPIL-Modells $M$ wird jedes mögliche zukünftige Ereignis mit der Historie vergangener Ereignisse kombiniert. Nun muss das Ausführungssystem prüfen, ob das jeweilige neue Ereignis verboten, nicht empfohlen, neutral oder empfohlen ist. DPIL-Modelle bestehen aus einer Menge von Entitäten $E$ – beispielsweise Aktivitäten – und Prozessregeln $R$, sodass gilt $M = (E, R)$. Eine Regel $r \in R$ ist eine logische Aussage, die entweder erfüllt oder verletzt sein kann. Die Menge der Regeln unterscheidet sich in obligatorisch zu befolgende $R_H$ Regeln, Empfehlungen $R_S$ und Meilensteine $R_M$, sodass gilt $R = R_H \cup R_S \cup R_M$. Eine Prozessinstanz $I$ ist eine Kette von Ereignissen innerhalb eines Prozesses. Sind alle Meilensteine $R_M$ erfüllt, wird der Prozess als beendet angesehen. Ist kein Meilenstein definiert, wird der Prozess niemals explizit abgeschlossen, sodass ein entsprechender manueller Abbruch notwendig ist. Zur Vereinfachung der nachfolgenden Erläuterungen wird nachfolgend mit $\delta(M, N) = (M \backslash N, N \backslash M)$ die Änderung einer Menge $M$ zu einer Menge $N$ bezeichnet. Die Änderung der Menge $\{1, 2\}$ zu einer Menge $\{2, 3\}$ ist folglich $\delta(\{1, 2\}, \{2, 3\}) = (\{1\}, \{3\})$.

*Verbotene und erlaubte Ereignisse*

Die Auswertung $p$ eines Ereignisses $e$ ist die Menge der durch dieses Ereignis verletzten Regeln, sodass gilt $p \subset R$. Die Ereignisauswertung $p(I \cup e)$ ist demnach die Auswertung der Instanz $I$ nach dem Auftreten des Ereignisses $e$. Das Ereignis ist *verboten*, wenn die Auswertung Regeln aus der Menge der obligatorisch zu befolgenden Regeln enthält. Andernfalls ist das Ereignis somit *erlaubt*:

$$\begin{aligned} \mathbf{verboten}(e) &\leftarrow p(I \cup e) \cap R_H \neq \emptyset \\ \mathbf{erlaubt}(e) &\leftarrow p(I \cup e) \cap R_H = \emptyset \end{aligned} \tag{1}$$

*Abschließende Ereignisse*

Weiterhin wird ein Ereignis als *abschließend* angesehen, wenn die Auswertung keine Überdeckungen mit der Menge der Meilensteine aufweist. Im Umkehrschluss bedeutet dies, dass das Auftreten des Ereignisses dafür sorgt, dass danach *alle* Meilensteine erfüllt sind:

$$\mathtt{abschließend}(e) \leftarrow R_M \neq \emptyset \wedge p(I \cup e) \cap R_M = \emptyset \tag{2}$$

Die Änderung der Instanzauswertung $\Delta p_I$ ist der Unterschied zwischen der Auswertung der Instanz $I$ *ohne* das Auftreten des Ereignisses $e$ und der Auswertung *mit* diesem Ereignis:

$$\Delta p_I(e) = \delta(p(I), p(I \cup e)) \tag{3}$$

*Neutrale Ereignisse*

Damit ist nun eine differenziertere Unterscheidung der *erlaubten* Ereignisse möglich. Ein Ereignis gilt als *neutral*, wenn es keine Änderung der Instanzauswertung verursacht:

$$\mathtt{neutral}(e) \leftarrow \Delta p_I(e) = (\emptyset, \emptyset) \tag{4}$$

*Empfohlene Ereignisse*

Dagegen gilt ein erlaubtes Ereignis als *empfohlen*, falls es einen Teil der früheren Verletzungen $P$ von Empfehlungen und/oder Meilensteinen aufhebt und keine anderen Regeln durch dieses Ereignis verletzt werden:

$$\mathtt{empfohlen}(e) \leftarrow \Delta p_I(e) = (P, \emptyset) \wedge P \cap (R_S \cup R_M) \neq \emptyset \tag{5}$$

*Nicht empfohlene Ereignisse*

Im Umkehrschluss gelten demnach erlaubte Ereignisse als *nicht empfohlen*, wenn sie neue Regelverletzungen verursachen:

$$\mathtt{nichtempfohlen}(e) \leftarrow \Delta p_I(e) = (P, Q) \wedge Q \cap (R_S \cup R_M) \neq \emptyset \tag{6}$$

Damit ist die Ausführungssemantik der multi-modalen, deklarativen DPIL-Regeln logisch fundiert. Darauf aufbauend kann die Validität der Abbildung von DPIL-Modellen auf ein Simulationsmodell – diskutiert in einem der Kernbeiträge der Arbeit (Abschnitt 4.2) – geprüft werden.

Die kontrollflussorientierte Prozessmodellierungssprache Declare beinhaltet einen vordefinierten Satz an Regelschablonen. Mit den von DPIL unterstützten Makros lassen sich ebenfalls Regelschablonen definieren. In [199] und [161] wird eine Bibliothek an grundlegenden Prozessschablonen identifiziert. Diese Schablonen sind nachfolgend aufgeführt, ihre Semantik wird überblicksartig erläutert[3] und sie dienen als Referenz für die Konzeption der später vorgestellten Technik zur Simulation multi-perspektivischer, deklarativer Prozessmodelle (Abschnitt 4.2). Bereits in den Arbeiten zur Schablonen-Bibliothek selbst wird angemerkt, dass diese keinesfalls vollständig sein kann. Allerdings zeigt sie das breite *Spektrum* möglicher Prozessregeln und ist damit ein geeigneter Startpunkt für DPIL-basierte Werkzeugentwicklungen.

*Verhaltensorientierte DPIL-Makros*

In Codeausschnitt 3.7 sind Regelschablonen zur Formulierung von Einschränkun-

3 Zugunsten der Übersichtlichkeit wird bei der Erklärung der DPIL-Makros davon aus gegangen, dass sie zur Formulierung *verpflichtender* Regeln (`ensure`) verwendet werden. Dieselben Makros können, wie im aktuellen Abschnitt erwähnt, auch zur Formulierung von Empfehlungen (`advise`) verwendet werden.

```
sequence(a,b) iff
    start(of b at :t) implies complete(of a at <t)

directSequence(a,b) iff
    start(of b at :t) implies complete(of a at :s) and not start(at >s and at <t)

response(a,b) iff
    complete(of a at :t) implies complete(of b at >t)

once(a) iff
    start(of a at :t) implies not complete(of a at <t)

serializeAll iff
    start(of :a at :t) and not complete(of a at >t) implies not start(at >t)
```

Codeausschnitt 3.7: DPIL-Regelschablonen: Verhaltensorientierte Perspektive

gen der Verhaltensorientierten Perspektive dargestellt. Das Makro `sequence(a,b)` besagt, dass vor der Bearbeitung von Aufgabe `b` Aufgabe `a` mindestens einmal abgeschlossen worden sein muss. Das darauffolgende Makro – `directSequence(a,b)` – ähnelt der vorangegangenen und formuliert, dass Aufgabe `a` *unmittelbar vor* Beginn von Aufgabe `b` abgeschlossen sein muss. Das Makro `response(a,b)` legt fest, dass nach Abschluss von Aufgabe `a` und vor dem Ende des Prozesses mindestens einmal Aktivität `b` durchgeführt werden muss. Mit `once(a)` wird bestimmt, dass Aufgabe `a` nicht öfter als einmal pro Prozessinstanz ausgeführt werden darf. Die Unterscheidung zwischen verschiedenen Ereignistypen, welche die Zustände der Bearbeitung einer Aufgabe abbilden sollen, erlaubt implizit die parallele Durchführung zweiter oder mehrerer Aktivitäten. Mit dem `serializeAll`-Makro kann jedoch festgelegt werden, dass nach Beginn der Ausführung einer Aktivität diese zunächst beendet werden muss, bevor mit einer anderen Aufgabe begonnen werden kann.

Der multi-perspektivische Charakter der deklarativen Prozessmodellierungssprache DPIL gestattet auch die Formulierung von Regeln und Makros, welche Zusammenhänge mehrerer Prozessperspektiven betreffen. Somit beschränken die in Codeausschnitt 3.8 dargestellten Regelschablonen sowohl die Verhaltensorientierte als auch die Datenorientierte Perspektive sowie beide im Zusammenspiel. Beispielsweise beschränkt das Makro `local(a,o)` den Lesezugriff auf das Datenobjekt `o` auf die Laufzeit der Durchführung von Aktivität `a`. Außerhalb der Bearbeitung dieser Aufgabe sind Lesezugriffe verboten. Die beiden Makros `produces(a,o)` und `consumes(a,o)` beschreiben die Aufgabe `a` als Datenquelle respektive Datensenke für das Datenobjekt `o`. Das Makro `dataSequence(a,b,o,v)` ähnelt dem `sequence`-Makro, enthält aber eine zusätzliche Aktivierungsbedingung. Diese bewirkt, dass die Sequenz-Regel zwischen den Aktivitäten `a` und `b` nur dann gilt, wenn die Variable `o` mit dem Wert `v` belegt ist. Mit `dataOrdering(o1,o2)` wird festgelegt, dass vor der Produktion des Datenobjekts `o2` das Datenobjekt `o1` erzeugt werden muss.

Codeausschnitt 3.9 zeigt die Regelschablonen der Schablonenbibliothek, welche sowohl die Verhaltensorientierte als auch die Organisatorische Perspektive betreffen. Das `direct(a,i)`-Makro legt den Ausführenden für Aktivität `a` auf die

```
local(a,o) iff
    read(of o at :t1) implies start(of a at <tr at :t2) and
        not complete(of a at >t2 at <t1)

produces(a,o) iff
    complete(of a at :t) implies write(of o at <t)

consumes(a,o) iff
    start(of a at :t) implies write(of o at <t)

dataSequence(a,b,o,v) iff
    start(of b at :t) and variablewrite(of o value v at <t)
        implies complete(of a at <t)

dataOrdering(o1,o2) iff
    write(of o2 at :t) implies write(of o1 at <t)
```

Codeausschnitt 3.8: DPIL-Regelschablonen: Verhaltensorientierte mit Datenorientierter Perspektive

organisatorische Ressource `i` fest. Die Regelschablone `role(a,r)` beschreibt eine schwächere Form der Ressourcenbeschränkung, indem für die Durchführung von Aktivität `a` lediglich gefordert wird, dass die zugewiesene Ressource die Rolle `r` bekleidet. Die beiden nachfolgenden Makros beschränken den Ausführenden von Aktivität `b` dynamisch anhand des Ausführenden von Aufgabe `a`. Im Falle von `separate(a,b)` wird festgelegt, dass die beiden Aufgaben nicht von derselben Ressource bearbeitet werden dürfen, beschreibt also eine Trennung von Zuständigkeiten. Damit kann beispielsweise das *Vieraugenprinzip* umgesetzt werden. Das Makro `binding(a,b)` beschreibt im Gegensatz dazu die Verknüpfung von Verantwortlichkeiten, legt also fest, dass die beiden Aktivitäten von derselben organisatorischen Ressource ausgeführt werden müssen. Möchte man diese Einschränkungen für alle im Prozess enthaltenen Aktivitäten treffen, dann müssten dafür nahezu so viele `binding`-Regeln formuliert werden, wie Aktivitäten im Modell existieren. Das Makro `caseHandling` beschreibt dieselbe Einschränkung, reduziert den Aufwand der Regelformulierung jedoch drastisch. Mit `bindingRole(a,b)` wird bestimmt, dass die Aufgaben `a` und `b` von Ressourcen mit derselben Rolle ausgeführt werden müssen. Das Makro `capability(a,rt,g)` beschreibt eine Aktivitätszuweisung auf Basis der Fähigkeiten der Ressourcen. Die geforderten Fähigkeiten werden in der Form eines Prädikats `rt` und eines Objekts `g` beschrieben. Mit dem Makro `orgDistMulti(a,b,rt)` können Einschränkungen bezüglich der Aktivitätszuweisung auf Basis der organisatorischen Beziehungen (`rt`) zwischen einzelnen Ressourcen getroffen werden. Die Regelschablone `roleSequence(a,b,g)` ähnelt dem `sequence`-Makro, beschränkt die Relevanz dieser Regel aber auf Ressourcen, die zur Gruppe `g` gehören. Für Ressourcen, die nicht zu dieser Gruppe gehören ist die Einhaltung der Sequenzbeziehung zwischen den Aktivitäten `a` und `b` nicht gefordert. Das `resourceSequence(a,b,i)`-Makro schränkt die Prüfung der Gültigkeit derselben Sequenzbeziehung stärker ein, indem es die Relevanz der Regel auf den Ausführenden `i` beschränkt. Auf alle anderen Ressourcen hat diese Regel

keine Auswirkungen. Diese Begrenzung des Prüfbereichs der Regeln gleicht den Aktivierungsbedingungen von MP-Declare (Abschnitt 3.1.4).

```
direct(a,i) iff
    start(of a) implies start(of a by i)

role(a,r) iff
    start(of a by :i) implies relation(subject i predicate hasRole object r)

separate(a,b) iff
    start(of a by :i) and start(of b) implies start(of b by !=i)

binding(a,b) iff
    start(of a by :i) and start(of b) implies start(of b by i)

caseHandling iff
    forall(task a start(of a) implies start(of a by :p))

bindingRole(a,b) iff
    start(of a by :i1) and relation(subject i1 predicate hasRole object :g)
        and start(of b by :i2) implies relation(subject i2 predicate hasRole
             object g)

capability(a,rt,g) iff
    start(of a by :p) implies relation(subject p
    predicate rt object g

orgDistMulti(a,b,rt) iff
    start(of a by :p1) and start(of b by :p2) implies relation(subject p1
         predicate rt object p2)

roleSequence(a,b,g) iff
    start(of b by :i at :t) and relation(subject i predicate hasRole object g)
         implies complete(of a at <t)

resourceSequence(a,b,i) iff
    start(of b by i at :t) implies complete(of a at <t)

mpSequence(o,val,r1,r2,b) iff
    variablewrite(of o value val at :t by :i) and relation(subject i predicate
        hasRole object r1) implies complete(of b at >t by :j) and relation(
            subject j predicate hasRole object r2)
```

Codeausschnitt 3.9: DPIL-Regelschablonen: Verhaltensorientierte mit Organisatorischer Perspektive und multi-perspektivisch

Ziel von Abschnitt 3.1 ist es, den Zugang zu Notation, Ausdrucksmächtigkeit und Ausführungssemantik einzelner, für die vorliegende Arbeit relevanter Prozessmodellierungssprachen zu erleichtern. Besonderer Fokus liegt dabei auf der multi-perspektivischen, multi-modalen, deklarativen Prozessmodellierungssprache DPIL, welche bezüglich der im Rahmen der vorliegenden Arbeit entwickelten Ansätze (Teil ii) eine zentrale Rolle einnimmt. Einerseits wird sie für die beispielhaften Untersuchungen des später beschriebenen Simulationsansatzes (Abschnitt 4.2) verwendet und andererseits werden auf Basis von DPIL-Modellen Möglichkeiten einer automatisierten Generierung natürlichsprachlicher Prozessbeschreibungen untersucht. Zudem ist auch DPIL eine der Modellierungssprachen, welche zur Evaluation (Abschnitt 8.4.2) des im Rahmen der vorliegenden Arbeit entwickelten Translationskonzepts (Abschnitt 5.2) für Prozessmodelle verwendetet wird.

### 3.1.6 *DPIL-Modell in Anlehnung an realen Prozess*

Nach Einführung der multi-perspektivischen, deklarativen Prozessmodellierungssprache DPIL (Abschnitt 3.1.5), wird nun ihre Verwendung an einem konkreten Beispiel demonstriert. Ziel dieses Abschnitts ist es, ein grundlegendes Verständnis für die Modellierung mit DPIL und den im vorangegangenen Abschnitt deklarierten Regelschablonen zu vermitteln.

```
use group Mitarbeiter
use group Sekretariat
use group DienstreiseAdministration

use repository Localhost {
  url "http://localhost:8080/chemistry-fileshare/atom11"
  user "ai4"
  password "pw123_1!"
  id "ai4"
}

process Dienstreiseantrag {

  // Aufgaben
  task Informieren "Informieren"
  task AntragStellen "Antrag stellen"
  task AntragPruefen "Antrag prüfen"
  task AntragGenehmigen "Antrag genehmigen"
  task UnterkunftBuchen "Unterkunft buchen"
  task DokumenteZusammenstellen "Reisedokumente zusammenstellen"
  // Datenobjekte
  variable Kostenvoranschlag "Kostenvoranschlag"
  document Antrag "Antrag" at Localhost
  variable Antragsstatus "Antragsstatus"
  variable UKosten "U-Kosten"
  variable Fixkosten "Fixkosten"

  // Regeln der Verhaltensorientierten und Datenorientierten Perspektive
  ensure once(Informieren)
  ensure once(AntragStellen)
  ensure once(AntragPruefen)
  ensure once(AntragGenehmigen)
  ensure sequence(Informieren, AntragStellen)
  ensure produce(AntragStellen, Antrag)
  ensure consume(AntragPruefen, Antrag)
  ensure complete(of AntragPruefen at :t) implies variablewrite(of
    Antragsstatus value "geprüft" at <t)
  ensure start(of AntragGenehmigen at :t) implies variablewrite(of
    Antragsstatus value "geprüft" at <t)
  advise sequence(AntragGenehmigen, UnterkunftBuchen)
  ensure complete(of DokumenteZusammenstellen at :t) implies
    variablewrite(of Antragsstatus value "archiviert" at <t)
  // Regeln der Organisatorischen Perspektive
  ensure binding(Informieren, AntragStellen)
  ensure binding(AntragStellen, UnterkunftBuchen)
  ensure roleSequence(AntragGenehmigen, UnterkunftBuchen, Mitarbeiter)
  ensure role(AntragPruefen, Sekretariat)
  ensure role(AntragGenehmigen, DienstreiseAdministration)
  ensure role(DokumenteZusammenstellen, Sekretariat)

  // Prozessziele
  milestone "End of process": complete(of DokumenteZusammenstellen) or
    variablewrite(of Antragsstatus value "abgelehnt")
}
```

Codeausschnitt 3.10: DPIL-Modell des Uni-BT-Dienstreiseprozesses

Im dargestellten Beispiel-Modell (Codeausschnitt 3.10) ist eine vereinfachte Version eines universitätsinternen Abwicklungsprozesses für Dienstreisen beschrieben. Zunächst werden die in einem entsprechenden Organisationsmodell definierten organisatorischen Gruppen ins DPIL-Modell übernommen. Anschließend werden Adresse und Zugangsdaten für ein Dokumentrepositorium angegeben. Damit ist die Präambel abgeschlossen. Im ersten Teil des eigentlichen Prozessmodells, welches mit *Dienstreiseantrag* benannt ist, werden die im Prozess zu bewältigenden Aufgaben deklariert. Unmittelbar danach erfolgt die Deklaration der für den Prozess relevanten Datenobjekte. Variablen sind dabei nur innerhalb der jeweiligen Prozessinstanz sichtbar, während Dokumente darüber hinaus persistent sind. Aus diesem Grund muss für ein Dokument auch das Ziel-Repositorium angegeben werden, wie das Dokument *Antrag* im Beispiel zeigt. Damit ist die Deklaration der prozessrelevanten Entitäten abgeschlossen.

Im zweiten Teil des Prozessmodells werden mittels DPIL-Regeln Einschränkungen für valide Prozessverläufe formuliert. Die dafür verwendete Bibliothek für Regelschablonen muss dazu in einem eigenen DPIL-Modell gekapselt sein. Die ersten vier Regeln (`once...`) stellen sicher, dass alle Aufgaben außer *UnterkunftBuchen* und *DokumenteZusammenstellen* höchstens einmal ausgeführt werden. Die darauffolgende `sequence`-Regel zwingt jeden Antragsteller dazu, sich zuerst über die zu erwartenden Reisekosten zu informieren. Das Regelpaar `produces-consumes` sagt einerseits aus, dass das Dokument *Antrag* im Rahmen der Bearbeitung von Aufgabe *AntragStellen* erzeugt und während der Aktivität *AntragPruefen* verwertet wird. Dadurch entsteht implizit ebenfalls eine zeitliche Abhängigkeit zwischen den beiden genannten Aufgaben. Die zwei nachfolgenden Regeln greifen nicht auf vordefinierte Makros der Schablonenbibliothek zurück. Sie legen fest, dass die Aufgabe *AntragPruefen* erst dann als beendet gelten kann, wenn der *Antragsstatus* mit dem Wert *geprüft* belegt ist, was gleichzeitig die Voraussetzung für den Beginn der Bearbeitung von Aufgabe *AntragGenehmigen* darstellt. Die darauffolgende `sequence`-„Regel" ist durch das Schlüsselwort `advise` als Empfehlung gekennzeichnet und besagt, dass es sinnvoll ist, erst dann eine Unterkunft zu buchen, wenn der Antrag genehmigt ist.

Nach der Formulierung von zeitlichen Abhängigkeiten unterschiedlicher Art, ist auch die korrekte Assoziation von Aufgaben mit Ressourcen im Prozessmodell zu reglementieren. Die beiden `binding`-Regeln legen beispielsweise fest, dass die Aufgabenpaare *Informieren* und *AntragStellen* respektive *AntragStellen* und *UnterkunftBuchen* jeweils von derselben Person ausgeführt werden müssen – was implizit auch dazu führt, dass *Informieren* und *UnterkunftBuchen* denselben Ausführenden haben müssen. Die im vorherigen Absatz beschriebene Empfehlung, auf die Genehmigung des Antrags zu warten, bevor eine Unterkunft gebucht wird, kann ignoriert werden. Bekleidet der Antragsteller jedoch die Rolle *Mitarbeiter*, dann ist die Einhaltung dieser Reihenfolge durch die im Beispielmodell enthaltene `roleSequence`-Regel verpflichtend. Die Kombination aus dieser und der weiter oben genannten Empfehlung erlauben die Modellierung von Kontrollflussvariationen auf Basis der Informationen aus dem organisatorischen Modell. Die drei letzten Prozessregeln sind Einschränkungen hinsichtlich der Rolle des Ausführenden der drei in den Regeln parametrierten Aktivitäten.

Als letzter Teil des Prozessmodells ist ein Meilenstein definiert. Dieser legt fest, dass eine Instanz des Prozesses *Dienstreiseantrag* genau dann als beendet angesehen wird, sobald entweder die Aufgabe *DokumenteZusammenstellen* abgeschlossen ist oder der *Antragsstatus* mit dem Wert *abgelehnt* belegt ist.

Das in diesem Abschnitt vorgestellte DPIL-Prozessmodell basiert auf einem realen Universitätsprozess und dient der Illustration von Verwendung und Semantik der Prozessmodellierungssprache DPIL. Zusätzlich wird es in Abschnitt 3.3.1 zur Erläuterung von Ereignisprotokollen wiederverwendet. Aus sich daraus ergebenden Gründen der Übersichtlichkeit und Verständlichkeit wird das Modell an einigen Stellen vereinfacht.

## 3.2 EREIGNISORIENTIERTE SIMULATION

Die Abschnitte dieses Teilkapitels beschäftigen sich mit dem Prinzip der *Ereignisorientierten Simulation* (engl. *Discrete-Event Simulation*). Die Ereignisorientierte Simulation bildet eine abstrakte Grundlage für das Verständnis der Simulation von Prozessen auf Basis imperativer oder deklarativer Prozessmodelle. Dieses Verständnis ist notwendig, um die grundlegenden Unterschiede existierender Simulationstechniken gegenüber der im Rahmen dieser Arbeit entwickelten (Abschnitt 4.2) nachvollziehen zu können. Der prägnanteste Unterschied ist hier die Simulationsdurchführung, die bei letzterer auf dem Lösen von booleschen Gleichungssystemen basiert. Nahezu alle bisherigen Verfahren basieren dagegen auf einem der nachfolgend vorgestellten Prinzipien der Simulationsdurchführung, die vollständig auf einer Kette lokaler Entscheidungen basieren. Deren Komplexität ist häufig sehr hoch, weswegen sie bei Simulationen häufig durch Zufallsvariablen approximiert werden. Daher wird in Abschnitt 3.2.3 detaillierter auf dafür gängige Berechnungsstrategien eingegangen. Abschnitt 3.2.5 konkretisiert die Grundlagen der allgemeinen, ereignisorientierten Simulation und gibt eine kurze Einführung in das Gebiet der Prozesssimulation. Dies dient der Hinführung zu der im Rahmen der vorliegenden Arbeit diskutierten Anwendung von Simulationstechniken zur Generierung künstlicher Prozessaufzeichnungen.

### 3.2.1 *Systeme und Simulationsmodelle*

*Simulation*

*Simulation* beschäftigt sich allgemein mit der künstlichen Nachahmung des Verhaltens realer Prozesse und Systeme [22]. Eine der wesentlichen Aufgaben ist dabei die Formulierung von Annahmen über das Verhalten des Systems zu einem *Simulationsmodell*. Diese Annahmen beschreiben Beziehungen zwischen den Entitäten des fraglichen Systems. Letztere besitzen Attribute und können an Aktivitäten teilnehmen. Grundlegend können mit einem solchen Modell verschiedene „Was-wäre-wenn-Fragen" beantwortet werden. Beispielsweise kann im Kontext eines Brieftransportsystems eine Analyse des Einflusses der Ladungskapazität der Transportmittel, basierend auf der Frage „Was wäre, wenn die Ladungskapazität jedes Transportmittels verdoppelt werden würde", durch Simulation unterstützt werden.

*Simulationsmodell*

*Systeme und Komponenten*

Ein *System* setzt sich aus mehreren *Komponenten* zusammen [22]. Diese sind am

Beispiel der Bestandteile des eben genannten Brieftransportsystems in folgender Liste zusammengefasst:

- Entitäten: z.B. Transportmittel, Fahrer,
- Attribute: z.B. Ladekapazität (Transportmittel), max. Arbeitsstunden (Fahrer),
- Aktivitäten: z.B. Verladen, Transport ausführen,
- Ereignisse: z.B. Verladen abgeschlossen, Ankunft am Briefzielort,
- Zustandsvariablen: z.B. Anzahl der im Depot zum Transport bereitliegenden Briefe, Anzahl der bereits auf dem Transportweg befindlichen Briefe

Ein System beinhaltet in diesem Kontext also eine Gruppe von Entitäten, die zweckorientiert interagieren und voneinander abhängig sein können. Der Zustand, in dem sich ein System befindet, wird über sogenannte Zustandsvariablen definiert. *Ereignisse* können den Zustand des Systems verändern und werden durch die systeminternen Aktivitäten (endogen) oder systemexterne Einflüsse (exogen) seitens der *Systemumgebung* hervorgerufen [22]. Als Systemumgebung werden Einflussfaktoren aufgefasst, die sich der Kontrolle des Systems entziehen. Für das verwendete Brieftransportsystem sind folglich die Faktoren, die das Eintreffen von Briefen steuern, als Teil der Systemumgebung und damit als exogene Einflüsse anzusehen.

*Zustandsvariablen und Ereignisse*

*Systemumgebung*

Für die vorliegende Arbeit ist die Unterscheidung zwischen *diskreten* und *kontinuierlichen* Systemen [22] von zentraler Bedeutung. Ändern sich die Zustandsvariablen des Systems ausschließlich in diskreten Zeitschritten, dann wird auch das System als diskret angesehen. Kontinuierliche Änderungen der Zustandsvariablen über die Zeit bedeuten hingegen, dass es sich um ein kontinuierliches System handelt. Das genannte Brieftransportsystem ist beispielsweise ein diskretes System, während ein Stausee bezüglich des Wasserspiegels durch kontinuierlichen Zu- oder Abfluss eher als kontinuierliches System zu betrachten ist.

*Diskrete vs. Kontinuierliche Systeme*

Zwei weitere grundlegende Unterscheidungen können auf Basis des Typs des Simulationsmodells vorgenommen werden. Einerseits können Simulationsmodelle *statisch* oder *dynamisch* sein. Andererseits lassen sie sich auch in *deterministisch* und *stochastisch* klassifizieren [22]. Dynamisch sind Simulationsmodelle dann, wenn sie Systeme beschreiben, deren Zustand sich mit der Zeit verändert. Folgerichtig beschreiben statische Modelle Systeme, deren Zustand über die Zeit unverändert bleibt. Als deterministisch wird ein Simulationsmodell bezeichnet, welches keinerlei *Zufallsvariablen* beinhalten. Damit erzeugt diese Klasse von Simulationsmodellen für dieselbe Eingabe auch immer dieselbe Ausgabe. Die Beurteilung der Kreditwürdigkeit eines Bankkunden ist ein Beispiel für einen deterministischen Prozess. Stochastische Modelle nutzen eine oder mehrere Zufallsvariablen als Möglichkeit der Annäherung eines komplexen, dynamischen Verhaltens des Systems. Die Ankunftsrate von Briefen im Brieftransportsystem würde beispielsweise mittels einer Zufallszahl approximiert werden. Eine interessante Anwendung für einen Simulator ist dann, eine Grenze für die Ankunftsrate zu ermitteln, bis zu welcher die Zustellungszeit als konstant betrachtet werden kann.

*Statische vs. Dynamische Modelle*

*Deterministische vs. Stochastische Modelle*

*Ereignisorientierte Simulation*

Auf Basis der bisher getroffenen Unterscheidungen kann nun auch der Begriff *Ereignisorientierte Simulation* (engl. *Discrete-Event System Simulation*) definiert werden. Sie ist die künstliche Nachahmung realer Systeme oder Prozesse, deren Zustand sich in diskreten Zeitschritten verändert. Ein Ereignis kapselt folglich die entsprechenden Änderungen der Zustandsvariablen des Modells. Derartige Simulationsmodelle werden üblicherweise mit numerischen statt mit analytischen Mitteln „gelöst". Als Lösung wird hier die Durchführung der Simulation zur Ermittlung der Ausgaben auf Basis entsprechender Eingaben bezeichnet.

### 3.2.2 *Allgemeine Prinzipien der Simulationsdurchführung*

*Simulationsdurchführung*

Ausgehend von einem Simulationsmodell betrachtet die *Simulationsdurchführung* die Mechanismen zur künstlichen Erzeugung von Ereignissen, die ein reales Systemverhalten imitieren sollen. Im aktuellen Abschnitt werden mehrere alternative Prinzipien für die Ereigniserzeugung zusammengefasst.

Zunächst wird ein grundlegendes Prinzip zur Simulationsdurchführung beschrieben. Alternative Ansätze stellen in den meisten Fällen lediglich Verbesserungen desselben dar. Die Erzeugung von Ereignissen geschieht grundlegend in diskreten Zeitschritten. Die chronologische Reihenfolge der Erzeugung der Ereignisse wird dabei in einer sogenannten *Zukunftsereignisliste (ZEL)* [22, S. 110 ff.] festgehalten. Diese Liste enthält alle *Ankündigungen* der Ereignisse, die zukünftig auftreten werden. Der Hintergrund ist, dass immer, wenn der Beginn einer Aktivität im System simuliert wird, deren Dauer berechnet wird. Dann wird eine neue Ereignisankündigung erzeugt, welche das Ende der Aktivität mit der entsprechenden Zeitinformation assoziiert. Dieses Ankündigung wird dann in die ZEL eingefügt. Zu jedem Zeitpunkt $t$ enthält die ZEL alle vorher eingereihten zukünftigen Ereignisankündigungen sowie die assoziierte Ereigniszeit. Diese Liste ist nach der Ereigniszeit geordnet – jedes Ereignis erfüllt demnach folgende Bedingung:

*Zukunftsereignisliste mit Ereignisankündigungen*

$$t < t_1 \leqslant t_2 \leqslant t_3 \leqslant \cdots \leqslant t_n \tag{7}$$

In jedem Zeitschritt wird eine sogenannte *prototypische Momentaufnahme* erzeugt, die den Fortschritt der Simulation, die aktuelle ZEL selbst und auch den jeweils aktuellen Zustand des simulierten Systems beschreibt. Ein Fortschritt in Bezug auf die Simulationsdurchführung wird mittels der ZEL und gemäß des sogenannten *Event-scheduling/time-advance-Algorithmus* erzielt. Dieser beinhaltet, hier vom Startzeitpunkt der Simulation $t_0$ ausgehend, die folgenden fünf Schritte:

*Event-scheduling/time-advance-Algorithmus*

Schritt 1: Entferne die Ereignisankündigung des Zeitschritts $t_1$ aus der ZEL,

Schritt 2: Erhöhe den Zeitschrittzähler $t$ von $t_0$ auf $t_1$,

Schritt 3: Führe das in Schritt 1 entfernte Ereignis aus, wobei der Systemzustand aktualisiert wird,

Schritt 4: Generiere, falls notwendig, zukünftige Ereignisse und sortiere ihre Ankündigung gemäß der geplanten Ereigniszeit in die ZEL ein,

Schritt 5: Aktualisiere die kumulative Statistik zur Auswertung am Ende der Simulationsdurchführung.

Ausgegangen wird von einer initialen Momentaufnahme des Systems zum Zeitpunkt 0. Diese definiert sich vor allem durch die initiale Belegung der Zustandsvariablen des Systems und die Generierung sogenannter *exogener* Ereignisse. Der *Initialzustand* des Brieftransportsystems kann beispielsweise durch die initial bereitstehende Anzahl an Transportmitteln charakterisiert sein. Angenommen, dass im realen System mit dem Verladen der Briefe begonnen wird, sobald eine genügend große Anzahl an Briefen – eine sogenannte *Transporteinheit* – im Depot zum Versand bereitliegt. Da dieses Bereitlegen nicht Teil des modellierten Brief*transport*systems ist, wird das entsprechende Ereignis der Bereitstellung der Transporteinheit als exogen angesehen. Eine der wichtigsten Unterklassen exogener Ereignisse sind demnach die sogenannten *Startereignisse*. Jedes dieser Ereignisse reiht die Ankündigung eines weiteren Startereignisses in die ZEL ein. Ein einzelnes Endereignis terminiert dagegen die komplette Simulationsdurchführung. Die Festlegung dieses Endereignisses kann dabei schon vor der Simulationsdurchführung angekündigt, d.h. in die ZEL mit einem festgelegten Zeitpunkt eingefügt werden. Ein weiterer Weg ist, dieses Ereignis durch die Überwachung der Simulationsdurchführung zu erzeugen, beispielsweise indem ein Zähler die Anzahl der bereits eingefügten Startereignisse aufzeichnet und die Simulation mit einer maximalen Anzahl dieser Ereignisse parametriert wird. Unabhängig davon, welche dieser beiden Varianten gewählt wird, ist die *Ankunftsrate* der Startereignisse ein entscheidender Faktor für die abschließende statistische Auswertung der Leistungsfähigkeit des simulierten Systems. Zudem entscheidet die Ankunftsrate in Verbindung mit der Durchführungszeit der Aktivitäten und dem geplanten Endereignis über die maximale Anzahl von Simulationsdurchläufen.

*Initialzustand*

*Startereignisse*

*Ankunftsrate*

Ein alternativer Ansatz zur Simulationsdurchführung, welcher das eben vorgestellte grundlegende Prinzip erweitert, ist der *Process-interaction-Algorithmus* [22, S. 114 f.]. Die Ausführung von Aktivitäten kann beispielsweise von der Verfügbarkeit bestimmter Entitäten abhängig sein. Der Lebenszyklus jeder Entität wird dabei als Prozess modelliert, wodurch eine Interaktion zwischen Entitäten als Interaktion zwischen Prozessen abgebildet wird. Auch die Abhängigkeit von Ressourcen, deren Kapazität in der Regel begrenzt ist, wird in diesem Algorithmus berücksichtigt. Das geschieht, indem zusätzlich zur ZEL Warteschlangen für Ressourcen aufgebaut werden. Benötigt eine Entität für die Bewältigung einer Aufgabe Zugriff auf eine bestimmte Ressource, wobei diese gerade von einer anderen Entität beansprucht wird, muss erstere Entität pausieren und in der ressourcenbezogenen Warteschlange auf die Freigabe derselben warten. Damit setzt sich ein Prozess folglich aus einer zeitlich geordneten Liste an Ereignisankündigungen, Aktivitäten und durch Ressourcenmangel hervorgerufenen Verzögerungen zusammen.

*Process-interaction-Algorithmus*

Sowohl der Event-scheduling/time-advance- als auch der Process-interaction-Algorithmus basieren auf einem zwar diskreten aber variablen Zeitfortschritt, der durch die geplante Auslösezeit des nächstfolgenden Ereignisses definiert wird. Dagegen nutzt der dritte Ausführungsansatz, der *Activity-scanning-Ansatz*, diskrete und konstante Zeitschritte. Die Entscheidung, ob eine bestimmte Aktivität zum jeweiligen Zeitschritt begonnen wird, wird in diesem Ansatz regelbasiert getroffen. Jede Aktivität, die alle Regeln erfüllt, wird ausgeführt. Da zu jedem Zeitschritt für jede Aktivität geprüft werden muss, ob eine Aktivität ausgeführt werden kann,

*Activity-scanning-Ansatz*

neigt dieser Ansatz zu einer eher schlechten Laufzeit. Aus diesem Grund wird der reine Activity-scanning-Ansatz um Elemente des Event-scheduling/time-advance-Algorithmus erweitert. Im Wesentlichen will man damit die regelbasierte Entscheidung, ob eine Aktivität begonnen werden kann beibehalten, jedoch die Häufigkeit der Regelprüfungen durch die Verwendung dynamischer Zeitschritte reduzieren. Weiterhin werden Ereignisse als Aktivitäten mit einer Ausführungsdauer von 0 Zeiteinheiten aufgefasst und Aktivitäten werden in zwei Kategorien unterteilt:

B-Aktivitäten: Verpflichtend durchzuführende Aktivitäten, wie exogene Ereignisse und unbedingte Aktivitäten,

C-Aktivitäten: Bedingte Aktivitäten oder Ereignisse, deren Ausführung von der Erfüllung einer oder mehrerer Regeln abhängt.

Der daraus resultierende *Three-phase-Algorithmus* kann wie folgt beschrieben werden:

Phase 1: Entferne die nächstfolgende Ereignisankündigung aus der ZEL und setze Zeitschrittzähler auf die Zeit dieser Ereignisankündigung. Entferne außerdem alle weiteren Ereignisankündigungen mit derselben Auslösezeit,

Phase 2: Löse alle B-Typ-Ereignisse aus, die aus der ZEL entfernt wurden,

Phase 3: Suche die Regeln, welche die C-Typ-Aktivitäten auslösen und aktiviere jede, die alle Regeln erfüllt. Wiederhole diese letzte Phase solange bis keine weitere C-Typ-Aktivität mehr begonnen und kein Ereignis ausgelöst wird.

Die in diesem Abschnitt zusammengefassten Konzepte beschreiben mehrere Ansätze zur Durchführung der Simulation, die sich hinsichtlich der unterschiedlichen Steuerung des Zeitfortschritts voneinander abgrenzen lassen. Grundlegend kristallisiert sich ein dynamisches Zeitinkrement, auf Basis einer Liste mit zukünftig geplanten Ereignissen, als beste Steuerungsmöglichkeit heraus. Die Bevorzugung einer Entscheidung, ob Aktivitäten durch Prozessinteraktionen oder durch die Erfüllung der Regeln eines Regelsatzes ausgelöst werden, ist dagegen nicht eindeutig. In Bezug auf die Anwendung für die Simulation von Geschäftsprozessen auf Basis konzeptueller Geschäftsprozessmodelle hängt diese Entscheidung von mehreren Faktoren ab (Abschnitt 1.1.4 und Abschnitt 1.2). Wie in [22, S. 116] angedeutet wird, lassen sich die meisten, imperativen Simulationsmodelle auf den Process-interaction-Ansatz abbilden. Der Activity-scanning- oder der Three-phase-Ansatz ist aufgrund des jeweils deklarativen Charakters für die Abbildung von Simulationsproblemen auf Basis deklarativer Prozessmodelle prädestiniert.

### 3.2.3 *Mathematische Modelle und Zufallsvariablen*

Im vorangegangenen Abschnitt werden im Rahmen der Simulationsdurchführung mehrere Berechnungen durchgeführt. Diese umfassen vor allem die Dauer von Aktivitäten und die Ankunftsrate exogener Ereignisse, wie beispielsweise von Startereignissen. Aber auch die Anzahl verfügbarer Ressourcen und die initiale Belegung

der Zustandsvariablen muss ermittelt werden. Aus diesem Grund beschäftigt sich dieser Abschnitt überblicksweise mit *mathematischen* und *statistischen* Modellen zur Berechnung dieser Parameter. Um eine Simulationsdurchführung realitätsnäher zu gestalten, werden komplexe Zusammenhänge mittels Zufallszahlen modelliert, welche auf einer passenden Häufigkeitsverteilung basieren.

---

**Definition 3.1 (Diskrete Zufallsvariable).**

$$X \in R_x \text{ und } R_x = \{x_1, x_2, ..., x_i\}, i = 1, 2, ... \tag{8}$$

*Dann gilt für jedes* $x_i \in R_x$ *die Wahrscheinlichkeit* $p(x_i) = P(X = x_i)$*, dass die Zufallsvariable mit* $x_i$ *belegt wird. Für jedes* $p(x_i)$ *muss gelten:*

$$p(x_i) \geqslant 0, \text{für alle } i \tag{9}$$

$$\sum_{i=1}^{\infty} p(x_i) = 1 \tag{10}$$

---

*Diskrete vs. kontinuierliche Zufallsvariablen*

*Zufallsvariablen* können *diskret* oder *kontinuierlich* sein. Eine Zufallsvariable $X$ ist diskret, wenn die Anzahl ihrer möglichen Werte entweder finit oder zählbar unendlich ist. Nicht alle Werte in $X$ müssen in jedem Simulationskontext realistisch sein. Aus diesem Grund gilt, dass für jeden Wert in $X$ die Wahrscheinlichkeit auch 0 sein kann. Die Summe der Wahrscheinlichkeiten aller diskreten Belegungen der Zufallsvariable muss 1 ergeben (Gleichung 10). Ein Beispiel für eine diskrete Zufallsvariable ist die Anzahl der täglich ankommenden Briefe im vorherig verwendeten Brieftransportsystem-Beispiel.

---

**Definition 3.2 (Kontinuierliche Zufallsvariable).**

$$P(a \leqslant X \leqslant b) = \int_a^b f(x)\,dx \tag{11}$$

*Die Funktion* $f(x)$ *bezeichnet hier die sogenannte* Wahrscheinlichkeitsdichtefunktion (WDF) *der Zufallsvariable X und erfüllt folgende Bedingungen:*

$$f(x) \geqslant 0, \text{für alle } x \in R_x \tag{12}$$

$$\int_{R_x} f(x)\,dx = 1 \tag{13}$$

$$f(x) = 0, wenn\ x \notin R_x \tag{14}$$

---

Wenn der Wertebereich von $X$ dagegen durch ein Intervall oder eine Menge von Intervallen definiert ist, handelt es sich um eine kontinuierliche Zufallsvariable. Damit kann die Wahrscheinlichkeit, dass $X$ im Intervall $[a, b]$ liegt, mit Gleichung 11

angegeben werden. Auch für kontinuierliche Zufallsvariablen gilt, dass jeder gültige Wert mit einer nicht-negativen Wahrscheinlichkeit als Belegung für $X$ selektiert wird (Gleichung 12). Die Wahrscheinlichkeit, dass $x \in R_x$ gilt ist 1. Wenn $x$ kein Wert aus $R_x$ ist, dann ist seine Wahrscheinlichkeit 0 (Konsequenzen aus Gleichung 13 und Gleichung 14). Ein Beispiel für eine kontinuierliche Zufallsvariable ist die Lebensdauer der Transportmittel im Brieftransportsystem. Die Werte dieser Zufallsvariable können beispielsweise mittels einer Exponentialverteilung errechnet werden.

Die Belegung einer Zufallsvariable mit einem konkreten Wert zum Zeitpunkt der Simulationsdurchführung kann manuell erfolgen. Häufig kann, besonders bezüglich exogener Ereignisse, lediglich symptomatisch beobachtet werden, in welcher Häufigkeit und Frequenz diese Ereignisse auftreten. Zudem sind exogene Ereignisse per Definition nicht Teil des modellierten Systems. Aus diesem Grund wird die erkannte Symptomatik häufig mittels *Wahrscheinlichkeitsverteilungen* approximiert [2, 22]. Das bedeutet beispielsweise, dass nicht bekannt ist, welche Faktoren und welche Regeln die Ankunft von Briefen im Brieftransportsystem kontrollieren. Dafür ist jedoch durch Beobachtung bekannt, wie viele Briefe im *Durchschnitt* täglich auf den Versandweg gebracht werden und wie groß die *Schwankungen* sind. Das bedeutet, dass für die Belegung der Zufallsvariablen des modellierten Systems eine *Zufallszahl* gemäß einer gewünschten Wahrscheinlichkeitsverteilung generiert werden kann. Da Computer deterministisch sind, ist die generierte Zahl immer eine sogenannte Pseudo-Zufallszahl. Sogenannte *Zufallsgeneratoren* errechnen auf Basis unterschiedlicher Startwerte unterschiedliche Pseudo-Zufallszahlen. Diese Startwerte können entweder automatisch, beispielsweise in Form des Systemtakts, oder manuell durch eine Nutzereingabe ermittelt werden. Um die Wiederholbarkeit eines Simulationsexperiments sicherzustellen, sollte der Startwert manuell angegeben werden [2].

*Wahrscheinlichkeitsverteilungen*

*Zufallszahl*

*Zufallsgeneratoren*

In den seltensten Fällen kann für die Generierung der benötigten Zufallszahlen eine Gleichverteilung angenommen werden [22]. Aus diesem Grund greift man auf verschiedene Wahrscheinlichkeitsverteilungen zurück. Eine Auswahl der gebräuchlichsten Verteilungen sind in Tabelle 3.3 dargestellt, klassifiziert nach kontinuierlichen und diskreten Wertebereichen.

Die Tabelle stellt kontinuierliche und diskrete Verteilungen hinsichtlich ihres Wertebereichs und der zugehörigen Wahrscheinlichkeitsfunktionen respektive -dichtefunktionen dar. Für diskrete Verteilungen kann die Wahrscheinlichkeit, dass die Zufallsvariable $X$ den Wert $x_i$ annimmt (also $P(X = x_i)$), berechnet werden. Bei kontinuierlichen Verteilungen gilt für jedes $x_i$ die Wahrscheinlichkeit $P(X = x_i) = 0$. Grund ist das kontinuierliche Spektrum und die dadurch unbegrenzte Anzahl an Werten des Wertebereichs. Aus diesem Grund greift man für kontinuierliche Verteilungen auf sogenannte *Wahrscheinlichkeitsdichtefunktionen* zurück. Diese geben an, wie wahrscheinlich die Zufallsvariable $X$ einen Wert von *maximal* $x_i$ annimmt. Ergänzend sind auch der Erwartungs- oder Mittelwert ($\mathbb{E}(X)$) sowie die Varianz ($Var(X)$), welche sich aus der Standardabweichung vom Mittelwert ableitet, aufgeführt.

*Wahrscheinlichkeitsdichtefunktionen*

Die Wahl einer passenden Verteilung hängt vom Betrachtungsgegenstand ab. Generell können auf Grundlage von Beobachtungswerten mittels der sogenannten

| Verteilung | Wertebereich | Wahrscheinlichkeit | $\mathbb{E}(X)$ | $Var(X)$ |
|---|---|---|---|---|
| Kontinuierliche Wahrscheinlichkeitsdichtefunktionen: $f_X(x)$ | | | | |
| Gleich<br>$(a < b)$ | $a \leqslant x \leqslant b$ | $\frac{1}{b-a}$ | $\frac{a+b}{2}$ | $\frac{(b-a)^2}{12}$ |
| Exponential<br>$(\lambda > 0)$ | $x \geqslant 0$ | $\lambda e^{-\lambda x}$ | $\frac{1}{\lambda}$ | $\frac{1}{\lambda}$ |
| Standard-Normal<br>$(\mu \in \mathbb{R}; \sigma > 0)$ | $x \in \mathbb{R}$ | $\frac{1}{\sqrt{2\pi\sigma^2}} e^{\frac{-(x-\mu)^2}{2\sigma^2}}$ | $\mu$ | $\sigma^2$ |
| Gamma<br>$(r, \lambda > 0)$ | $x > 0$ | $\frac{\lambda(\lambda x)^{\sigma-1} e^{-\lambda x}}{\Gamma(r)}$ | $\frac{r}{\lambda}$ | $\frac{r}{\lambda^2}$ |
| Erlang<br>$(\lambda > 0, r \in \{1, 2, ...\})$ | $x > 0$ | $\frac{\lambda(\lambda x)^{\sigma-1} e^{-\lambda x}}{(r-1)!}$ | $\frac{r}{\lambda}$ | $\frac{r}{\lambda^2}$ |
| Dreiecks<br>$(a < b; a \leqslant c \leqslant b)$ | $a \leqslant x \leqslant b$ | $\begin{cases} 0 & x < a \\ \frac{2(x-a)}{(b-a)(c-a)} & a \leqslant x \leqslant c \\ \frac{2}{b-a} & x = c \\ \frac{2(b-x)}{(b-a)(b-c)} & c < x \leqslant b \\ 1 & b < x \end{cases}$ | $\frac{a+b+c}{3}$ | $\frac{a^2+b^2+c^2-ab-ac-bc}{18}$ |
| Log.-Normal<br>$(\mu \in \mathbb{R}; \sigma > 0)$ | $x \geqslant 0$ | $\frac{1}{x\sigma\sqrt{2\pi}} e^{\frac{-(\ln x-\mu)^2}{2\sigma^2}}$ | $e^{\mu+\frac{\sigma^2}{2}}$ | $\left(e^{\sigma^2}\right) e^{2\mu+\sigma^2}$ |
| Beta<br>$(a < b; r, s > 0)$ | $a \leqslant x \leqslant b$ | $\frac{1}{b-a} \frac{\Gamma(r+s)}{\Gamma(r)\Gamma(s)} \left(\frac{x-a}{b-a}\right)^{r-1} \left(\frac{b-x}{b-a}\right)^{s-1}$ | $a + (b-a)\frac{r}{b-a}$ | $\frac{rs(b-a)^2}{(r+s)^2(r+s+1)}$ |
| Diskrete Wahrscheinlichkeitsfunktionen: $P(X = x_i)$ | | | | |
| Bernoulli<br>$(0 \leqslant p \leqslant 1)$ | $x_i \in \{0, 1\}$ | $\begin{cases} 1-p & x_i = 0 \\ p & x_i = 1 \end{cases}$ | $p$ | $p(1-p)$ |
| Binomial<br>$(0 \leqslant p \leqslant 1; n \in \{1, 2, ...\})$ | $x_i \in \{0, 1, ..., n\}$ | $\binom{n}{x_i} p^{x_i}(1-p)^{n-x_i}$ | $np$ | $np(1-p)$ |
| Geometrische<br>$(0 \leqslant p \leqslant 1)$ | $x_i \in \{1, 2, ...\}$ | $p(1-p)^{x_i-1}$ | $\frac{1}{p}$ | $\frac{1-p}{p^2}$ |
| Homogene<br>$(a < b)$ | $x_i \in \{a, ..., b\}$ | $\frac{1}{(b-a)+1}$ | $\frac{a+b}{2}$ | $\frac{(b-a)((b-a)+2)}{12}$ |
| Poisson<br>$(\lambda > 0)$ | $x_i \in \{0, 1, ...\}$ | $\frac{\lambda^k!}{k!} e^{-\lambda}$ | $\lambda$ | $\lambda$ |

Tabelle 3.3: Kontinuierliche und diskrete Wahrscheinlichkeitsverteilungen

*Ausgleichsrechnung* [138] die unbekannten Parameter für eine gegebene Funktion geschätzt werden. Beispiele für oft verwendete Methoden der Ausgleichsrechnung sind *Regressionanalysen* und *Fitting*. Standardmäßig wird im Falle der Regressionsanalyse die *Methode der kleinsten Quadrate* [22] eingesetzt. Grundsätzlich muss also eine Idee existieren, welcher Typ der Wahrscheinlichkeitsverteilungen die beobachteten Daten am besten beschreiben kann. Für die Ereignisorientierte Simulation existieren hierfür bereits Erfahrungswerte [2].

*Ausgleichsrechnung*

Der *Ankunftsprozess* exogener Startereignisse wird häufig mittels einer *Negativ-Exponentialverteilung* modelliert. Wenn die Startereignisse sequentiell eintreten, der Eintrittszeitpunkt des ersten Startereignisses keinen Einfluss auf die Verteilung hat und das Eintreten früherer Startereignisse das Eintreten späterer Startereignisse in keiner Form beeinflusst, spricht man auch vom sogenannten *Poissonprozess*.

*Ankunftsprozess und Negativ-Exponentialverteilung*

*Poissonprozess*

Eine Dauer wird hingegen meist mittels einer *Normal-* oder einer *Betaverteilung* approximiert. Die Standard-Normalverteilung ist zwar ein sehr weit verbreitetes und vergleichsweise einfaches Modell, birgt jedoch die Gefahr, dass die zugehörige Zufallsvariable auch negative Werte annehmen kann (Tabelle 3.3). In [2] wird vorgeschlagen, eine Standard-Normalverteilung ausschließlich dann für die Modellierung einer Dauer zu verwenden, wenn die Wahrscheinlichkeit, dass die entsprechende Zufallsvariable negative Werte annimmt, sehr gering ist. Mit $\mathbb{E}(X) - 2\sqrt{Var(X)} < 0$ wird auch eine Faustregel angegeben, wann die

*Standard-Normalverteilung*

Standard-Normalverteilung *nicht* verwendet werden sollte. Alternativ könnte auch, falls ein negativer Wert generiert wird, solange ein neuer Wert erzeugt werden, bis dieser positiv ist. Das würde aber den Mittelwert und die Varianz der Verteilung beeinflussen, sodass dieses Mittel nicht empfohlen wird. Die Betaverteilung ist besonders für Fälle geeignet, bei denen die Zufallsvariable eine klare untere und obere Grenze aufweisen. Die Betaverteilung ist, durch die Parameter $a$ und $b$ auf den Wertebereich $a \leqslant x_i \leqslant b$ und damit auf ein finites Intervall beschränkt. Ein Vorteil der Betaverteilung ist, dass mittels der Parameter $r$ und $s$ sehr unterschiedliche Funktionen gestaltet werden können.

*Betaverteilung*

Die Entscheidung, welche der Verteilungen am besten geeignet ist, ist nur für den individuellen Fall möglich. Aus diesem Grund wird von einer detaillierten Beschreibung *aller* möglichen Verteilungstypen abgesehen. Der aktuelle Abschnitt soll damit lediglich zeigen, dass mit statistischen Mitteln Verhalten modelliert werden kann, welches von Unsicherheitsfaktoren betroffen ist. Die meisten Prozesse müssen flexibel auf schwer oder gänzlich nicht zu kontrollierende exogene Einflüsse reagieren, weswegen Verfahren zur Simulation derselben von Wahrscheinlichkeitsverteilungen Gebrauch machen. Das ermöglicht die Modellierung des *Beobachteten*, ohne die Ursachen vollständig analysieren zu müssen.

### 3.2.4 *Verifikation und Validierung von Simulationsmodellen*

Im Verlauf der Entwicklung des Simulationsmodells muss der Modellentwickler mehrere Entscheidungen treffen. Dazu zählen einerseits die Abgrenzung des zu analysierenden Systems vom Systemumfeld und die Modellierung der Abhängigkeiten innerhalb des Systems als auch die Auswahl geeigneter statistischer Mittel für die Berücksichtigung von Unsicherheitsfaktoren, inklusive einer geeigneten Parametrierung. Das Simulationsmodell soll dazu verwendet werden, das Systemverhalten unter bestimmten Rahmenbedingungen zu beobachten. Aus diesen Beobachtungen sollen sich Schlussfolgerungen ableiten lassen, unter welchen Bedingungen jeweils ein zweckmäßig günstiges oder ungünstiges Verhalten des realen Systems zu erwarten ist. Um auf diese Erkenntnisse vertrauen zu können, ist ein *Validierungs- und Verifikationsprozess* erforderlich [22], der die *Korrektheit* des Simulationsmodells aus zumindest zwei Perspektiven analysiert. Die Abgrenzung von Validierung zu Verifikation lässt sich im aktuellen Kontext wie folgt beschreiben (nach [22], ISO DIN 9001:2008 [81] und ANSI/IEEE-Std 1012-2016 [80]):

*Korrektheit des Simulationsmodells*

*Verifikation*

- Die *Verifikation* beschäftigt sich mit der Prüfung, ob das *Modell korrekt* konstruiert wurde. Hierbei spielt besonders die Richtigkeit der Abbildung vom konzeptuellen Modell auf das Simulationsmodell eine zentrale Rolle. Auch die Korrektheit der Eingabeparamater ist Gegenstand der Verifikation des Simulationsmodells.

*Validierung*

- Die *Validierung* beschäftigt sich mit der Prüfung, ob das *korrekte Modell* konstruiert wurde. Zentraler Gegenstand ist folglich die Analyse, wie akkurat das reale System im Simulationsmodell abgebildet ist. Diese Analyse wird meist auf Basis des Vergleichs des realen System- mit dem Modellverhalten während der Simulation umgesetzt.

Abbildung 3.7 gibt einen Überblick über die Einbettung von Verifikation und Validierung in den Modellierungsprozess der Ereignisorientierten Simulation.

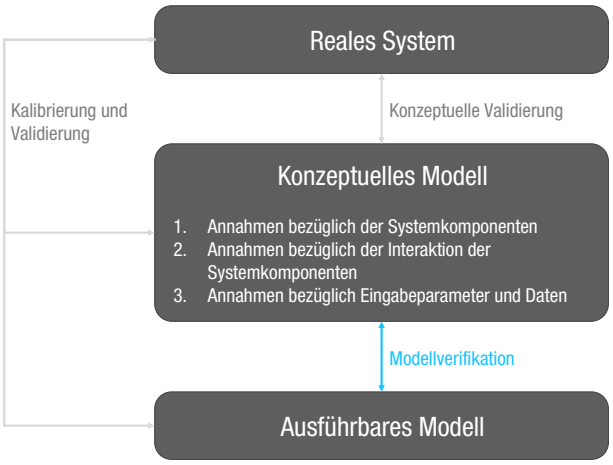



Abbildung 3.7: Modellierungsprozess der Ereignisorientierten Simulation (Quelle: [22])

*Modellierungsprozess des Simulationsmodells*

Der *Modellierungsprozess des Simulationsmodells* besteht im Wesentlichen aus drei Schritten. Im ersten Schritt wird das reale System beobachtet, wobei Informationen über die miteinander interagierenden Komponenten und deren Datenaustausch erfasst werden. Diese Beobachtungen werden üblicherweise mittels Interviews und Diskussionsrunden mit Menschen, die mit dem System vertraut sind, aufgenommen oder ergänzt. Im zweiten Schritt wird ein konzeptuelles Modell erstellt, welches im Wesentlichen eine Sammlung von Annahmen über Komponenten, Struktur und Eingabeparametrierungen des realen Systems beinhaltet. Aus diesem konzeptuellen Modell wird im dritten und letzten Schritt ein ausführbares Modell abgeleitet, welches dann in der Regel in einem Format vorliegt, welches für ein konkretes Simulationssystem spezifisch ist. Damit sind starke Parallelen zu den ersten drei Schritten des Prozesslebenszyklus (Abbildung 1.1) zu erkennen. Hier sind die Ergebnisse der einzelnen Schritte Ansatzpunkte eines iterativen Validierungs- und Verifikationsprozesses.

Da sich die Verwendung der Ereignisorientierten Simulation in der vorliegenden Arbeit auf die Generierung beispielhafter Prozessausführungsspuren für ein gegebenes Prozessmodell bezieht, beschränkt sich der aktuelle Abschnitt auf die Methoden, die sicherstellen, dass die Abbildung des konzeptuellen auf ein ausführbares Modell korrekt ist. Grundlegend bedeutet dies, dass sich das konzeptuell und meist abstrakt modellierte Systemverhalten im operativen Verhalten des ausführbaren Simulationsmodells widerspiegeln muss. Wie in Abbildung 3.7 dargestellt ist, handelt es sich bei diesen Untersuchungen um den Verifikationsteil des Prüfprozesses.

*Allgemeine Empfehlungen für Bestandteile der Verifikation*

Für den Verifikationsprozess werden in [22] acht allgemeine Vorschläge angegeben. Da für die vorliegende Arbeit immer ein bereits existierendes konzeptuelles Prozessmodell als Ausgangspunkt dient, wird folglich davon ausgegangen, dass dieses das reale System zweckgemäß und im gewünschten Umfang akkurat abbildet. Damit reduzieren sich die anwendbaren Vorschläge auf die folgenden:

1. Kontrolle des Modells durch einen Experten für die verwendete Simulationssoftware,
2. Untersuchen der Ausgaben des Modells in Abhängigkeit von den Eingabeparametern,
3. Ausgabe der Eingabeparameter am Ende der Simulation, um eine Veränderung gegenüber der ursprünglichen Eingabe auszuschließen,
4. Detaillierte Dokumentation jedes Bestandteils des Simulationsmodells,
5. Bereitstellen eines interaktiven *Debuggers*, um Fehler während der Simulationsdurchführung zu erkennen,
6. Bereitstellung einer grafischen Schnittstelle für das Simulationsmodell.

Diese Empfehlungen sind einerseits allgemein genug, um auch in der Domäne der Geschäftsprozessmodellierung Gültigkeit zu besitzen. Andererseits sind sie *zu* allgemein, um von Verifikations*methoden* zu sprechen. Aus diesem Grund werden diese Empfehlungen im Evaluationsteil der vorliegenden Arbeit erneut aufgegriffen und auf die Prozessdomäne angepasst.

In diesem Abschnitt wird der Validierungs- und Verifikationsprozess für Simulationsmodelle in groben Zügen vorgestellt. Die vorliegende Arbeit beschränkt sich dabei auf den Verifikationsteil, welcher die Überprüfung der Korrektheit der Abbildung zwischen konzeptuellem und ausführbarem Modell beinhaltet. Eine wesentliche Erkenntnis aus der Literatur ist, dass für die Verifikation Ereignisorientierter Simulationsmodelle vergleichsweise wenige konkrete, statistische Verfahren bekannt sind.

### 3.2.5 *Ereignisbasierte Simulation im Prozesskontext*

Dieser Abschnitt beschreibt die Simulation von imperativ oder deklarativ modellierten Prozessen auf Basis des Prinzips der Ereignisorientierten Simulation. Aus der Prozessdomäne wird hier vor allem eine Vorgehensweise zur Erzeugung von Simulationsmodellen sowie die Rolle von Zufallsvariablen für die Approximation des realen Prozessverhaltens diskutiert. Die allgemeine Vorgehensweise wird dabei auf die Generierung künstlicher Prozessausführungsspuren ausgerichtet, die einerseits zur Erfüllung von Anforderung A2 und andererseits von dem in Kapitel 2 skizzierten induktiven Translationsansatz für Prozessmodelle benötigt werden. Damit gehören Teile der nachfolgenden Beschreibung zum Beitrag der vorliegenden Arbeit. Diese Teile basieren auf den Grundlagen der Prozesssimulation nach [2]. Zur Abgrenzung von diesen grundlegenden Simulationskonzepten werden Simulationsansätze, welche auf die Generierung von Prozessausführungsspuren fokussiert sind, in allen nachfolgenden Abschnitten und Kapiteln *Spurgeneratoren* genannt.

*Spurgeneratoren*

Die Simulation von Prozessen unter der Nutzung von Rechnersystemen ist bereits im Zweiten Weltkrieg in Form der *Monte-Carlo-Simulation* bekannt geworden. Auch wenn die analytische Fragestellung sich auf die Auswirkungen atomarer Explosionen bezog, gab es starke Parallelen zur statistischen Simulation der

*Monte-Carlo-Simulation*

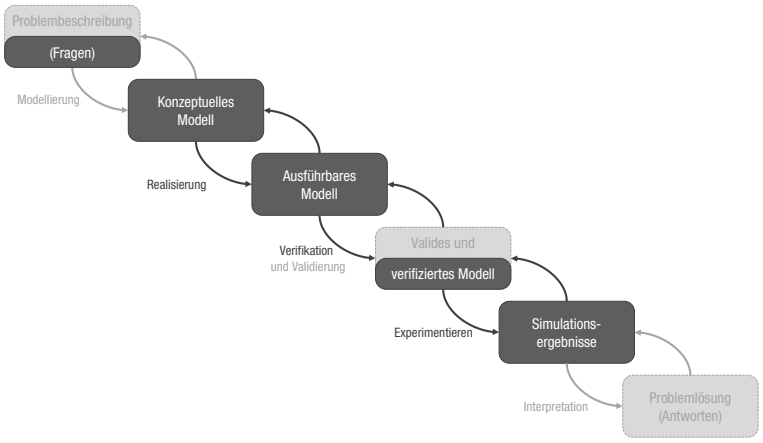



Abbildung 3.8: Allgemeine Phasen der Prozesssimulation (basierend auf: [2])

Gewinnchancen bei bestimmten Kartenspielen und unter bestimmten Rahmenparametrierungen. Später fanden sich im Finanzbereich, in der Telekommunikation, der Logistik und der Organisation von Arbeitsabläufen weitere Anwendungen für die Monte-Carlo-Simulation. Daher ist es nicht verwunderlich, dass die Simulation auch eine der am weitesten verbreiteten und am besten unterstützten Analysetechniken im Bereich des Geschäftsprozessmanagements darstellt [2].

Die Simulation von Prozessmodellen ist traditionell ein Instrument der modellbasierten Analyse in der Designphase des Prozesslebenszyklus (Abbildung 1.1). Das in Abbildung 3.8 dargestellte allgemeine Vorgehen zur Simulation von Prozessen umfasst sechs Phasen und erinnert an ein um iterative Elemente erweitertes Wasserfallmodell. Das Originalmodell aus [2] wird modifiziert, um dem Ziel der Spurgenerierung zu entsprechen.

*Vorgehensmodell Prozesssimulation*

Beginnend mit der *Problembeschreibungsphase* werden zunächst die Ziele und der Untersuchungsrahmen der Simulationsstudie festgelegt. Dies definiert einerseits die Inhalte des *Simulationsmodells* und andererseits die Fragestellungen, die mit selbigem beantwortet werden sollen. In der vorliegenden Arbeit ist das Ziel der Simulation die Erzeugung einer Menge beispielhafter Prozessausführungspuren. Fragestellungen können hier beispielsweise in folgender Form auftreten: „Welche möglichen Pfade durch den Prozess verbleiben, wenn mit den Aktivitäten X, Y und Z begonnen wird." Die generierten Prozessausführungsspuren können dann beispielsweise zum Verbessern des Prozessverständnisses eingesetzt werden (vgl.A2).

*Problembeschreibungsphase*

In der sich anschließenden *Modellierungsphase* wird ein konzeptuelles Modell erzeugt, welches *Entitäten* und *Relationen* zwischen diesen beinhaltet. Das sind in dem bereits früher beschriebenen Dienstreiseprozess (Abschnitt 3.1.5) beispielsweise Mitarbeiter aber auch der Antrag. Ein konzeptuelles Prozessmodell kann hier entweder direkt als Simulationsmodell verwendet werden oder zumindest als Grundlage zu dessen Erstellung dienen. Meist sind von der Prozessabbildung unabhängige Parameter – wie beispielsweise die Anzahl der Ressourcen pro Rolle – nicht Teil des konzeptuellen Prozessmodells. Diese Prozessparameter müssen für die Simulation ebenfalls bestimmt und mit entsprechenden Werten belegt werden. Zusätzlich zum Prozess selbst muss auch die Umgebung des modellierten Systems – hier die Umgebung eines Prozesses – für den gewählten Untersu-

*Modellierungsphase*

chungsrahmen modelliert werden (Abschnitt 3.2.1). Zur Prozessumgebung gehören hier beispielsweise Ankunftsraten für neue Prozessinstanzen und die Frequenz des Auftretens exogener Ereignisse. Auch die zu erfassenden *Kennzahlen* (engl. *Key Performance Indicator (KPI)*) müssen festgelegt werden. Sie dienen als Grundlage für die Beantwortung der im vorherigen Schritt identifizierten Fragestellungen. Eine häufig analysierte KPI ist die durchschnittliche Durchlaufzeit durch Prozessinstanzen. Grundsätzlich kann die Modellierungsphase Widersprüche in der Problemdefinition und weitere interessante Fragestellungen aufdecken, weswegen oft eine Wiederholung der Problembeschreibungsphase notwendig ist.

*KPIs*

Spurgeneratoren gehen von einem existierenden konzeptuellen Prozessmodell aus. Folglich entfällt der Teil der Modellierungsphase, in welcher der zu simulierende Prozess beschrieben wird. Die Parameter des Prozesses müssen hingegen auch in diesem Verwendungskontext modelliert werden. In [2] wird eine obligatorische Grundmenge an Konfigurationsparametern vorgeschlagen. Diese werden im späteren Verlauf des aktuellen Abschnitts kurz vorgestellt und es wird begründet, warum Parameter der Prozessumgebung in diesem Kontext irrelevant sind.

*Realisierungsphase*

Die *Realisierungsphase* überführt ein konzeptuelles in ein ausführbares Simulationsmodell und ähnelt folglich der Konfigurations- und Implementierungsphase des Prozesslebenszyklus (Abbildung 1.1). Je nach gewählter Simulationssoftware werden ausführbare Modelle auf unterschiedliche Art abgeleitet. Entweder erfolgt die Definition des konzeptuellen Modells und seiner Implementierung im selben Arbeitsschritt oder die Software ist selbst in der Lage, aus dem konzeptuellen Modell eine ausführbare Form abzuleiten. In jedem Fall ist abschließend eine korrekte Parametrierung des ausführbaren Modells und gegebenenfalls auch der Simulationssoftware notwendig. Diese wird üblicherweise ebenfalls aus dem konzeptuellen Simulationsmodell abgeleitet. Die Realisierungsphase ist im klassischen Simulationskontext, besonders aber für die Generierung künstlicher Prozessausführungsspuren von großer Bedeutung.

Wie bereits in Abschnitt 3.2.4 dargelegt, ist die Erstellung von ausführbaren Simulationsmodellen potentiell fehlerbehaftet. Konsequenterweise schließt sich auch in der Prozesssimulation eine Validierungs- und Verifikationsphase an. Die Verifikation prüft dabei die *intrinsische* und die Validierung die *extrinsische* Korrektheit des Modells. Innerhalb der Verifikation wird also analysiert, in wie fern das Modell eine in sich fehlerhafte „Programmierung" oder widersprüchliche Parametrierung aufweist. Auch die Prüfung der Robustheit bei der Simulation von Extremsituationen ist Teil der Verifikation. Dazu können beispielsweise sogenannte Stresstests verwendet werden. Im Validierungsteil wird hingegen das Verhalten des ausführbaren Simulationsmodells mit dem Verhalten des realen Prozesses – beispielsweise auf Basis historischer Prozessaufzeichnungen – verglichen. Da für die Generierung künstlicher Ausführungsspuren ein – in der Annahme korrektes – konzeptuelles Prozessmodell als Ausgangspunkt dient, entfällt hier eine Validierung. Es wird demnach von einem extrinsisch korrekten Prozessmodell ausgegangen. Daher beinhaltet diese Phase ausschließlich die Verifikation der Abbildung des konzeptuellen Prozessmodells und seiner Parametrierung auf eine ausführbare Repräsentation. So identifizierte modellinterne Fehler und Unstimmigkeiten können eine Anpassung des konzeptuellen Modells und sogar der Problemdefinition erfordern.

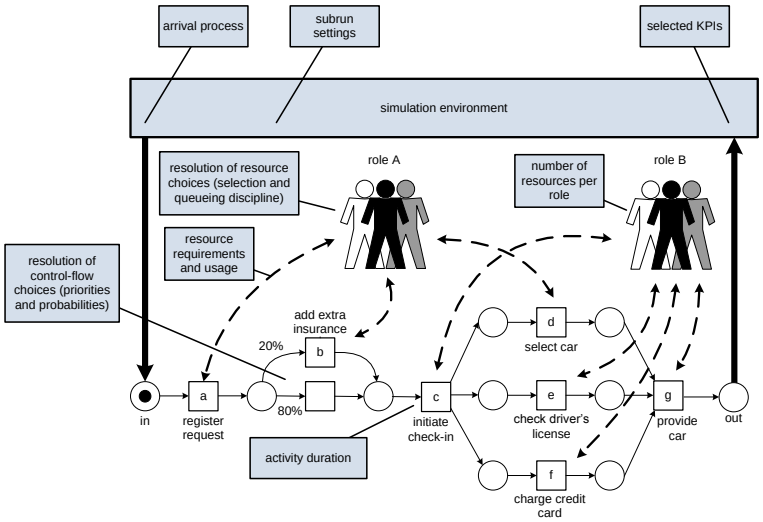



Abbildung 3.9: Parameter der Prozesssimulation (Quelle: [2])

Nachdem das ausführbare Simulationsmodell schließlich verifiziert und im Kontext der Spurgenerierung per Festlegung als valide eingestuft ist, können in der *Experimentierphase* Messungen des Prozessverhaltens für die gewählte Parametrierung vorgenommen werden. Um so effizient wie möglich vertrauenswürdige Simulationsergebnisse zu erhalten, muss hier die notwendige Anzahl an Simulationsdurchläufen sorgfältig abgewogen werden. Die Durchführung der Simulation beinhaltet (Abschnitt 3.2.2) die Generierung einer künstlichen Ausführungshistorie des jeweiligen Prozesses. Länge und Anzahl der darin enthaltenen Prozessausführungsspuren müssen demnach auch in diesem Kontext festgelegt werden.

*Experimentierphase*

In der *Interpretationsphase* erfolgt die Beurteilung der Simulationsergebnisse, die in Form von Messwerten für die gewählten KPIs vorliegen. Da diese Messungen immer von der jeweiligen Parametrierung des Modells und der Anzahl der Simulationsdurchläufe abhängig sind, muss für jede KPI ein Konfidenzintervall angegeben werden. Weiterhin dienen auch die KPIs meist nur als Grundlage für eine anschließende Beantwortung der in der Problemdefinitionsphase identifizierten Fragestellungen. Mit den Konfidenzintervallen kann zudem angegeben werden, als wie vertrauenswürdig die jeweiligen Antworten einzuschätzen sind. Das Ergebnis der Interpretationsphase ist in klassischen Simulationsszenarien ein Bericht mit auf die Fragestellungen bezogenen Statistiken und Analyseergebnissen.

*Interpretationsphase*

Für die Spurgenerierung entfällt diese Phase vollständig, da die gewünschten Ergebnisse hier die aufgezeichneten Prozessausführungsspuren selbst sind. Deren Interpretation ist nicht mehr Teil der Simulation, kann aber beispielsweise mittels geeigneter Process-Mining-Werkzeuge nachträglich vorgenommen werden.

In der Realisierungsphase ist die Übertragung der Simulationsparametrierung auf das ausführbare Modell und auf die Simulations-Software eine der Hauptaufgaben. Alle bisher genannten Parameter sind relevant. In [2] wird die in Abbildung 3.9 dargestellte Menge an Simulationsparametern als obligatorisch eingestuft. Es wird aber auch argumentiert, dass diese Parameter in der Regel nicht ausreichen. Im Folgenden werden diese Parameter sowohl kurz beschrieben, als auch hinsichtlich ihrer Relevanz im Kontext der Spurgeneratoren bewertet. Ein großer

Teil der Parameter repräsentiert dynamische, teilweise nicht vorhersehbare und daher nicht statisch modellierbare Zusammenhänge der Realität. Die Belegung dieser Parameter zum Simulationszeitpunkt erfolgt daher meist auf Basis eines Zufallszahlengenerators, welcher Werte entsprechend einer gewünschten Wahrscheinlichkeitsverteilung erzeugt (Tabelle 3.3). Die in Abbildung 3.9 dargestellten Parameter, die auch im Kontext der Generierung künstlicher Ereignisprotokolle relevant sind, werden in [2] wie folgt beschrieben:

*Relevante Simulationsparameter für Spurgeneratoren*

- Die *Auflösung von Kontrollflussentscheidungen* basiert im realen Prozess auf Entscheidungen der Prozessbeteiligten, datenbasiert oder auf Basis einer Kombination aus beidem. Diese Entscheidungen werden in der Prozesssimulation, wie auch allgemein in der Ereignisorientierten Simulation, mittels Wahrscheinlichkeitsverteilungen modelliert.

- Die *Anzahl der Ressourcen pro Rolle* entscheidet vor allem über die Zeit von der Ankunft eines neuen Falls bis zur Beendigung der zugehörigen Prozessinstanz, in Abhängigkeit von der Gesamtmenge der zum jeweiligen Zeitpunkt aktiven Instanzen. Sind alle Ressourcen belegt, wird der neue Fall in eine ebenfalls zu modellierende Warteschlange eingereicht.

*Auflösung von Entscheidungen*

Die Auflösung von Kontrollflussentscheidungen ist für Spurgeneratoren von zentraler Bedeutung. Um mit der Simulationsausführung alle möglichen Pfade durch das Prozessmodell zu erschließen, müssen die Entscheidungspunkte so konfiguriert sein, dass jede Aktivität des Prozesses mit einer Wahrscheinlichkeit von $> 0$ erreicht werden kann. Anzumerken ist hier, dass dieser Parameter ausschließlich für Prozessmodelle, welche auf gerichteten Graphen basieren, sowie daraus resultierende Simulationsmodelle relevant ist. Für die in [200] vorgestellte deklarative Modellierungssprache DPIL und das in Abschnitt 4.2 vorgestellte Simulationsverfahren ist er demzufolge irrelevant.

*Festlegung der Ressourcenanzahl*

Auch die Festlegung der Anzahl der Ressourcen pro Rolle ist wichtig, um während der Simulation zwischen den zwei folgenden Zusammenhängen im Prozessmodell unterscheiden zu können. Existiert pro Rolle nur eine Ressource, dann kann nicht zwischen der direkten Zuweisung einer Aktivität zu einer konkreten Ressource und der Zuweisung auf Basis der Rolle der Ressource unterschieden werden. Dieser Unterschied muss sich jedoch auch in den künstlich erzeugten Prozessausführungsspuren widerspiegeln.

*Irrelevante Simulationsparameter für Spurgeneratoren*

Die folgenden Parameter sind für traditionelle Simulationsprojekte notwendig, haben aber keine Relevanz für die in dieser Arbeit fokussierten Spurgeneratoren:

- Das Modell des *Ankunftsprozesses* beinhaltet Informationen über die Ankunftsrate neuer Fälle, für die jeweils eine neue Prozessinstanz startet.

- *Teilausführungen* beschreiben die Mehrfachsimulation eines Prozessausschnitts. Die Anzahl dieser Iterationen bestimmt die Präzision des Konfidenzintervalls hinsichtlich Vertrauenswürdigkeit der Messung der KPIs.

- Die *KPI-Auswahl* wird an den spezifischen Fragestellungen des konkreten Simulationsproblems ausgerichtet. Ein Beispiel für einen häufig bestimmten KPI ist die mittlere Durchlaufzeit einer Prozessinstanz.

- Die *Aktivitätsdauer* bestimmt den Abstand zwischen Beginn und Ende der Durchführung einer einzelnen Aktivität.
- Aktivitäten können Anforderungen stellen, die eine Prüfung der *Ressourceneignung* ermöglichen.
- Für die Zuweisung einer Aktivität zu einer Ressource ist eine *Auflösung der Ressourcenzuteilung* notwendig. Falls mehrere Ressourcen für die Bearbeitung der jeweiligen Aktivität geeignet sind, muss die Auswahl zusätzlich durch Regeln oder auf statistischer Basis modelliert werden.

*Prozessumgebung nicht im Fokus der Spurgeneratoren*

Parameter der Prozessumgebung sind für den Fokus der vorliegenden Arbeit grundsätzlich irrelevant. Der Hintergrund ist, dass die künstlich generierten Spuren in den beiden identifizierten Anwendungsfeldern exemplarisch Zusammenhänge im Prozessmodell, nicht aber zwischen Prozess und Prozessumgebung aufzeigen sollen. Zudem sind diese Parameter Einflussgrößen für die Ermittlung bestimmter KPIs. Ziel der Simulationsdurchführung im Kontext der vorliegenden Arbeit ist jedoch die Spurgenerierung auf Basis eines konzeptuellen Modells und nicht die Bewertung der Leistungsfähigkeit des Prozesses auf Basis von Kennzahlen. Folglich ist auch die Konfiguration des Ankunftsprozesses, von Teilausführungen und die Dauer der Aktivitäten irrelevant. Ohne eine Auswahl an entsprechenden KPIs besteht beispielsweise keine Notwendigkeit, die Aktivitätsdauer gezielt festzulegen. In [2] werden die Anforderungen an und der Auswahlprozess von Ressourcen als Simulationsparameter modelliert, da bestimmte Modellierungssprachen – wie beispielsweise Petri-Netze – nicht ausdrucksmächtig genug sind, um diese im konzeptuellen Prozessmodell festzulegen. In der vorliegenden Arbeit sollen jedoch lediglich Zusammenhänge aus den künstlichen Ausführungsspuren erkennbar sein, die vom konzeptuellen Prozessmodell vorgegeben sind. Aus diesem Grund sind die zugehörigen Parameter für den aktuellen Kontext irrelevant.

*Allgemeiner Mangel an Spurgeneratoren*

Grundsätzlich sind die wenigsten existierenden Simulatoren reine Spurgeneratoren. Aus Untersuchungen im Rahmen unterschiedlicher Projekte mit Industriepartnern hat sich herausgestellt, dass der Großteil der verwendeten Simulatoren hauptsächlich den traditionellen Simulationsansatz unterstützt und vergleichsweise wenige eine Ausleitung der intern erzeugten, künstlichen Prozessausführungsspuren erlauben. Für die im Kontext dieser Arbeit betrachteten Sprachen stehen jedoch entsprechende Werkzeuge zur Verfügung, welche im Implementierungsteil beschrieben werden. In einigen Fällen müssen für die Verwendung der jeweiligen Implementierung Parameter konfiguriert werden, die im aktuellen Abschnitt für Spurgeneratoren als irrelevant klassifiziert werden.

Eine wesentliche Erkenntnis von Abschnitt 3.2 ist, dass für ein Simulationsvorhaben nicht nur der zu betrachtende Prozess selbst, sondern auch dessen Umgebung modelliert werden muss. Weiterhin müssen komplexe Abhängigkeiten, wie beispielsweise menschliche Entscheidungen, im Simulator nachgebildet werden. Sowohl dafür als auch zur Modellierung der Prozessumgebung werden üblicherweise Zufallsvariablen verwendet. Für den in der vorliegenden Arbeit beschriebenen Spurgenerator (Abschnitt 4.2) ist die Prozessumgebung jedoch zweitrangig, da diese primär die Leistungsfähigkeit, nicht aber die möglichen Ausführungsspuren eines Prozesses beeinflussen.

## 3.3 EREIGNISPROTOKOLLE FÜR DIE AUSFÜHRUNG VON PROZESSEN

*Ereignisprotokolle*

*Ereignisprotokolle* (engl. *event logs*) sind Aufzeichnungen von Prozessausführungen. Sie können historisch oder künstlich sein. In ersterem Fall werden die Protokolle durch ein oder mehrere IT-Systeme erzeugt, die in die Prozessausführung eingebunden sind oder diese sogar steuern. Künstliche Protokolle werden dagegen entweder manuell zusammengestellt oder, mittels der im vorherigen Abschnitt beschriebenen Simulationstechniken, automatisiert generiert. Die vorliegende Arbeit beschäftigt sich ausschließlich mit letzterem Fall. Ein grundlegendes Verständnis des Konzepts der Ereignisprotokolle ist einerseits für die Interpretation der Ergebnisse des später vorgestellten Spurgenerators notwendig. Andererseits dienen sie auch als Transfermedium bei dem ebenfalls später diskutierten Translationsprinzip für Prozessmodelle.

Zunächst wird, für ein besseres Verständnis der Protokollthematik der Zusammenhang zwischen Prozessausführungsspuren und Ereignisprotokollen beschrieben. Anschließend wird die Gewinnung derartiger Protokolle aus produktiven Datenquellen und auf künstlicher Basis verglichen. Eine wesentliche Erkenntnis aus diesem Vergleich ist, dass sich künstliche Ereignisprotokolle für die Verwendung im Translationskontext besser eigenen. Danach werden die Ereignisprotokolle formal definiert. Diese Definition dient als Grundlage für die Spezifikation des Meta-Modells für Prozessausführungsspuren im Rahmen der später vorgestellten Technik zur Spurgenerierung. In Abschnitt 3.3.4 wird beschrieben, wie Ereignisprotokolle in strukturierter und maschinenlesbarer Form abgelegt werden können. Das dort diskutierte Format muss für die praktische Anwendung der im Rahmen dieser Arbeit entwickelten Simulations- und Translationstechniken geläufig sein.

### 3.3.1 *Prozessausführungsspuren und Ereignisprotokolle*

Prozessausführungsspuren und Ereignisprotokolle dokumentieren die Ausführung eines Prozesses. Ein Ausschnitt aus einem exemplarischen Ereignisprotokoll ist in Tabelle 3.4 dargestellt. Das Protokoll enthält Informationen aus unterschiedlichen Perspektiven (Beschreibung der Perspektiven in Abschnitt 1.1.2) eines realen, universitären Prozesses zur Abwicklung von Dienstreisen. Der Prozess entspricht weitestgehend den Vorgaben der Modelle aus Abschnitt 3.1.6. Jedes Ereignisprotokoll muss eindeutig einem bestimmten Prozess zugeordnet werden können, was jedoch nicht in der Tabelle dargestellt ist. Da in der vorliegenden Arbeit jedoch ausschließlich künstliche, mittels Simulation erzeugte Protokolle betrachtet werden, können über das entsprechende Simulationsmodell Rückschlüsse auf den zugrundeliegenden Prozess gezogen werden.

*Ereignisprotokoll beschreibt genau einen Prozess*

Ein Ereignisprotokoll beinhaltet Ereignisse (Zeilen in Tabelle 3.4), wobei sich jedes auf die Ausführung genau einer *Aktivität* bezieht. Jedes der Ereignisse ist genau einer *Prozessinstanz* – in der Literatur auch als *Fall* (engl. *case*) bezeichnet – zugeordnet. Wie auch aus der ersten Spalte von links in Tabelle 3.4 ersichtlich wird, werden die einzelnen Fälle anhand eindeutiger Bezeichner (engl. *case ID*) unterschieden. Somit sind in der Tabelle die Abwicklungen zweier unterschiedlicher Dienstreisen auszugsweise festgehalten. Aktivitäten erhalten ebenfalls einen

*Prozessinstanz*

| Fall | Zeitstempel | Ereignistyp | Aktivität | Ressource | Daten | ... |
|---|---|---|---|---|---|---|
| LA-Swed2015 | 2015-05-08T09:18 | start | Informieren | LA | - | ... |
| | 2015-05-08T09:47 | complete | Informieren | LA | Kostenvoranschlag = 1200 | ... |
| | 2015-05-08T10:01 | start | Antrag stellen | LA | - | ... |
| | 2015-05-08T10:13 | complete | Antrag stellen | LA | Antragsstatus = geschrieben, Antrag = /SJ-Rhodos2016.pdf | ... |
| | 2015-05-08T12:47 | start | Antrag prüfen | KH | | ... |
| | 2015-05-08T13:08 | complete | Antrag prüfen | KH | Antragsstatus = geprüft | ... |
| | 2015-05-10T10:00 | start | Antrag genehmigen | DS | - | ... |
| | 2015-05-10T10:22 | complete | Antrag genehmigen | DS | Antragsstatus = genehmigt | ... |
| | 2015-05-11T13:58 | start | Unterkunft buchen | LA | - | ... |
| | 2015-05-11T14:30 | complete | Unterkunft buchen | LA | U-Kosten = 420 | ... |
| | 2015-05-11T09:33 | start | Reisedokumente zusammenstellen | KH | - | ... |
| | 2015-05-11T09:50 | complete | Reisedokumente zusammenstellen | KH | Antragsstatus = archiviert, Fixkosten = 1147 | ... |
| SJ-Rhodos2016 | 2016-10-13T15:34 | start | Informieren | SJ | - | ... |
| | 2016-10-13T15:51 | complete | Informieren | SJ | Kostenvoranschlag = 1500 | ... |
| | 2016-10-14T09:08 | start | Antrag stellen | SJ | - | ... |
| | 2016-10-14T09:20 | complete | Antrag stellen | SJ | Antragsstatus = geschrieben | ... |
| | 2016-10-14T12:50 | start | Antrag prüfen | KH | | ... |
| | 2016-10-14T13:15 | complete | Antrag prüfen | KH | Antragsstatus = geprüft | ... |
| | 2016-10-15T07:40 | start | Unterkunft buchen | SJ | | ... |
| | 2016-10-15T07:55 | complete | Unterkunft buchen | SJ | U-Kosten = 480 | ... |
| | 2016-10-16T13:59 | start | Antrag genehmigen | DS | - | ... |
| | 2016-10-16T14:25 | complete | Antrag genehmigen | DS | Antragsstatus = genehmigt | ... |
| | 2016-10-17T10:12 | start | Reisedokumente zusammenstellen | KH | - | ... |
| | 2016-10-17T10:24 | complete | Reisedokumente zusammenstellen | KH | Antragsstatus = archiviert, Fixkosten = 1420 | ... |
| ... | ... | ... | ... | ... | ... | ... |

Tabelle 3.4: Ereignisprotokoll für realen Prozess der Dienstreiseabwicklung

eindeutigen Bezeichner, um einzelne Schritte des Prozesses voneinander unterscheiden zu können. Auch für Ereignisse kann es wichtig sein, sie anhand von IDs voneinander zu unterscheiden. Zwar ist auch ein Zeitstempel[4] ein typisches Attribut von Ereignissen in Ereignisprotokollen, wenn ein Ausführungssystem aber *Nebenläufigkeit*, also die parallele Bearbeitung von Aktivitäten erlaubt, ist dieser Zeitstempel nicht immer eindeutig. Das wird dann zum Problem, wenn zwischen Start und Beendigung einer Aktivität unterschieden wird und Aktivitäten zusätzlich wiederholt werden können. Grund ist, dass mit diesen Möglichkeiten eine Aufgabe mehrfach begonnen werden kann, bevor sie einmal beendet ist. Damit können Ereignisse, welche dieselbe Aktivität betreffen, auch mehrfach pro Fall und verschränkt im Ereignisprotokoll auftreten. Das ist in Tabelle 3.4 nicht berücksichtigt, weswegen hier von rein sequentieller Ausführung ausgegangen werden müsste.

*Problematik der Nebenläufigkeit*

Unter *Prozessausführungsspur* ist die Aufzeichnung genau eines vollständigen Prozessdurchlaufs zu verstehen. Eine Prozessausführungsspur setzt sich aus den Aufzeichnungen eines oder mehrerer *Ereignisse* zusammen. Mehrere Spuren werden zu einem *Ereignisprotokoll* zusammengefasst.

*Prozessausführungsspur*

Um im späteren Verlauf die Funktionsweise der Simulations- und der Process-Mining-Techniken formalisieren zu können, müssen auch die Begriffe zur Beschreibung von Ereignisprotokollen formalisiert werden. Dafür wird auf eine Kombination[5] gängiger Notationen zurückgegriffen [17, 161, 176] (Definition 3.3).

4 Das Datumsformat der Zeitstempel in Tabelle 3.4 wird aus Gründen der Übersichtlichkeit vereinfacht.

5 In der Literatur werden Ereignisprotokolle meist im Process-Mining-Kontext formal definiert. Da es zum aktuellen Zeitpunkt kein Process-Mining-Verfahren gibt, welches neben der Funktionalen und

**Definition 3.3 (Ereignisprotokolle).** *Ein Ereignisprotokoll* $L$ *ist eine Multimenge (Index* $b$*) von Prozessausführungsspuren* $\sigma_i^L$*:*

$$L = \left\{ \sigma_1^L, \ldots, \sigma_{|L|}^L \right\}_b \tag{15}$$

*Jede Prozessausführungsspur ist dabei eine Sequenz von Ereignissen* $e_j$ *aus der Menge aller Ereignisse* $E$ *des Ereignisprotokolls:*

$$\sigma_i^L = \left\langle e_1, ..., e_{|\sigma_i^L|} \right\rangle \in L \text{ mit } 1 \leqslant i \leqslant |L| \tag{16}$$

*Diese Sequenz ist eine Totalordnung* $\succ$ *aller Ereignisse einer Ausführungsspur:*

$$\succ \subseteq E \times E \text{ und } \forall_{1 \leqslant k < l \leqslant |\sigma_i^L|} (e_l, e_k) \in \succ \tag{17}$$

*Ein Ereignis ist generell durch ein Tupel von Attribut-Wert-Paaren definiert:*

$$e = \left( (attr_1^e, val_1^e), ..., (attr_{|e|}^e, val_{|e|}^e) \right) \tag{18}$$

*Die Menge aller Attribute* $ATTR^e$ *und aller zugehöriger Attributwerte* $VAL^e$ *eines Ereignisses ist wie folgt definiert:*

$$ATTR^e = \left\{ attr_1^e, ..., attr_{|e|}^e \right\}, VAL^e = \left\{ val_1^e, ..., val_{|e|}^e \right\} \tag{19}$$

*Mit* $l^e(attr_i^e) = val_i^e$ *wird nachfolgend beschrieben, dass im Ereignis* $e$ *der Wert für Attribut* $attr_i^e$ *gleich* $val_i^e$ *ist. Jedes Ereignis muss mindestens einen eindeutigen Bezeichner* $id$*, eine eindeutige Zuordnung zu einer Prozessinstanz* $case$*, einen Typ* $type$*, den Namen der ausgeführten Aktivität* $activity$ *sowie einen Zeitstempel* $time$ *aufweisen:*

$$\forall_{e \in E} (id^e, case^e, type^e, activity^e, time^e) \subseteq ATTR^e \tag{20}$$

*Jedes Ereignis ist dabei genau einer Prozessinstanz zugeordnet:*

$$\forall_{e \in E} \left( l^e(case^e) = \sigma_i^L \right) \Leftrightarrow \left( e \in \sigma_i^L \right) \tag{21}$$

*Die Totalordnung der Ereignisse einer Prozessinstanz wird wie folgt realisiert:*

$$\forall_{1 \leqslant j < k \leqslant |\sigma_i^L|} l^{e_j}(time^{e_j}) < l^{e_k}(time^{e_k}) \tag{22}$$

*Alle weiteren Attribute beschreiben beliebige, optionale Datenwerte.*

*Ein Ereignisprotokoll ist redundanzfrei, falls jede der enthaltenen Spuren einzigartig ist. Das trifft zu, wenn sich jede Spur bezüglich der Totalordnung in mindestens einem Attribut eines Ereignisses von allen anderen Spuren unterscheidet:*

$$\forall \sigma_i^L, \sigma_j^L \in L; i \neq j \left| \exists e_m \in \sigma_i^L, e_n \in \sigma_j^L \right| (e_m \cap e_n) \neq e_m \vee (e_n \cap e_m) \neq e_n \tag{23}$$

Ziel des aktuellen Abschnitts ist die formale Definition von Ereignisprotokollen und darin enthaltener Prozessausführungsspuren. Dies dient als Grundlage für die spätere Beschreibung der Simulations- und Process-Mining-Verfahren.

der Verhaltensorientierten auch die Organisatorische *und* die Datenbezogene Perspektive behandelt, sind auch die einzelnen Definitionen der Ereignisprotokolle entsprechend unvollständig.

### 3.3.2 *Datenquellen für historische Ereignisprotokolle*

*Informationssysteme* haben die Aufgabe, Informationen zu beschaffen, zu verarbeiten und gezielt zu verteilen. Eine Unterklasse dieser Systeme, die sogenannten *Prozessbewussten Informationssysteme (PBIS)* (engl. *Process-Aware Information Systems*) [6, 60], unterscheidet sich dadurch von allgemeinen Informationssystemen, dass einerseits der Prozessgedanke den Informationsfluss beeinflusst und andererseits das System selbst Informationen über die Ausführung des zugrundeliegenden Prozesses sammelt. Damit lassen sich derartige Systeme klar von beispielsweise reinen Datenbanksystemen unterscheiden. Wiederum eine Subklasse der PBIS bilden *generische* PBIS, die explizit durch ein austauschbares Prozessmodell gesteuert werden. Beispiele dafür sind die bereits erwähnten *Workflow-* und *Geschäftsprozess-Management-Systeme*. Erstere orientieren sich mehr auf die Prozessautomatisierung, während letztere einen breiteren Fokus aufweisen und somit, neben der ebenfalls vorhandenen Prozessautomatisierung, zusätzlich einen großen Teil des in Abbildung 1.1 dargestellten Prozesslebenszyklus unterstützen. Auch ERP[6]- und CRM-Systeme[7] sowie Krankenhausinformationssysteme gehören zu den PBIS.

*Informationssysteme*

*Prozessbewusste Informationssysteme*

Alle aktuellen Lösungen aus den genannten Systemklassen protokollieren in meist heterogen strukturierter und verteilter, dafür aber digitaler und detaillierter Form die ausgeführten Aktivitäten [6, S. 32]. Das Spektrum möglicher Datenablagetypen reicht dabei von einfachen Textdateien und E-Mails über Tabellenkalkulationen bis hin zu Datenbanken. Damit ist in der Realität sehr häufig ein größerer Aufwand einzuplanen, um vollständige Ereignisprotokolle zusammenzustellen. Dies wird zusätzlich erschwert, wenn unternehmensübergreifende Prozesse protokolliert werden sollen. In diesem Fall müssen die potentiell unterschiedlichen Datenschemata der einzelnen Unternehmen aufeinander abgebildet werden. Besonders bei größeren Unternehmen häufen sich zudem riesige Datenmengen an. Damit die Vereinheitlichung und Integration der Daten in einem vertretbaren Rahmen bleibt, werden meist nicht alle Daten zusammengetragen, sondern es wird auf Basis einer klaren Definition des Analyseziels und klarer Fragestellungen eine Selektion vorgenommen. Dieses Vorgehen wird auch bereits im Bereich der *Business Intelligence* in sogenannten *Data-Warehouse-Anwendungen* angewendet. Die aufwendige Datenintegration wird dort im sogenannten *ETL-Prozess* vorgenommen [71, S. 93 ff.]. Dieser besteht aus den folgenden drei Schritten:

*Verteilte Ereignisprotokollierung*

*Vielfältige Formen der Datenablage*

*Zielorientierte Datenbeschaffung*

*Business Intelligence und Data Warehousing*

*ETL-Prozess*

1. *Extract:* Extraktion der Daten aus den verteilten und heterogenen Quellen,
2. *Transform:* Transformation der Daten in ein zielorientiertes Format,
3. *Load:* Einfügen der Daten in das Data Warehouse.

Der Aufwand alle Daten aufzubereiten und damit unterschiedliche Bezeichner, Datumsformate oder Datentypen zu vereinheitlichen sowie Duplikate zu identifizieren ist in den meisten Fällen immens [6, S. 143f]. Die Probleme reichen hier von

6 Systeme zur zweckmäßigen Ressourcenplanung in Unternehmen.

7 Systeme, die den Kundenkontakt durch Bereitstellung von Kundendaten und Interaktionshistorien erleichtert sowie Verknüpfungen zwischen Kundendaten und davon abhängigen, weiteren Systemen unterstützt.

vergleichsweise einfachen Vereinheitlichungen hinsichtlich der Groß- und Kleinschreibung über die Transformationen lokalisierter Zeitstempel in ein gemeinsames Datumsformat bis hin zum Ausgleich von Unterschieden in der Datengranularität. Ein Beispiel für letzteres ist die Vereinheitlichung der Ortsangaben für Sensoraufzeichnungen, bei denen in einer Datenquelle lediglich die Stadt und in einer zweiten Datenquelle konkrete geografische Koordinaten erfasst werden.

Die Datengrundlage kann auch unvollständig oder ungenau sein. Beispielsweise könnte eine Aktivität in der Realität durchgeführt worden sein, ohne, dass das Informationssystem diese Durchführung erfasste. Auch die gegenteilige Situation ist möglich, in der fälschlicherweise die Durchführung einer Aktivität verzeichnet wird, obwohl dies in der Realität nicht stattfindet. Weiterhin können auch bei vollständiger Erfassung aller Aktivitäten die Informationen *über* diese unvollständig, inkorrekt oder zu unpräzise sein [6, S. 148]. Letzteres tritt besonders in Form von zu ungenauen Zeitstempeln und anonymisierten, firmenkritischen Informationen auf. Derartige Probleme lassen sich durch eine Analyse der entsprechenden Ereignisprotokolle nicht ermitteln. Zwei weitere Klassen von Problemen bezüglich der Qualität von Ereignisprotokollen werden allgemein als *Rauschen* (engl. *noise*) oder als *Unvollständigkeit* (engl. *incompleteness*) bezeichnet. Als Rauschen wird in der Literatur protokolliertes Verhalten bezeichnet, welches selten auftritt und für das typische Prozessverhalten nicht repräsentativ ist. Dieser Begriff ist im allgemeinen Fall als fragwürdig anzusehen. Hintergrund ist, dass nicht *jedes* selten beobachtete Verhalten zwangsläufig als irrelevant eingestuft werden kann. Beispielsweise verlaufen Banktransaktionen durch zahlreiche Absicherungsmechanismen in der Regel fehlerfrei. Dennoch ist das Verhalten im Fehlerfall von immenser Wichtigkeit, um agil die nötigen Kompensationsprozesse einzuleiten[8]. Unvollständigkeit ist vom Prozess eigentlich gestattetes Verhalten, welches sich jedoch nie im Ereignisprotokoll manifestiert und dadurch nur ein Teil des Prozesses durch beispielhafte Instanzen beschrieben ist. Der Begriff Rauschen wird in der Literatur in vielen Fällen (z.B. [34]) stellvertretend für *beide* Problemklassen verwendet. Die vorliegende Arbeit bedient sich ebenfalls dieser Vereinfachung und übernimmt aus Gründen der Konsistenz zur einschlägigen Literatur den ansonsten fragwürdigen Begriff des Rauschens.

*Rauschen*
*Unvollständigkeit*

Ein weiterer, wesentlicher Nachteil realer Prozessaufzeichnungen ist die mangelnde Kontrolle über die Inhalte. Enthält ein Prozess selten durchzuführende Teilprozesse, dann sinkt die Wahrscheinlichkeit, dass dieser Teilprozess in einem realen Protokoll auftritt. Ein Beispiel dafür ist eine Produktionskette, die beim Ausfall einer bestimmten Maschine anders verläuft als im regulären Betrieb. Ist in der gesamten Zeit der Protokollaufzeichnung kein einziger Ausfall dieser Maschine aufgetreten, dann wird die alternative Produktionskette nie protokolliert. In der vorliegenden Arbeit wird diese Problematik vollständig vermieden, da alle Ereignisprotokolle mittels Prozesssimulation erzeugt werden.

8 Dass derartige Fehler auch mit heutigen Technologien auftreten können, beweist die *Bank of the Philippine Islands*, die am 07.06.2017 einen internen Datenverarbeitungsfehler meldete (https://www.bloomberg.com/news/articles/2017-06-07/account-error-at-big-philippine-bank-causes-social-media-storm, abgerufen am: 31.07.2017).

### 3.3.3 *Datenquellen für künstliche Ereignisprotokolle*

Die im vorangegangenen Abschnitt diskutierten historischen Ereignisprotokolle basieren auf Aufzeichnungen realer Prozessverläufe. Zusammenfassend ist zu sagen, dass die Qualität dieser Ereignisprotokolle stark von der Qualität der von den involvierten IT-Systemen erfassten Daten abhängig ist. Möchte man ein historisches Ereignisprotokoll, beispielsweise für die Evaluation von Process-Mining-Verfahren, verwenden, dann kann das Bewertungsergebnis durch Qualitätsprobleme verzerrt werden [41]. Somit sind *künstliche* Ereignisprotokolle unter anderem eine wertvolle Ergänzung zur Bewertung von Process-Mining-Techniken. Folgerichtig ist die Bereitstellung qualitativ hochwertiger Ereignisprotokolle eine der Schlüsselherausforderungen für die *IEEE Task Force on Process Mining* [182], die sich unter anderem mit Aufgaben der Qualitätssicherung für Ereignisprotokolle beschäftigt.

*Künstliche Ereignisprotokolle*

Künstliche Ereignisprotokolle können prinzipiell manuell erzeugt werden. Die Schwierigkeiten dabei liegen im zeitlichen Aufwand für die Anfertigung größerer Protokolle und in der Fehleranfälligkeit, wenn diese Protokolle auf Basis eines existierenden Prozessmodells erzeugt werden sollen. Besonders in letzterem Fall ist eine automatische Erzeugung durch *Simulation* des Prozessmodells naheliegend. In [22] wird der Kern des Simulationsproblems wie folgt beschrieben:

> "[...] simulation involves the generation of an artificial history of a system and the observation of that artificial history to draw inferences concerning the operating characteristics of the real system."

Damit können in Bezug auf die Simulation von Prozessen auf Basis konzeptueller Prozessmodelle zwei grundlegende Aufgaben voneinander getrennt werden. Die erste Aufgabe beinhaltet die Generierung einer künstlichen Historie von Prozessausführungen, während die zweite Aufgabe sich mit der Ableitung der Charakteristika des modellierten Systems auf Basis dieser Historie beschäftigt. Folglich ist die Ereignisorientierte Simulation durch ihre Übertragung in den Prozesskontext eine mögliche Quelle für künstliche Ereignisprotokolle.

*Simulation: Generierung einer künstlichen Historie*

### 3.3.4 *Serialisierungsformate*

Unabhängig davon wie ein Ereignisprotokoll gewonnen wird, ist für den Austausch und das Speichern ein standardisiertes Format notwendig. Bis vor wenigen Jahren war die XML-basierte Syntax der *Mining eXtensible Markup Language (MXML)* [58] der diesbezügliche Quasistandard. Mit diesem Format ist es möglich, Ereignisprotokolle, wie das in Tabelle 3.4 dargestellte, zu serialisieren. Ein wesentlicher Faktor für die intensive Nutzung in zahlreichen Applikationen ist die Möglichkeit der Erweiterung um beliebige Datenelemente zu Ereignissen und Prozessausführungsspuren. Diese Erweiterbarkeit erfordert jedoch eine individuelle Interpretation der als Erweiterung hinzugefügten Datenattribute. Jede dieser Erweiterungen besteht lediglich aus einem Schlüssel und einem zugehörigen Wert in Form einer Zeichenkette – ohne jede Typinformation. Dies führt zu zahlreichen Einschränkungen und Hindernissen in praktischen Anwendungen [6, S. 138].

*MXML*

Hinsichtlich der Standardisierung ist das 2010 von der *IEEE Task Force on Process Mining* eingeführte Format namens *eXtensible Event Stream (XES)* [70, 185]

*XES*

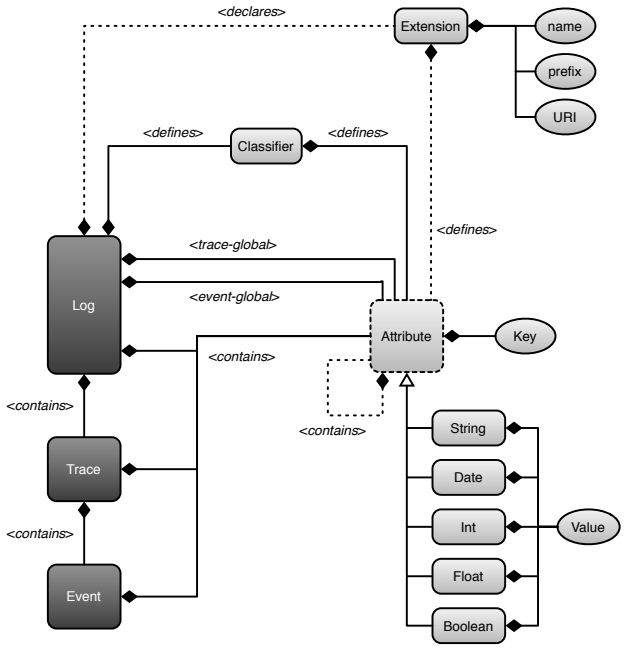



Abbildung 3.10: XES-Metamodell (Quelle: [185])

erfolgreicher. Derzeit wird das Format extensiv mit dem Ziel geprüft, den neuen Quasistandard zu einem offiziellen IEEE-Standard zu erklären. XES ist ebenfalls XML-basiert; das Metamodell ist in Form eines UML-Klassendiagramms in Abbildung 3.10 dargestellt. XES ist in der Lage, die formale Definition von Ereignisprotokollen aus Abschnitt 3.3.1 abzubilden. Ein XES-Dokument stellt genau ein Ereignisprotokoll (*Log*) dar und umfasst mehrere Prozessausführungsspuren (*Trace*), die aber alle Aufzeichnungen von Ausführungen des *selben* Prozesses darstellen. Jede dieser Ausführungsspuren beschreibt genau eine Prozessinstanz und besteht aus den sequentiellen Aufzeichnungen eines oder mehrerer Ereignisse (*Event*). Gemäß der formalen Definition eines Ereignisses besteht selbiges aus einer ungeordneten Menge an *Attributen*. Auch die übergeordnete Prozessausführungsspur und ihr wiederum übergeordnetes Ereignisprotokoll können Attribute aufweisen. Entgegen der MXML-Spezifikation ist jedes Attribut hinsichtlich eines konkreten Datentyps spezialisiert. Zu den unterstützten Datentypen gehören beispielsweise *String*, *Int*, *Date* aber auch komposite Typen wie *List*. Entgegen der formalen Spezifikation (Abschnitt 3.3.1) gibt XES keine verpflichtenden Attribute vor. Das Metamodell bietet jedoch die Möglichkeit für Erweiterungen (*Extension*), was eine semantische Zuordnung von Attributen ermöglicht. Dieser flexible Erweiterungsmechanismus wird von zwei Standarderweiterungen verwendet, der *Time*- und der *Organizational Extension*. Ersteres führt ein Zeitstempel-Attribut ein, während letzteres die Zuordnung der durchführenden organisatorischen Ressource zum jeweiligen Ereignis gestattet. Je nach Domäne, verfügbaren Informationen und Fokus der Protokollierung können beliebig zusätzliche Erweiterungen hinzugefügt werden. Um letztlich die formale Forderung nach Informationen wie Zeitstempel, eindeutigem Bezeichner der Prozessinstanz oder der ausgeführten Aktivität zu erfüllen, können Attribute im Kopf des XES-Dokuments als verpflichtend

*Log*
*Trace*
*Event*
*Attribute*
*Datentypen*
*Extension*

deklariert werden. Dafür wird im Metamodell mit *trace-global* und *event-global* eine Unterscheidung zwischen zwei *globalen Attribut-Kollektionen* getroffen. Ersteres beschreibt eine Liste an Attributen, die für alle Prozessausführungsspuren verpflichtend ist. Letzteres beschreibt verpflichtende Attribute für alle Ereignisse.

*Globale Attribute*

```
<log>
  <lifecycle:model type="LITERAL" value="standard" />
  <extension name="Concept" prefix="concept" uri="..." />
  <extension name="Time" prefix="time" uri="..." />
  <extension name="Organizational" prefix="org" uri="..." />
  <extension name="Lifecycle" prefix="lifecycle" uri="..." />
  <global scope="trace">
    <string key="concept:name" value="name" />
  </global>
  <global scope="event">
    <string key="concept:name" value="name" />
    <string key="org:resource" value="resource" />
    <string key="time:timestamp" value="2013-06-06T15:31:00.000+01:00" />
  </global>
  <trace>
    <string key="concept:name" value="SJ-Rhodos2016"/>
    <event>
      <string key="org:resource" value="SJ"/>
      <date key="time:timestamp" value="2016-10-13T15:34:00.000+01:00"/>
      <string key="concept:name" value="Informieren"/>
      <string key="lifecycle:transition" value="start"/>
    </event>
    <event>
      <string key="org:resource" value="SJ"/>
      <date key="time:timestamp" value="2016-10-13T15:51:00.000+01:00"/>
      <string key="concept:name" value="Informieren"/>
      <float key="Kostenvoranschlag" value="1500.0"/>
      <string key="lifecycle:transition" value="complete"/>
    </event>
    <event>
      <string key="org:resource" value="SJ"/>
      <date key="time:timestamp" value="2016-10-14T09:08:00.000+01:00"/>
      <string key="concept:name" value="Antrag stellen"/>
      <string key="lifecycle:transition" value="start"/>
    </event>
    <event>
      <string key="org:resource" value="SJ"/>
      <date key="time:timestamp" value="2016-10-14T09:20:00.000+01:00"/>
      <string key="concept:name" value="Antrag stellen"/>
      <string key="Antrag" value="/SJ-Rhodos2016.pdf" />
      <string key="Antragsstatus" value="geschrieben"/>
      <string key="lifecycle:transition" value="complete"/>
    </event>
    ...
    <event>
      <string key="org:resource" value="KH"/>
      <date key="time:timestamp" value="2016-10-17T10:12:00.000+01:00"/>
      <string key="concept:name" value="Reisedokumente zusammenstellen"/>
      <string key="lifecycle:transition" value="start"/>
    </event>
    <event>
      <string key="org:resource" value="KH"/>
      <date key="time:timestamp" value="2016-10-17T10:24:00.000+01:00"/>
      <string key="concept:name" value="Reisedokumente zusammenstellen"/>
      <string key="Antragsstatus" value="archiviert"/>
      <float key="Fixkosten" value="1420"/>
      <string key="lifecycle:transition" value="complete"/>
    </event>
  </trace>
```

Codeausschnitt 3.11: XES-Repräsentation des Ereignisprotokolls aus Tabelle 3.4

So wie Prozessmodellierungssprachen zur Beschreibung von Prozessen verwendet werden, ist XES eine domänenspezifische Sprache zur Beschreibung möglicher Prozess*instanzen*. Eine domänenspezifische Sprache setzt sich aus einer abstrakten und einer konkreten Syntax zusammen (Abschnitt 1.1.1). Mit Abbildung 3.10 ist lediglich die abstrakte Syntax der Sprache XES beschrieben. Theoretisch können dazu mehrere konkrete Syntaxen angegeben werden. Für Serialisierung und Deserialisierung eignet sich jedoch die am weitesten verbreitete, XML-basierte besonders gut. In Codeausschnitt 3.11 ist ein Auszug aus dem Dienstreise-Ereignisprotokolls (Tabelle 3.4) im gebräuchlichen XML-basierten Serialisierungsformat dargestellt.

*XML-basierte konkrete Syntax und Serialisierung*

*Vier Erweiterungen*

Das Beispiel nutzt den vom Metamodell angebotenen Erweiterungsmechanismus für die Einführung von vier speziellen Attributen. Die vier Erweiterungen sind entsprechend *Concept*, *Time*, *Organizational* und *Lifecycle*:

*Concept*

- Die Erweiterung *Concept* (Schlüsselpräfix: *concept*) führt das *name*-Attribut für Prozessausführungsspuren und Ereignisse ein. Im Falle der Spuren wird dieses Attribut meist mit einem eindeutigen Bezeichner für die jeweilige Prozessinstanz belegt. Für Ereignisse ergibt sich hingegen eine kompakte Möglichkeit, den Namen der ausgeführten Aktivität anzugeben.

*Organizational*

- Die Erweiterung *Organizational* (Schlüsselpräfix: *org*) definiert vor allem das Attribut *resource*, das verwendet werden kann, um der Ausführung einer Aktivität die ausführende oder verantwortliche Person zuzuordnen.

*Time*

- Die *Time*-Erweiterung (Schlüsselpräfix: *time*) wird benötigt, um mittels des damit eingeführten *timestamp*-Attributs eine zeitliche Einordnung der Ereignisse angeben zu können. Wie im Beispiel setzt sich der Wert meist aus Datum, Uhrzeit und Zeitzonen-Angabe zusammen.

*Lifecycle*

- Mit der *Lifecycle*-Erweiterung (Schlüsselpräfix: *lifecycle*) ist es möglich, zwischen verschiedenen Stadien der Ausführung einer Aktivität zu unterscheiden. Dazu wird das mit der Erweiterung definierte *transition*-Attribut verwendet. Übliche Belegungen können beispielsweise *start* und *complete* sein (Aktivitätslebenszyklus von DPIL in Abschnitt 3.1.5).

Im Kopf des XES-Ereignisprotokolls kann angegeben werden, welche Attribute für Spuren und Ereignisse jeweils verpflichtend anzugeben sind (⟨*global*⟩). In Codeausschnitt 3.11 sind das Attribut *concept:name* für Ausführungsspuren und die Attribute *concept:name*, *org:resource* sowie *time:timestamp* für alle Ereignisse als verpflichtend angegeben. Gleichzeitig wird so die Wohlgeformtheit der ausgewählten Attribute garantiert, indem ein Standardwert hinterlegt wird (*value*).

```
...
  <lifecycle:model type="LITERAL" value="standardPlusWrite" />
  ...
  <trace>
    <event>
      <string key="org:resource" value="SJ"/>
      <date key="time:timestamp" value="2016-10-13T15:34:00.000+01:00"/>
      <string key="concept:name" value="Informieren"/>
      <string key="lifecycle:transition" value="start"/>
    </event>
    <event>
      <string key="org:resource" value="SJ"/>
      <date key="time:timestamp" value="2016-10-13T15:49:00.000+01:00"/>
      <string key="data:name" value="Kostenvoranschlag"/>
      <float key="data:value" value="1200.0"/>
      <string key="lifecycle:transition" value="write"/>
    </event>
    <event>
      <string key="org:resource" value="SJ"/>
      <date key="time:timestamp" value="2016-10-13T15:51:00.000+01:00"/>
      <string key="concept:name" value="Informieren"/>
      <string key="lifecycle:transition" value="complete"/>
    </event>
...
```

Codeausschnitt 3.12: XES-Ereignisprotokoll mit Datenwerten als separates Ereignis

*Datenwerte*

Die im Beispielprotokoll angegebenen Informationen über die Veränderung von Datenwerten – wie beispielsweise das Erfassen der Kostenschätzung – können

prinzipiell auf zwei Arten XES-konform abgelegt werden. In Codeausschnitt 3.11 wird die *Native Form* verwendet, welche innerhalb eines Ereignisses einen Eintrag darstellt, der aus dem Namen der Datenvariable als Schlüssel und ihrem Wert zum Ereigniszeitpunkt besteht. Eine zweite Variante ist die in Codeausschnitt 3.12 dargestellte *Ereignisform*, in der die Veränderung von Datenwerten als eigenständiges Ereignis im Protokoll eingetragen wird. Für diese zweite Repräsentation muss allerdings das Lebenszyklus-Modell der Aktivitäten erweitert und unter dem Schlüssel *lifecycle:model* im Kopf des Ereignisprotokolls eingetragen werden. In Codeausschnitt 3.11 wird das Lebenszyklus-Modell standardmäßig übernommen, während in Codeausschnitt 3.12 ein erweitertes Modell (*standardPlusWrite*) angegeben wird. Zusätzlich müsste eine eigene Erweiterung definiert werden, die die Attribute *data:name* und *data:value* einführt. Ein Nachteil dieser Repräsentation ist, dass das Schreibereignis für das jeweilige Datum nicht explizit mit einer Aktivität des Prozessmodells assoziiert ist. Grund ist, dass jeder *event*-Eintrag des Protokolls als unabhängiges Ereignis betrachtet wird. In Codeausschnitt 3.12 wird das Schreibereignis von einem Start- und einem Endereignis umschlossen. Setzt man zusätzlich voraus, dass Aktivitäten nicht parallel ausgeführt werden können, dann kann daraus implizit geschlussfolgert werden, dass der Variable *Kostenvoranschlag* innerhalb der Aktivität *Informieren* der Wert 1200 zugewiesen wird. Die Repräsentation von Schreibereignissen als eigenständiger *event*-Eintrag wird im Folgenden nicht näher betrachtet, da alle im Rahmen dieser Arbeit besprochenen Techniken auf die native Repräsentation von schreibenden Datenzugriffen zurückgreifen. Für weitere Informationen über das XES-Format und zum Nachschlagen der vollständigen XES-Syntax verweist der Autor auf [70].

*Native Form*

*Ereignisform*

Der im Konzeptteil vorgestellte Spurgenerator (Abschnitt 4.2) legt erzeugte Prozessausführungsspuren in dem im aktuellen Abschnitt beschriebenen Format ab. Zudem fungieren solche XES-Ereignisprotokolle oft auch als Austauschformat bei der Translation von Prozessmodellen (Abschnitt 5.2).

## 3.4 PROCESS MINING

In diesem Abschnitt wird eine Familie von *induktiven* Techniken beschrieben, die in der Lage sind, aus Ereignisprotokollen Prozessmodelle abzuleiten. Die Algorithmen für die dafür notwendigen Analysen sind teilweise bereits zur Lösung allgemeinerer *Data-Mining*-Probleme verwendet, aber auf die Prozessdomäne spezialisiert worden. Diese spezialisierte Form des Data-Mining wird daher als *Process-Mining* bezeichnet. Einerseits werden diese Techniken zur Analyse bereits formalisierter und andererseits zur initialen Modellierung bisher nicht formalisierter Prozesse verwendet. Im Kontext der vorliegenden Arbeit werden Techniken aus dem Bereich des Process-Mining als Komponenten zur Translation von Prozessmodellen verwendet. Daher wird in den nachfolgenden Abschnitten sowohl das Grundprinzip als auch speziellere Konzepte für das Process-Mining im Kontext imperativer respektive deklarativer Prozessmodelle beschrieben.

*Induktivität*

*Data-Mining und Process-Mining*

### 3.4.1 *Grundprinzip und Herkunft*

*Daten, Informationen und Wissen*

Organisationen verfügen oft über zum Teil riesige Datenbestände aus den unterschiedlichsten Bereichen des Unternehmens. Das Extrahieren von Informationen und das Erlangen von darauf aufbauendem Wissen ist für den Erhalt oder die Steigerung der Wettbewerbs- und Leistungsfähigkeit obligatorisch. Auch wenn diese Erkenntnis nicht neu ist, wird sie unter dem Begriff der *Data Science* seit kurzem in unterschiedlichen Bereichen von Wissenschaft und Wirtschaft als Brennpunktthema behandelt. Data Science wird dabei als interdisziplinäres Feld beschrieben, welches das Ziel verfolgt, aus Daten realen Nutzen zu ziehen. Der Nutzen ist meist die Beantwortung von Fragen, die in die folgenden vier Hauptkategorien eingeteilt werden können [6, S. 10f]:

*Data Science*

- *Was* ist passiert? (Bericht)
- *Warum* ist es passiert? (Diagnose)
- Was *wird* passieren? (Vorhersage)
- Was *sollte* passieren? (Empfehlung)

Zur Beantwortung dieser Fragen müssen mittels geeigneter Infrastrukturen Daten extrahiert, vorbereitet, erforscht, transformiert, gespeichert, verknüpft und anderweitig verwertet werden. Diese Arbeitsschritte sind die operativen Grundbausteine der Data Science und gestalten sich je nach Fragestellung anders. Die zur Bewältigung der Arbeitsschritte notwendigen Techniken stammen dabei aus verschiedenen *Subdisziplinen*. Beispiele für Subdisziplinen der Data Science sind *Statistik*, *Data Mining* und *Maschinelles Lernen*. Data Mining hat seinen Ursprung im Bereich der Datenbanken, während maschinelle Lernverfahren im Kontext künstlicher Intelligenz erforscht werden. Beide wenden Prinzipien der Statistik-Disziplin an. Folglich sind die Subdisziplinen der Data Science nicht überschneidungsfrei.

*Subdisziplinen*

Auch das *Process Mining* ist eine Subdisziplin der Data Science und überschneidet sich ebenfalls mit dem Bereich Data Mining. Unter Data Mining werden Verfahren und Algorithmen zusammengefasst, welche eine automatisierte Analyse großer Datenmengen durchführen [64]. Daraus gewonnene Informationen werden zur Vereinfachung und Übersichtlichkeit oft in Form eines *Modells* abgebildet. Dieses bereitet die Informationen durch Abstraktion, Reduktion und Pragmatismus in einer geeigneten Notation für die angestrebte Verwendung auf [170]. Ein Modell kann hierbei beispielsweise eine mathematische Funktion sein, die eine gegebene Menge an Datenpunkten adäquat beschreibt, was wiederum oft grafisch dargestellt werden kann. Ein Modell kann auch aus einer Menge sogenannter *Assoziationsregeln* bestehen. Assoziationsregeln werden klassisch zur Beschreibung von Abhängigkeiten beim Kauf von Produkten verwendet. Diese Modelle lassen sich durch die Analyse transaktionaler Verkäufe gewinnen und erlauben unter anderem Schlussfolgerungen, welche Produkte möglichst gemeinsam präsentiert werden sollten. Die Erzeugung eines Modells kann *manuell* oder *automatisiert* erfolgen. Besonders im Hinblick auf die derzeit aktuelle *Big-Data*-Problematik ist eine manuelle Modellierung in vielen Fällen höchst ineffizient. Dies ist der Grund

*Data Mining*

*Assoziationsregeln*

*Manuelle vs. Automatisierte Modellerzeugung*

für die Entwicklung von Data-Mining-Techniken zur automatisierten Ableitung von Modellen auf einer gegebenen Datengrundlage.

*Process Mining und Data Mining*

*Process Mining* bezeichnet die Verwendung von Data-Mining-Techniken zur Verarbeitung von prozessbezogenen Daten für die automatisierte Erstellung von Prozessmodellen. Damit stellt es auch ein Bindeglied zwischen Prozessmanagement und Data Mining dar. Die grundlegende Idee ist, aus Aufzeichnungen realer Prozessabläufe Modelle zu erzeugen, welche eben diese Abläufe adäquat beschreiben, und mittels dieser Modelle den beschriebenen Prozess zu erfassen, zu überwachen und zu verbessern [6]. Es existieren zahlreiche Systeme, welche Informationen über den operativen Prozess in strukturierter Form protokollieren (Abschnitt 3.3.2). Die Aufzeichnung erfolgt meist ereignisbasiert in Form von Ereignisprotokollen (Abschnitt 3.3.1). Ereignisse quittieren dabei verschiedenste Vorgänge, wie das Abgeben eines Lotteriescheins, das Anmelden in einer Arztpraxis oder die Beantragung eines Personalausweises. Die Ereignisprotokollierung ist eine notwendige Grundlage. Allerdings liegt die eigentliche Herausforderung darin, die vorliegenden Protokolle automatisiert *nutzbar* zu machen – was wiederum die Motivation für die Entwicklung von Data-Mining-Techniken war und ist. Mittels dieser ist es möglich, die Vorgänge und Zusammenhänge im Prozess besser zu verstehen, um Flaschenhälse zu identifizieren, Abweichungen zwischen gewünschtem und tatsächlichem Verhalten aufzudecken sowie darauf aufbauend den Prozess zu optimieren.

Ausgangspunkt für das Process Mining sind also zeitlich geordnete Aufzeichnungen von Ereignissen. Die Protokollinhalte können auf drei verschiedene Arten genutzt werden:

*Prozessentdeckung*

- Das *Entdecken* (engl. *process discovery*) von Prozessen im Sinne der Extraktion eines Prozessmodells ist die prominenteste Form des Process Mining. Hierbei werden Ereignisprotokolle nach Mustern durchsucht, aus denen üblicherweise ein Prozessmodell abgeleitet wird.

*Konformitätsprüfung*

- In der *Konformitätsprüfung* (engl. *conformance checking*) werden ein gegebenes Prozessmodell und ein ebenso vorliegendes Ereignisprotokoll verglichen. Ziel dieses Vergleichs ist entweder eine Einschätzung, wie gut das Modell die protokollierten Abläufe beschreibt oder wie strikt letztere den Vorgaben des Modells folgen. Das Ergebnis sind quantitative Informationen über Abweichungen und Übereinstimmungen zwischen Modell und Protokoll.

*Verbesserung*

- Die *Verbesserung* von Prozessmodellen (engl. *process enhancement*) nutzt identifizierte Abweichungen zwischen einem gegebenen Modell und einem bereitgestellten Ereignisprotokoll, um ersteres zu korrigieren (engl. *repair*) oder zu erweitern (engl. *extension*). Erlaubt das Protokoll einer realen Prozessausführung beispielsweise eine Abfolge von Aktivitäten, die vom Modell verboten ist, wird das Modell entsprechend korrigiert. Eine Erweiterung kann dann notwendig sein, wenn das Modell beispielsweise nur die sequentielle Abfolge von Aktivitäten nicht aber die Zuordnung von Prozessbeteiligten beschreibt. Enthält das verfügbare Protokoll diese Informationen, dann wird das existierende Modell um diese organisatorischen Informationen und damit letztlich um eine weitere Perspektive erweitert. Das Resultat ist ein korrigiertes und/oder erweitertes Prozessmodell.

| Fall | Zeitstempel | Aktivität | Abkürzung |
|---|---|---|---|
| LA-Swed2015 | 2015-05-08T09:47 | Informieren | a |
| | 2015-05-08T10:13 | Antrag stellen | b |
| | 2015-05-10T10:22 | Antrag genehmigen | c |
| | 2015-05-11T14:30 | Unterkunft buchen | d |
| | 2015-05-11T09:50 | Reisedokumente zusammenstellen | e |
| SJ-Rhodos2016 | 2016-10-13T15:34 | Informieren | a |
| | 2016-10-14T09:20 | Antrag stellen | b |
| | 2016-10-15T07:55 | Unterkunft buchen | d |
| | 2016-10-16T14:25 | Antrag genehmigen | c |
| | 2016-10-17T10:24 | Reisedokumente zusammenstellen | e |

Tabelle 3.5: Vereinfachtes Ereignisprotokoll für Dienstreiseabwicklungsprozess

Das Process Mining basiert nicht nur auf dem Data Mining, sondern auch auf der logischen Inferenz und Aspekten des maschinellen Lernens. Mark Gold stellte bereits 1967 eine Reihe von Problemen logischer Inferenz vor [67], in denen aus einer unbegrenzten Kette an Beispielen Regeln angeleitet werden. Aus dem Bereich des Data Mining steht mit dem Apriori-Algorithmus eine Technik zur automatisierten Extraktion von Assoziationsregeln aus großen Datenbeständen zur Verfügung. Erweiterungen dieses Algorithmus erlauben die Entdeckung von wiederkehrenden Sequenzen in zeitlich geordneten Daten, ohne jedoch ein vollständiges Prozessmodell erzeugen zu können. Dies wurde in den späten 90er Jahren erstmals im Bereich der *Software-Entwicklungsprozesse* – kurz: Softwareprozesse – ermöglicht [44]. Nebenläufigkeit und Parallelisierung wurden jedoch ausgeblendet. Zudem fokussiert ein Softwareprozess immer die Erzeugung von Software-Artefakten, während die Ziele in einem Geschäftsprozess vielfältiger ausfallen können.

*Softwareprozesse vs. Geschäftsprozesse*

In den nachfolgenden Abschnitten werden an jeweils einem Process-Mining-Verfahren die Grundprinzipien der Extraktion von imperativen respektive deklarativen Prozessmodellen aus Ereignisprotokollen erläutert. Die kompakte Form der Darstellung ist darauf zurückzuführen, dass im Rahmen der vorliegenden Arbeit keine neuen Process-Mining-Verfahren entwickelt, sondern lediglich existierende Verfahren *angewendet* werden sollen. Nahezu alle der nachfolgend erwähnten Techniken sind als Erweiterung der Plattform *ProM*[9] implementiert.

### 3.4.2 *Process Mining für imperative Prozessmodelle*

Imperative Prozessmodelle beschreiben alle möglichen Prozessverläufe *explizit* und greifen in der Regel auf eine Darstellung als gerichteter Graph zurück (Abschnitt 1.1.4). Process-Mining-Verfahren, die aus Ereignisprotokollen imperative Prozessmodelle extrahieren, müssen also aus den einzelnen Ausführungsspuren eines Protokolls einen solchen gerichteten Graphen ableiten. Nachfolgend wird das grundsätzliche Vorgehen am Beispiel des *α-Algorithmus* [6, 8] beschrieben.

*α-Algorithmus*

Der α-Algorithmus gilt in der Process-Mining-Gemeinschaft als gute Einfüh-

9 `http://www.promtools.org`, zuletzt aufgerufen: 30.05.2017

rung in das Thema [6, 161]. Gleichzeitig wird von einer praktischen Anwendung abgeraten, da der simpel gehaltene Algorithmus anfällig für Rauschen ist und nicht in der Lage ist, komplexere Konstrukte in Prozessmodellen zu realisieren. Dennoch werden in den später entwickelten Process-Mining-Techniken viele Ideen aus dem $\alpha$-Algorithmus übernommen. Die Eingabe für den $\alpha$-Algorithmus ist ein Ereignisprotokoll $L$ (Definition 3.3). Das Resultat ist ein rein kontrollflussbasiertes Petri-Netz, welches unter anderem Nebenläufigkeit enthalten kann.

*Mustersuche für Paare von Aktivitäten*

Grundlegend durchsucht der Algorithmus das Ereignisprotokoll nach Mustern zwischen Paaren von Aktivitäten und bildet diese kombiniert als Petri-Netz ab. Zur Veranschaulichung des Verfahrens ist in Tabelle 3.5 eine vereinfachte Version des Beispielprotokolls aus Tabelle 3.4 dargestellt. Der $\alpha$-Algorithmus unterscheidet nicht zwischen dem Beginn und dem Ende der Bearbeitung einer Aktivität, weswegen diese Spalte ausgeblendet wird. Da der Algorithmus außerdem lediglich Reihenfolge- und Existenzbeziehungen zwischen Aktivitäten untersucht, kann die Definition eines Ereignisses $e$ (Definition 3.3) vereinfacht angegeben werden als $e = ((\mathtt{activity}^e, \mathtt{act}), (\mathtt{time}^e, t)$, wobei $\mathtt{act}$ der Name der Aktivität und $t$ der jeweilige Zeitstempel ist. In einer zusätzlichen Spalte *Abkürzung* werden die Namen der Aktivitäten durch je einen Buchstaben kodiert. Ausgehend von der in Definition 3.3 geforderten Totalordnung ist eine weitere Vereinfachung möglich, in der ein Ereignis mit dem Kürzel für die jeweilige Aktivität kodiert wird: $e = ((\mathtt{activity}^e, \mathtt{act}), (\mathtt{time}^e, t) = \mathtt{act}$. Folglich sind die Ereignisse der ersten Ausführungsspur (Tabelle 3.5) definiert als $e_1 = a$, $e_2 = b$ und so weiter.

*Reihenfolge- und Existenzabhängigkeiten*

Der $\alpha$-Algorithmus sucht im Ereignisprotokoll gezielt nach vier verschiedenen Mustertypen zwischen zwei Aktivitäten $a$ und $b$. Diese sind:

- $a > b$, falls gilt: $\exists \sigma = \langle e_1, e_2, ..., e_n \rangle : \exists e_i, e_{i+1} : e_i = a \wedge e_{i+1} = b$,
- $a \rightarrow b$, falls gilt: $a > b \wedge b \not> a$,
- $a \# b$, falls gilt: $a \not> b \wedge b \not> a$ und
- $a \parallel b$, falls gilt: $a > b \wedge b > a$

Für das Beispiel Tabelle 3.5, dann ergibt sich für das dargestellte Ereignisprotokoll $L = \{\langle a, b, c, d, e\rangle, \langle a, b, d, c, e\rangle\}$ die folgenden Relationen:

- $>= \{(a, b), (b, c), (c, d), (d, e), (b, d), (d, c), (c, e)\}$
- $\rightarrow = \{(a, b), (b, c), (d, e), (b, d), (c, e)\}$
- $\# = \{(a, a), (b, b), (c, c), (d, d), (e, e), (a, c), (a, d), (a, e), (b, e), (c, a), (d, a), (e, a), (e, b)\}$
- $\parallel = \{(c, d), (d, c)\}$

*Einfache Sequenz*

Die $>$-Beziehung beinhaltet geordnete Paare von Aktivitäten, die in mindestens einer Prozessausführungsspur in der angegebenen Reihenfolge direkt aufeinanderfolgen. So gilt im Beispiel $a > b$, weil $b$ sogar in beiden Spuren der direkte Nachfolger von $a$ ist. Dagegen gilt $b > e$ nicht, weil diese Sequenz in keiner der beiden Spuren auftritt.

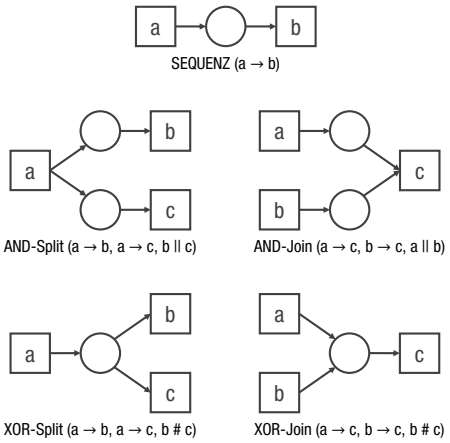



Abbildung 3.11: Kontrollflussmuster des $\alpha$-Algorithmus (Quelle: [6])

*Unidirektionale Sequenz*

Mit $\rightarrow$ wird eine kausale Beziehung bezeichnet, die dann gilt, wenn die Sequenz zweier Aktivitäten auftritt, niemals aber eine Sequenz in umgekehrter Reihenfolge. Folglich ist das Paar $(a, b)$ auch Teil dieser Relation, da es in $>$ und damit unter anderem in der ersten Spur auftritt, während die umgekehrte Sequenz und damit das Paar $(b, a)$ nicht Teil von $>$ ist. Dies interpretiert der Algorithmus als kausale Abhängigkeit. Folglich geht er von der Vollständigkeit des Ereignisprotokolls, also von dem Auftreten aller im Prozess erlaubten Sequenzen aus.

*Voneinander unabhängige Paare*

Die #-Beziehung beschreibt, nach Auffassung des Algorithmus, geordnete Paare von Aktivitäten, die in keiner direkten Abhängigkeit zueinander stehen. So gehört das Paar $(a, e)$ zu dieser Relation, da diese in den Prozessausführungsspuren in keiner Reihenfolge unmittelbar aufeinanderfolgen.

*Voneinander abhängige Paare*

Die $\|$-Relation beschreibt schließlich geordnete Paare von Aktivitäten, die mindestens einmal in jeder Reihenfolge in mindestens einer der Prozessausführungsspuren unmittelbar aufeinanderfolgen. Das trifft im genannten Beispiel auf das Paar $(c, d)$ und folglich auch auf das Paar $(d, c)$ zu, da die unmittelbare Folge dieser beiden Aktivitäten in den zwei Prozessausführungsspuren gegensätzlich ist.

Das Ergebnis dieser Analyse sind Mengen geordneter Paare von Aktivitäten, die in einer der oben beschriebenen Beziehungen stehen. Der nächste Schritt im Algorithmus ist, diese Einzelbeziehungen zu einem Graphen zu verknüpfen. Dies erfolgt auf Basis vorgegebener Kontrollflussmuster (Abbildung 3.11).

*Erzeugung von Verzweigungen*

Zunächst sucht der Algorithmus nach den vier komplexeren Mustern, um XOR- und AND-Verzweigungen respektive -Zusammenführungen zu materialisieren. Dazu wird nach Relationen gesucht, wie sie unter jedem Kontrollflussmuster angegeben sind. Für das in Tabelle 3.5 dargestellte Ereignisprotokoll würden für die Relationen $b \rightarrow c, b \rightarrow d, c \parallel d$ ein AND-Split und für $c \rightarrow e, d \rightarrow e, c \parallel d$ eine entsprechende Zusammenführung erzeugt werden. XOR-Verzweigungen gibt es keine. Die Verknüpfung der übrigen Aktivitäten erfolgt dann mittels Sequenz-Beziehungen und das resultierende Petri-Netz entspricht dem in Abbildung 3.12.

Die Idee, das Kontrollflussgerüst imperativer Prozessmodelle durch das Extrahieren von Relationen für Paare von Aktivitäten aus dem Ereignisprotokoll aufzubauen, wird von vielen verbesserten Techniken übernommen. Ein Beispiel dafür ist der *HeuristicsMiner* [190]. Der $\alpha$-Algorithmus weist jedoch zahlreiche Limi-

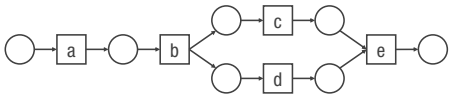



Abbildung 3.12: Vom $\alpha$-Algorithmus aus dem Ereignisprotokoll abgeleitetes Petri-Netz

tierungen auf. Einerseits berücksichtigt er keinerlei datenbezogene Informationen, Ressourcen-Zuweisungen oder den Einsatz von Software-Unterstützung. Folglich sind die einzigen abgedeckten Perspektiven die Funktionale und die Verhaltensorientierte. Weiterhin ist der Ansatz äußerst anfällig für *verrauschte* Ereignisprotokolle. Ist beispielsweise die Sequenz $a \rightarrow b$ in zahlreichen Prozessausführungsspuren eingehalten und wird lediglich einmal missachtet, dann würde dies vom $\alpha$-Algorithmus direkt als Verzweigung umgesetzt werden. Es ist somit nicht möglich, den *typischen* Prozessverlauf aus einem verrauschten Protokoll zu extrahieren. Dennoch zeigt der Algorithmus anschaulich das Grundprinzip der Extraktion eines imperativen Prozessmodells aus einem Ereignisprotokoll.

### 3.4.3 *Process Mining für deklarative Prozessmodelle*

Deklarative Prozessmodelle beschreiben alle möglichen Prozessverläufe *implizit* und greifen meist auf eine regelbasierte Darstellung zurück (Abschnitt 1.1.4). Process-Mining-Techniken, welche aus Ereignisprotokollen deklarative Prozessmodelle ableiten, müssen aus den einzelnen Prozessausführungsspuren auf allgemeingültige Regeln schließen. Das Grundprinzip dieser Klasse von Techniken wird nachfolgend am Beispiel des *DeclareMiners* [116] beschrieben.

*DeclareMiner*

Der DeclareMiner ist eines der ersten Process-Mining-Verfahren für deklarative Prozessmodelle. Als Eingabe dient erneut ein Ereignisprotokoll und das Resultat ist ein Prozessmodell, welches in der deklarativen Prozessmodellierungssprache Declare formuliert ist. Für Declare existiert mit ConDec zwar eine auf Graphen basierende konkrete Syntax, allerdings handelt es sich dabei zum einen nicht um *gerichtete* Graphen und zum anderen stellen die Graphen keine expliziten Abläufe des Prozesses dar. Stattdessen beschreiben sie Regeln, welche die möglichen Prozessverläufe beschränken. Der DeclareMiner extrahiert aus dem Ereignisprotokoll nur Regeln, die mittels der in Abschnitt 3.1.4 beschriebenen Regelschablonen formuliert werden können. Das Verfahren umfasst die folgenden Schritte:

1. *Generierung der Regelkandidaten:* Es werden alle Regelschablonen mit allen Permutationen der im Ereignisprotokoll auftretenden Aktivitäten instanziiert. Die Instanzen werden als Regelkandidaten bezeichnet.

   *Generierung der Regelkandidaten*

2. *LTL-basierte Gültigkeitsprüfung:* Nun erfolgt eine Prüfung der Gültigkeit jedes Regelkandidaten für alle der im Ereignisprotokoll enthaltenen Ausführungsspuren. Dazu wird jeder Regelkandidat in eine LTL-Formel übersetzt deren Erfüllung gegenüber den Ausführungsspuren mittels des *LTLChecker* [4] geprüft wird. Ist eine LTL-Formel nicht erfüllt, wird sie verworfen.

   *Gültigkeitsprüfung*

3. *Modellerzeugung:* Im letzten Schritt werden die nicht herausgefilterten LTL-Formeln wieder in Declare-Regeln umgewandelt.

   *Modellerzeugung*

Der erste Schritt erzeugt alle *möglichen* Regeln. Im zweiten Schritt werden über eine Abbildung auf LTL alle *zutreffenden* Regeln selektiert. Die Komplexität des ersten Schrittes ist durch den *Brute-force*-artigen Ansatz vergleichsweise hoch. Jede Regelschablone hat $k$ Parameter. Im Ereignisprotokoll ist eine endliche Anzahl $n$ an Aktivitäten $A$ enthalten, wobei gilt $n = |A|$. Damit werden für jede Regelschablone $n^k$ Regelkandidaten instanziiert. Im Falle des Ereignisprotokolls aus Tabelle 3.5 ergeben sich für beispielsweise die Regelschablone $\mathtt{precedence}(A, B)$ $5^2 = 25$ Regelkandidaten.

*Parameter zur Rauschfilterung*

Mit den beiden Parametern *Prozent an Prozessinstanzen* (engl. *Percentage of Instances (PoI)*) und *Prozent an Ereignissen* (engl. *Percentage of Events (PoE)*) ist der Algorithmus auch robust gegenüber *verrauschten*[10] Ereignisprotokollen. PoI legt den Anteil der Prozessausführungsspuren fest, in denen ein Regelkandidat erfüllt sein muss, damit er ins Zielmodell übernommen wird. Setzt man diesen Parameter beispielsweise auf 90%, dann genügt es, wenn der Regelkandidat in 9 von 10 Fällen erfüllt ist. Seltene Verstöße gegen diese Regel werden demnach ignoriert. Mit PoE kann die Erzeugung tendenziell *irrelevanter* Regelkandidaten vermieden werden. Mit PoE wird angegeben, welcher Anteil der Aktivitäten, die im Ereignisprotokoll auftreten für die Generierung der Regelkandidaten verwendet wird. Die Menge der so selektierten Aktivitäten setzt sich aus dem gewünschten Anteil der am häufigsten auftretenden Aktivitäten zusammen. Bei $\mathrm{PoE} = 25\%$ wird nur das Viertel der häufigsten Aktivitäten zur Kandidatenbildung verwendet. Das reduziert einerseits die Laufzeit des Verfahrens und andererseits die Komplexität des damit generierten Declare-Modells.

*Limitierungen des DeclareMiners*

Dieses Pionierverfahren des Process Mining für deklarativen Prozessmodelle weist einige Limitierungen auf, sodass auch hier eine praktische Anwendung fraglich ist. Weiter oben wird die hohe Komplexität des ersten Schrittes im Algorithmus erwähnt. Zwar kann diese durch den PoE-Parameter reduziert werden, allerdings geschieht dies auf Kosten der Genauigkeit des Modells. Weiterhin ist das resultierende Modell in vielen Fällen komplexer als nötig. Das Verfahren würde für das in Tabelle 3.5 dargestellte Ereignisprotokoll beispielsweise die Regeln $\mathtt{precedence}(a, b)$, $\mathtt{response}(a, b)$, $\mathtt{succession}(a, b)$ und $\mathtt{chain\ succession}(a, b)$ erzeugen. Die Kombination der $\mathtt{precedence}$- und der $\mathtt{response}$-Regel ist jedoch gleichbedeutend mit der Semantik der $\mathtt{succession}$-Regel. Die $\mathtt{chain\ succession}$-Regel ist dagegen eine „Verschärfung" der $\mathtt{succession}$-Regel, sodass letztere ebenfalls überflüssig ist. Auch transitive Abhängigkeiten würden explizit modelliert werden. Kurz gesagt sind die resultierenden Prozessmodelle deutlich komplexer als notwendig. Dies wird aber in späteren Arbeiten behandelt und, wie auch der $\alpha$-Algorithmus für imperative Modelle, ist der DeclareMiner ein verständlicher Einstieg in das Process Mining für deklarative Modelle. Zudem wird das Grundkonzept in weiterführenden Ansätzen wiederverwendet.

10 Eine kritische Diskussion des Themas „Rauschen" ist in Abschnitt 3.3.2 zu finden.

## 3.5 MODELL-ZU-MODELL-TRANSFORMATIONEN

Die Translation eines Prozessmodells in eine alternative Sprache wird in Abschnitt 1.2 unter anderem als Möglichkeit zur Verbesserung der Verständlichkeit der Modellsemantik und der Interoperabilität unterschiedlicher Prozessmodellierungssprachen identifiziert. Das gegenüber Modelltranslationen allgemeinere Prinzip der *Modell-zu-Modell-Transformationen (M2MT)* wird im aktuellen Abschnitt als Grundlage für erstere und das in Kapitel 5 vorgestellte Translationskonzept für Prozessmodelle eingeführt. Grundlegend unterscheidet man zwischen zwei Typen von Translationssystemen, welche jedoch beide auf Abbildungsregeln auf Modellebene basieren. Das in Kapitel 5 bietet eine dritte Alternative an, die in Teilen auf Ebene der Modell*instanzen* operiert.

*M2MT*

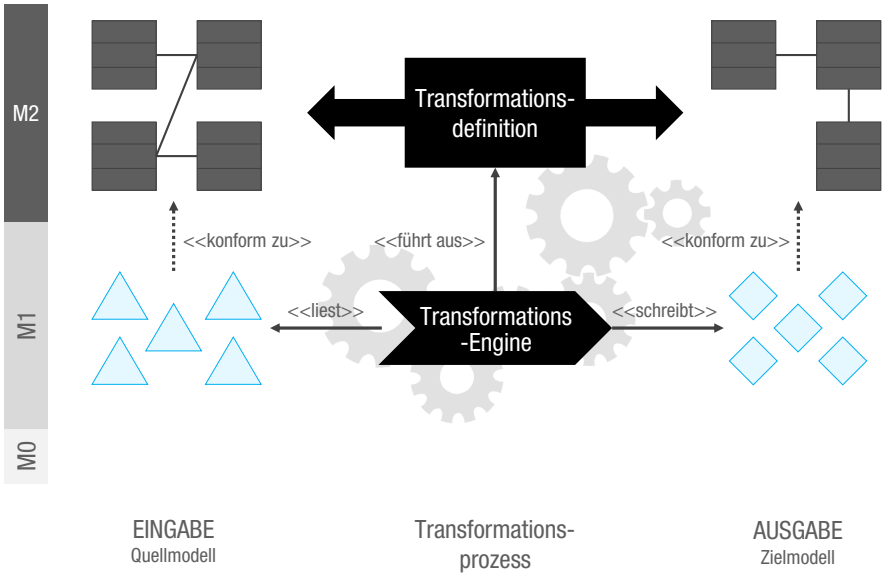



Abbildung 3.13: Allgemeines Prinzip der Modell-zu-Modell-Transformation (basierend auf: [48])

Das allgemeine Prinzip der Transformation [48] eines Modells von einer Modellierungssprache in eine andere ist in Abbildung 3.13 dargestellt und ist dem Bereich der *modellgetriebenen Softwareentwicklung* zuzuordnen. Das Ziel der Transformation ist die Umformung eines Quellmodells in ein für den angedachten Zweck adäquates Zielmodell. Modelle können – müssen jedoch nicht – mittels einer domänenspezifischen Modellierungssprache formuliert werden, welche durch ihre Domänenspezialisierung meist eine „handlichere" Darstellung erlauben als domänenunabhängige Modellierungssprachen. Prozessmodelle sind beispielsweise immer in einer für die Darstellung von Prozessen spezialisierten Sprache formuliert. Allgemein unterscheidet man bei Modellierungssprachen zwischen konkreter und abstrakter Syntax (Abschnitt 1.1.1). Letztere wird in Form eines sogenannten *Metamodells* beschrieben. Ein Modell, welches zu diesem Metamodell konform ist, wird durch Instanziierung erzeugt. Instanziiert man wiederum das Modell, so erhält man ein Artefakt, was im Kontext der modellgetriebenen Softwareentwicklung als eigentliche *Instanz* aufgefasst wird. Die beiden Instanziierungen bilden somit eine

*Metamodell, Modell und Instanz*

Brücke zwischen drei verschiedenen *Metaebenen*, die in der Abbildung mit M0 bis M2 angegeben sind.

*Transformations-definition*

Die Transformation wird auf der Metamodell-Ebene (M2) in Form von Transformationsregeln *definiert*. Würde man die Transformation auf Modell- oder gar Instanzebene definieren, dann müsste man alle möglichen Modelle und Instanzen vorhersehen können. Andernfalls kann nicht garantiert werden, dass für jedes gegebene Modell eine Transformation existiert. Viele Metamodelle können zudem unendlich viele mögliche Instanzen hervorbringen, was eine Vollständigkeit der Definition der Transformation auf Modellebene unmöglich macht.

*Transformations-Engine*

Die Ausführung der Transformation übernimmt eine *Transformations-Engine*. Selbige liest das zu transformierende Prozessmodell, nutzt die Transformationsdefinition, um für eines oder mehrere Quellmodellelemente eines oder mehrere Zielmodellelemente zu erzeugen und zu einem Zielmodell zu kombinieren. Letzteres ist konform zum gewünschten Ziel-Metamodell respektive zur gewünschten Zielsprache.

Für dieses allgemeine Prinzip der M2MT existieren zahlreiche konkrete Umsetzungen. Diese lassen sich nach verschiedenen Kriterien klassifizieren. Im Hauptteil der vorliegenden Arbeit wird neben einer Beschreibung der entwickelten Translationstechnik auch eine Einordnung nach eben diesen Kriterien vorgenommen. Da diese Technik von dem in Abbildung 3.13 skizzierten Grundprinzip abweicht, entfallen eine größere Zahl der Klassifikationskriterien. Daher wird an dieser Stelle auf eine Erläuterung derselben verzichtet und stattdessen auf Abschnitt 5.2 verwiesen.

Die Modell-zu-Modell-Transformation spielt eine Schlüsselrolle im Bereich der modellgetriebenen Softwareentwicklung und dienen nicht ausschließlich der Übersetzung der Modelle in eine andere Sprache. Alternative Anwendungsgebiete sind dabei [48]:

- Synchronisation von Modellen (z.B. im Bereich der Versionskontrolle),
- Die Veränderung des Abstraktionsgrades eines Modells,
- Die Erstellung anfragebasierter Sichten auf ein Modell und
- Der Vergleich von Modellen.

Beispiele für existierende Ansätze finden sich in der Literatur, in OMG-Standards und Werkzeugumsetzungen. Zu ersteren zählen Techniken wie *VIATRA* [184], *AToM* [51] und *MTL* [99]. Die OMG beschreibt mit dem *Query/View/Transformation*-Standard (QVT) eine Familie von drei Transformationssprachen, die sich aus einer Kernsprache, einer deklarativen und einer imperativen Sprache zusammensetzen. Zu den prominenteren Beispielen für Werkzeugumsetzungen zählen *Fujaba* und die *Java Emitter Templates (JET)*. Alle genannten Beispiele folgen dem in Abbildung 3.13 dargestellten Grundprinzip.

Modell-zu-Modell-Transformationstechniken sind die allgemeine Grundlage für den im Hauptteil der vorliegenden Arbeit beschriebenen Ansatz zur Translation von Prozessmodellen. Im verbleibenden Teil des aktuellen Abschnitts wird klargestellt, inwiefern die Translation von Modellen in alternative Modellierungssprachen eine Spezialisierung dieses Grundprinzips darstellt. Zusätzlich werden zwei grundlegend verschiedene Formen der Modelltranslation zusammengefasst.

Die Translation stellt auf Basis der zwei folgenden Charakteristika, welche der in [128] vorgestellten Taxonomie für Modelltransformationen entnommen sind, eine Spezialisierung allgemeiner Modell-zu-Modell-Transformationen dar:

- *Exogenität:* Quell- und Zielmodell sind verschieden, weswegen die Transformation als *exogen* bezeichnet wird. *exogen*
- *Sprachwechsel:* Die Modellierungssprachen von Quell- und Zielmodell sind verschieden. *Sprachwechsel*

Die Autoren von [128] unterscheiden zwischen endogenen und exogenen Transformationen. Transformationen, bei denen Quell- und Zielmodell identisch sind, werden als endogen bezeichnet. Beispiele für meist endogene Transformationen aus der Softwaretechnik sind Refactoring-Mechanismen. Diese verändern die innere Struktur eines Modells und sorgen für Optimierungen, welche die Leistungsfähigkeit steigern sollen, wobei die Semantik in beiden Fällen beizubehalten ist. Im Falle exogener Transformationen sind Quell- und Zielmodell folglich verschieden. Beispiele dafür sind Reverse Engineering, bei denen aus Modellen einer niederen Abstraktionsstufe Modelle einer höheren konstruiert werden und Migrationen von einer Version der Modellierungssprache in eine andere. Translationen sind immer exogen, da bei diesen Quell- und Zielmodell verschieden sind.

Die Exogenität ist zu großen Teilen durch das zweite Charakteristikum bedingt – dem Sprachwechsel. Allgemeine M2MT gestatten, dass Quell- und Zielsprache der Transformation gleich sind. Im Falle einer Translation sind die beiden Sprachen – per Definition – verschieden.

Modelltranslationen können in zwei wesentlichen Formen umgesetzt werden: *(i)* als *direkte* Transformation oder *(ii)* als *Interlingua-basierte* Transformation. Die erste Translationsform umfasst die Verfahren, bei denen lediglich zwei Modelle – nämlich Quell- und Zielmodell – involviert sind. Folglich bilden Transformationsregeln eines direkten Translationssystems Elemente und Konstrukte der Quellsprache direkt auf Elemente der Zielsprache ab. Im Gegensatz dazu sind an einer Interlingua-basierten Translation *drei* Modelle beteiligt. Neben Quell- und Zielmodell ist als drittes ein *Zwischenmodell* an der Übersetzung beteiligt. Zusätzlich existieren pro Translationsrichtung zwei Sätze an Transformationsregeln, von denen einer das Quellmodell in das Zwischenmodell übersetzt und der zweite aus diesem das Zielmodell erzeugt. Die Übersetzung erfolgt also nicht direkt, sondern auf dem Umweg der Übersetzung des Modells in eine Zwischensprache. Die Vor- und Nachteile beider Formen der Translation werden in Abschnitt 5.2 in Kombination mit der in der vorliegenden Arbeit vorgestellten dritten Form – der induktiven Translation – näher betrachtet. *Direkte vs. Interlingua-basierte Transformation*

## 3.6 NATURAL LANGUAGE GENERATION

Natürlichsprachliche Beschreibungen eines Sachverhalts bieten gegenüber formalen Repräsentationen den Vorteil, dass sie für einen weit größeren Personenkreis verständlich sind. Die Barriere des Erlernens einer formalen Syntax wird so umgangen. Allerdings sind natürlichsprachliche Texte nicht immer verfügbar, sodass sie häufig auf Basis formaler Repräsentationen manuell geschrieben werden müssen.

Dieses Teilkapitel beschäftigt sich mit einem grundlegenden Prinzip, welches diesen Schreibprozess automatisiert. Dieses wird als Natural Language Generation bezeichnet und bildet die konzeptionelle Grundlage für den in Abschnitt 2.1 skizzierten Ansatz zur Translation deklarativer Prozessmodelle in natürlichsprachliche Prozessbeschreibungen. Konkret basiert dieser Ansatz auf drei Modellen der Computerlinguistik. Das erste Modell ist ein weit verbreitetes allgemeinen Vorgehensmodell, welches in [154] beschrieben wird. Dieses pipelineartige Modell wird in Abschnitt 3.6.2 kurz vorgestellt, um die einzelnen Verarbeitungsschritte im Konzeptteil kompakter beschreiben zu können. Das im Verlauf von Abschnitt 3.6.3 ebenfalls beschriebene Bedeutung-Text-Modell liefert eine Datenstruktur für den Austausch von Zwischenergebnissen zwischen diesen Verarbeitungsschritten und stellt natürlichsprachliche Sätze als gerichteten Graphen dar. Die danach zusammengefasste Theorie rhetorischer Strukturen (Abschnitt 3.6.4), stellt dagegen Datenstrukturen bereit, um die Makro-Struktur des Zieldokuments, also die Anordnung von Kapiteln, Sektionen, Paragraphen sowie der darin enthaltenen Sätze zu organisieren. In Abschnitt 3.6.1 werden zunächst Gründe diskutiert, welche den Einsatz von NLG-Techniken rechtfertigen. Dies begründet gleichzeitig die Entwicklung des im Rahmen der vorliegenden Arbeit entwickelten NLG-Ansatzes. So nicht anders ausgewiesen, basieren die nachfolgenden Erläuterungen auf den Ausführungen von Reiter und Dale [154].

### 3.6.1 *Gründe und Rahmenbedingungen für den Einsatz von NLG-Techniken*

*Natural Language Generation*

Von der Sprachausgabe heutiger Fahrzeugnavigationssystemen bis hin zur automatischen Erzeugung von Wetterberichten oder personalisierten Briefen, die Transformation formaler Daten in *natürlichsprachliche* Repräsentationen ist bereits seit längerer Zeit im Alltag etabliert. Ein Teilgebiet des *Natural Language Processing*, die sogenannte *Natural Language Generation (NLG)*, beschäftigt sich mit den Herausforderungen der automatisierten Erzeugung natürlichsprachlicher Texte auf formaler Datengrundlage. Ziel ist es, den Einfluss der jeweiligen Anwendungsdomäne auf die Generierung zu erkennen und dadurch geeignete Techniken zu entwickeln, um qualitativ hochwertige natürlichsprachliche Texte zu produzieren. Das Forschungsgebiet ist stark interdisziplinär, da es eine Verknüpfung von Problemen aus dem Bereich der künstlichen Intelligenz, der Computerlinguistik, der Kognitionswissenschaften sowie der Mensch-Maschine-Interaktion darstellt [154]. Die grundlegenden Aufgaben und Herausforderungen sind die zweckgetriebene Extraktion der zu transformierenden Informationen aus der Datenquelle, deren Transformation in natürlichsprachliche Beschreibungen sowie die Anordnung und Vereinfachung einzelner Textteile. Im Kontext der vorliegenden Arbeit dienen die grundlegenden Prinzipien und Methoden der NLG als Basis für die Beschreibung einer Technik zur automatisierten Generierung natürlichsprachlicher Beschreibungen für deklarative Prozessmodelle.

Systeme zur Generierung natürlichsprachlicher Texte auf einer formalen Datengrundlage sind häufig sehr komplex, sodass die Erzeugung in vielen Fällen manuell oder auf Basis einfacher Textschablonen durchgeführt werden. Auch stellt sich in vielen Fällen die Frage, ob die natürlichsprachlich-textuelle Repräsentation oder

eine grafische Veranschaulichung eher geeignet ist, die gewünschten Informationen zu transportieren. Im aktuellen Abschnitt wird erläutert, unter welchen Umständen der Einsatz von NLG-Techniken sinnvoll ist.

*Natürlichsprachlicher Text vs. Grafik*

Eine erste, grundlegende Entscheidung ist, ob sich die natürlichsprachlich-textuelle Form für die Anwendungsdomäne, den jeweiligen Zweck und die jeweilige Zielgruppe eignet. Eine generelle Aussage ist hier nicht möglich. Allerdings zeigen kognitiv-psychologische Studien, dass die Eignung bildhafter und natürlichsprachlicher Repräsentationen unter anderem vom Typ der zu transportierenden Information abhängig ist. Konstruktionspläne und Aufbauanleitungen sind beispielsweise selten in rein natürlichsprachlicher Form zu finden. Dagegen erscheint die Beschreibung von Kausalitäten meist in textueller Form leichter verständlich.

*NLG vs. Textschablonen*

In vielen Anwendungsfällen ist die Erzeugung einer natürlichsprachlichen Repräsentation von Informationen auf Basis vorgefertigter *Textschablonen*, wie beispielsweise Formularen oder Briefvorlagen sinnvoll. Grundlegend kann ein solches Vorgehen ebenfalls als mögliche NLG-Technik betrachtet werden. Die Grenze ist hier fließend. Im Gegensatz zu dynamischeren Techniken sind Schablonen-basierte Verfahren in der Regel kostengünstig und simpel umzusetzen. Komplexere NLG-Systeme sind dagegen in der Lage, abwechslungsreichere und hochqualitativere Texte zu erzeugen. Um den gleichen Variantenreichtum für natürlichsprachliche Texte zu erzielen, ist in der Regel ein größerer Satz an Schablonen notwendig. In diesem Fall steigt der Wartungsaufwand allerdings drastisch, wenn sich die Anforderungen an die zu generierenden Dokumente ändern. NLG-Systeme arbeiten zwar meistens ebenfalls Schablonen-basiert, nutzen dabei aber verschiedene Modelle, um Informationen zentral zu verwalten, was den Wartungsaufwand verringern kann.

*NLG vs. menschlicher Autor*

Software-Systeme sind allgemein kostenintensiv in ihrer Entwicklung und es muss daher abgewogen werden, ob statt eines NLG-Systems ein *menschlicher Autor* mit der Aufgabe der natürlichsprachlichen Beschreibung einer formalen Datengrundlage betraut werden soll. Die Entscheidung ist dann vergleichsweise einfach, wenn die Anzahl zu produzierender Texte in Größenordnungen liegt, für die mehrere oder sogar eine große Zahl menschlicher Autoren eingesetzt werden müssten. Der Einsatz eines NLG-Systems kann jedoch weitere Vorteile gegenüber menschlichen Autoren bieten:

*Konsistenz*

- *Konsistenz:* Menschliche Autoren produzieren für dieselbe Datengrundlage meist unterschiedliche Texte. Nicht immer sind die im Text ausformulierten Informationen gegenüber der Datengrundlage konsistent. Menschliche Autoren neigen, beispielsweise durch Unachtsamkeit, Routine und Ermüdung zu Fehlern. NLG-Systeme produzieren natürlichsprachliche Texte direkt auf der Grundlage der Datenbasis und produzieren keinerlei Flüchtigkeitsfehler.

*Standardkonformität*

- *Standardkonformität:* In vielen Fällen müssen Texte den formalen und inhaltlichen Anforderungen bestimmter Standards entsprechen. Beispielsweise werden technische Dokumentationen im Bereich der Luftfahrtindustrie üblicherweise in einer vereinfachten Version der englischen Sprache nach dem Standard *ASD STE 100 Simplified Technical English* [18] der *Eu-*

*ropean Association of Aerospace Industries (AECMA)*[11] formuliert, um die Verständlichkeit zu verbessern. Menschlichen Autoren kann es schwer fallen, Texte vollständig standardkonform zu verfassen und diese Anforderung stellt demnach eine weitere Fehlerquelle dar. NLG-Systeme können durch eine entsprechende Konfiguration auf derartige Standards verlässlich eingestellt werden.

*Geschwindigkeit*

- *Geschwindigkeit der Texterzeugung:* In Echtzeitanwendungen, wie beispielsweise dem nutzergesteuerten Abruf von Wettervorhersagen kann ein menschlicher Autor ein blockierender Faktor sein. NLG-Systeme bieten hier eine Möglichkeit, Texte in größerer Stückzahl aber mit geringem Zeitaufwand zu produzieren.

*Sprachunabhängigkeit*

- *Sprachunabhängigkeit:* NLG-Systeme können befähigt werden, Texte in verschiedenen natürlichen Sprachen zu erzeugen. Menschliche Autoren können Texte entweder manuell oder mittels eines Übersetzungsprogramms in andere Sprachen übersetzen. Ersteres kann jedoch einen signifikanten Aufwand bedeuten und letzteres ist, beispielsweise durch Ambiguitäten natürlichsprachlicher Ausdrücke, im höchsten Grade fehleranfällig. Bei NLG-Systemen besteht diese Problematik nicht, da die formale Datengrundlage in der Regel frei von Mehrdeutigkeiten ist.

*Verfügbarkeit*

- *Verfügbarkeit:* Menschliche Autoren sind hinsichtlich ihrer täglichen Arbeitszeit und Verfügbarkeit eingeschränkt. Ein NLG-System bietet folglich den Vorteil höherer Verfügbarkeit.

Die im aktuellen Abschnitt genannten Entscheidungskriterien für oder gegen den Einsatz eines NLG-Systems sind allgemeiner Natur und gelten demnach auch in der Prozessdomäne. Für die vorliegende Arbeit dienen sie somit als Beleg, dass die Entwicklung eines NLG-Systems für die Transformation von Prozessmodellen in natürlichsprachliche Prozessbeschreibung prinzipiell sinnvoll ist.

### 3.6.2 *Eine allgemeine Architektur für NLG-Systeme*

Für die Einfachheit der Entwicklung und Wartung von Softwaresystemen bietet sich eine Dekomposition desselben in eigenständige, wohldefinierte und austauschbare Module an. Zwar kann nicht von *der* allgemeinen Architektur von Natural-Language-Generation-Systemen die Rede sein, allerdings können einige Module und eine grundlegende Anordnung derselben identifiziert werden. Die Grundlage dafür lieferten Untersuchungen einer größeren Menge an konkreten NLG-Systemen. Der daraus resultierende Architekturvorschlag wird nachfolgend beschrieben.

*NLG-Pipeline-Architektur*

Die meisten existierenden NLG-Systeme basieren im Wesentlichen auf einem dreischrittigen *Pipeline-Modell*, welches in Abbildung 3.14 dargestellt ist. Die drei Schritte sind üblicherweise in eigene Module gekapselt, was die Austauschbarkeit und eine leichtere Fehlersuche bei der Entwicklung sicherstellt. Die drei

11 http://www.asd-europe.org/home/, letzter Zugriff: 24.02.2017

Module sind die *Dokumentplanung*, die *Mikroplanung* und die *Oberflächenrealisierung*. Diese Module nutzen verschiedenartige Informationen, die nachfolgend überblicksartig beschrieben werden.

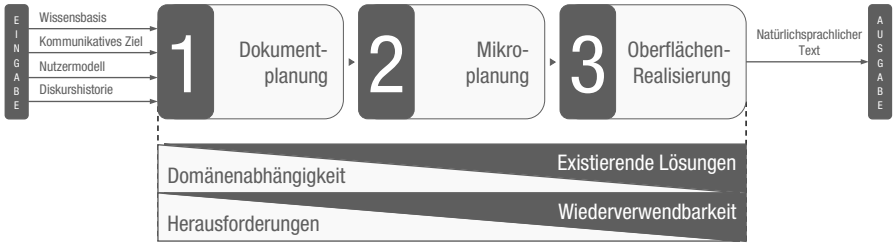



Abbildung 3.14: Pipeline-Architektur für NLG-Systeme

*Wissensbasis*

Die *Wissensbasis* bezeichnet die inhaltliche Grundlage für den zu generierenden natürlichsprachlichen Text. Im Kontext der automatisierten Erstellung von Wettervorhersagen sind dies beispielsweise Temperatur-, Niederschlags- und Luftdruckdaten. Die Wissensbasis beschreibt demnach, *was* mittels natürlicher Sprache beschrieben werden soll.

*Kommunikatives Ziel*

Die übrigen drei Eingabeinformationen spezifizieren dagegen, *wie* der zu generierende Text gestaltet werden soll. Das *kommunikative Ziel* beschreibt den Zweck der Generierung. Dieser ist nicht identisch mit dem allgemeinen Zweck des jeweiligen NLG-Systems. Stattdessen beschreibt das kommunikative Ziel eine konkrete Fragestellung. Im Kontext des Wettervorhersage-Beispiels kann das unter anderem die Anforderung einer Tages- oder Wochenvorschau sein, bei der nur Niederschlagswahrscheinlichkeit und die geschätzten Temperaturen präsentiert werden sollen.

*Nutzermodell*

Das *Nutzermodell* ist eine Charakterisierung der Zielgruppe, für welche die natürlichsprachlichen Texte zu generieren sind. Dazu zählen die Informationen, ob die Personen der Zielgruppe Experten oder Laien in der jeweiligen Domäne sind, ob sie bezüglich der Zielsprache Muttersprachler sind, welche Aufgabe sie unter Zuhilfenahme der zu generierenden Texte bewältigen sollen oder auch individuelle stilistische und linguistische Präferenzen.

*Diskurshistorie*

Nicht immer erfolgt die Generierung natürlichsprachlicher Texte kontextfrei. Durch frühere Interaktionen mit dem NLG-System ist es möglich, dass der Nutzer bereits einen Teil der Informationen erhalten hat. Die *Diskurshistorie* ist ein Modell, welches bereits erwähnte Entitäten, Eigenschaften und Zusammenhänge protokolliert. Dies ermöglicht die sprachliche Referenzierung durch beispielsweise Pronomen und eine kompaktere Präsentation durch den Ausschluss bereits kommunizierter Informationen. Die Diskurshistorie ist besonders für dialogorientierte NLG-Systemem wichtig.

*Dokumentplanung*

Die eben beschriebenen Informationen werden zu unterschiedlichen Anteilen von den drei Pipeline-Schritten verarbeitet. In der Phase der *Dokumentplanung* selektiert das System die formalen Inhalte, die in natürlichsprachlicher Form präsentiert werden sollen. Diese Inhalte stammen aus der Wissensbasis und die Entscheidung für oder gegen die Verwendung bestimmter Inhalte ist vom jeweiligen kommunikativen Ziel, von den Charakteristika der Zielgruppe und den bereits kommunizierten Informationen abhängig. Neben der Selektion der Inhalte ist auch deren Gruppie-

rung und Anordnung im Zieldokument Teil der Dokumentplanung. Das Ergebnis dieses Schrittes ist ein sogenannter *Dokumentplan*, welcher in einer Baumstruktur vorliegt. Die inneren Knoten des Baums bilden Strukturinformationen, wie beispielsweise Kapitelnamen. Blattknoten beinhalten dagegen die in diese Strukturen einzuordnenden Inhalte, die als *Nachrichten* bezeichnet werden. Nachrichten dienen als Mediatoren zwischen den Datenstrukturen der Wissensbasis und den linguistischen Datenstrukturen des NLG-Systems. Die in Abbildung 3.15 darge-

*Dokumentplan*

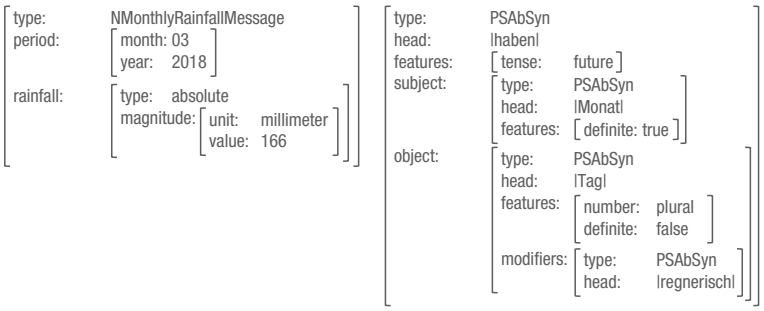



Abbildung 3.15: Beispiel für eine Nachricht (a) und für eine Phrasenspezifikation (b)

stellte Nachricht beschreibt die Informationsgrundlage für die Erzeugung eines Satzes, welcher die zu erwartende absolute Niederschlagsmenge für März 2018 in Millimetern beschreibt.

*Mikroplanung*

Während im vorherigen Modul die *Dokument*struktur bestimmt wird, beschäftigt sich das Modul *Mikroplanung* mit der Satzstruktur, der Lexikalisierung der Informationen, der Einführung referenzierender Ausdrücke sowie der gezielten Verschmelzung von Sätzen. Die Lexikalisierung beschäftigt sich mit der Wortwahl für die in der Dokumentplanungsphase selektierten Informationen. Referenzierende Ausdrücke dienen der Verbesserung der Lesbarkeit natürlichsprachlicher Texte. Dazu zählen beispielsweise Pronomen. Die Verschmelzung von Sätzen wird als Aggregation bezeichnet und dient ebenfalls der Verbesserung der Lesbarkeit der zu generierenden Texte. Dem Beispiel der Wettervorhersage folgend wird so aus den einzelnen Sätzen „Es wird sonnig" und „Es wird warm" die kombinierte Formulierung „Es wird sonnig und warm". Das Ergebnis des Moduls zur Mikroplanung ist eine *Textspezifikation*, welche ebenfalls eine Baumstruktur aufweisen. Die inneren Knoten des Baumes sind erneut die Strukturinformationen des Textes, während die Blätter die Sätze des Textes spezifizieren. Die Blattknoten werden *Phrasenspezifikationen* genannt. Selbige weisen eine ähnliche Struktur auf, wie die Nachrichten des Dokumentplans. Im Unterschied zu letzteren beinhalten Phrasenspezifikationen aber lexikalische und grammatikalische Informationen über den zu erzeugenden Text. Eine denkbare Phrasenspezifikation für die in Abbildung 3.15 (a) dargestellte Nachricht ist neben ersterer dargestellt. Dabei ist auffällig, dass die Phrasenspezifikation weder den genauen Monat noch die genaue Niederschlagsmenge der Wettervorhersage enthält. Das verdeutlicht den Einfluss der Informationen aus dem Nutzermodell, den Angaben zum kommunikativen Ziel oder auch der Diskurshistorie. Stellt ein Nutzer der Zielgruppe die Frage, ob es im März 2018 regnen wird, dann existieren exakt zwei Antworten: Ja und Nein. Folglich ist es nicht

das kommunikative Ziel, alle Details der Nachricht aus Abbildung 3.15 (a) zu kommunizieren.

Phrasenspezifikationen sind in der Regel sprachspezifisch. Eine Möglichkeit zwischen einer Nachricht aus der Dokumentplanung und einer sprachspezifischen Phrasenspezifikation eine weitere Abstraktionsschicht zu definieren, liefert die Theorie des sogenannten *Bedeutung-Text-Modells (BTM)* (engl. *Meaning-Text Theory*) (Abschnitt 3.6.3). Selbiges basiert auf der Analyse verschiedener Sprachen aus verschiedenen Sprachräumen und ermöglicht die schrittweise Übertragung eines geschriebenen oder gesprochenen Texts in seine Bedeutung und umgekehrt. Letzteres kann für ein mehrsprachiges Generierungssystem aber auch für einen allgemeinen Formalismus zur Repräsentation der Phrasenspezifikation genutzt werden.

*Bedeutung-Text-Modell*

In den vorangegangenen Modulen wird der Zieltext *modelliert* und in der Regel in Form einer Baumstruktur an das Modul zur *Oberflächenrealisierung* weitergeleitet. Die Baumstruktur ist eine Spezifikation des Textes. Die Oberflächenrealisierung produziert auf Basis dieser Spezifikation orthografisch und grammatikalisch korrekte Sätze. Für die in Abbildung 3.15 angegebene Phrasenspezifikation wird beispielsweise der Satz „Der Monat wird einige regnerische Tage haben" generiert.

*Oberflächenrealisierung*

Trotz der Tatsache, dass diese Architektur einige Nachteile aufweist, ist sie in vielfältigen Domänen verwendet worden. Ein solcher Nachteil ist beispielsweise, dass nicht in allen Fällen eine so strikte Trennung der Modulverantwortlichkeiten möglich ist. Ist die Anzahl der Zeilen oder Seiten des zu generierenden Texts beispielsweise begrenzt, dann ist ein Überschreiten dieser Begrenzung erst nach der Oberflächenrealisierung zu erkennen. Das Modul zur Oberflächenrealisierung müsste demnach mit dem Dokumentplaner interagieren, da der größte Einflussfaktor für den Umfang des resultierenden Textes die Menge der selektierten Informationen ist. Im Forschungsbereich der Natural Language Generation werden hierzu aber auch Kompensationsstrategien diskutiert, welche einen Teil dieser Nachteile relativiert oder sogar aufhebt.

Den angesprochenen Nachteilen stehen mehrere Vorteile gegenüber. Wie bereits erwähnt, begünstigt die Modularisierung und die Festlegung der Datenstrukturen für den Datenaustausch zwischen den Modulen den Austausch von Teilen der Pipeline-Architektur. Berücksichtigt man, dass besonders für große Teile der Pipeline – insbesondere für das zweite und dritte Modul – wiederverwendbare Lösungen existieren, dann reduziert das den Aufwand der Umsetzung eines neuen NLG-Systems erheblich. Die Verfügbarkeit von fertigen Lösungen für die hinteren Teile der Pipeline lässt sich damit begründen, dass die Domänenabhängigkeit der dort verwendeten Techniken gering oder sogar nicht existent ist. Dagegen ist die Selektion der zu präsentierenden Inhalte durch die Heterogenität möglicher Datenquellen und deren unterschiedliche Formen der Wissensrepräsentation stark domänenabhängig. Das Ergebnis der Mikroplanung ist dagegen eine Textspezifikation, die zwar inhaltlich vollkommen verschieden von anderen Textspezifikationen sein mag, dabei aber den grammatikalischen und lexikalischen Vorgaben der Zielsprache folgt. Folglich ist die Abhängigkeit zur Zielsprache größer als die zur Domäne, sodass für dieselbe Zielsprache tendenziell dieselbe Technik zur Oberflächenrealisierung verwendet werden kann. Gleiches gilt beispielsweise für Techni-

*Vorteile des Pipeline-Ansatzes*

ken der Mikroplanung. Die Aggregation englischer Sätze oder die Einführung von Personalpronomen ist nicht von der Domäne, sondern von der jeweiligen Sprache abhängig. Im Gegensatz dazu ist die Lexikalisierungskomponente desselben Moduls sowohl von der Zielsprache als auch von der Anwendungsdomäne abhängig, da das Domänenvokabular auf das allgemeingültige Vokabular der Zielsprache trifft. Durch den steigenden Grad der Wiederverwendbarkeit einzelner Techniken reduzieren sich gleichzeitig die Herausforderungen für neu zu entwickelnde NLG-Systeme, sodass die Hauptschwierigkeit im Bereich der Dokumentplanung liegt.

*Größte Herausforderung: Dokumentplanung*

Im aktuellen Abschnitt werden die wesentlichen Grundlagen der Konzeption von Systemen zur Natural Language Generation vorgestellt. In vielen Fällen ist eine Pipeline-Architektur, bestehend aus den Modulen Dokumentplanung, Mikroplanung und Oberflächenrealisierung als Rückgrat des Systems geeignet. Neben den eigentlichen zu präsentierenden Informationen nutzt ein NLG-System auch Informationen über den Zweck der Generierung, über die Zielgruppe des Textes sowie Diskursinformationen. Die Module transformieren diese Informationen schrittweise und nutzen verallgemeinerbare baumartige Datenstrukturen für die Informationsweiterleitung. Die Domänenabhängigkeit der Techniken sinkt vom ersten bis zum letzten Modul ab, weswegen die Techniken des hinteren Teils entweder zum Teil oder sogar ganzheitlich wiederverwendbar sind. Die größten Herausforderungen liegen folglich im ersten Teil, namentlich in der Selektion der Informationen und ihrer Anordnung in der Zielstruktur des zu erzeugenden Texts.

### 3.6.3 *Das Bedeutung-Text-Modell*

Das Bedeutung-Text-Modell [125] ist ein linguistisches Rahmenwerk, welches beispielsweise im Kontext der maschinellen Übersetzung aber auch im Bereich der Natural Language Generation Anwendung findet [94]. Da es auf der Analyse sehr verschiedenartiger Sprachen beruht, darunter germanische, romanische und slawische Sprachen, ist es ein geeignetes theoretisches Fundament für die Umsetzung *mehrsprachiger* Systeme. Zudem ermöglicht die klar definierte Trennung zwischen Text und Textspezifikation die Repräsentation von Phrasenspezifikationen. Auch die in Abschnitt 6.2 vorgestellte NLG-Technik nutzt diesen Formalismus zu Generierung natürlichsprachlicher Beschreibungen für deklarative Prozessmodelle. Aus diesem Grund wird das Bedeutung-Text-Modell nachfolgend zusammengefasst.

*Wörterbuch, Grammatik und Lexeme*

Das Modell setzt sich aus einem speziellen *Wörterbuch* und einer *Grammatik* zusammen. Das Wörterbuch beinhaltet Informationen über die Bedeutung und über Kombinationsmöglichkeiten von sogenannten *Lexemen*. Lexeme sind die Bedeutungseinheiten einer Sprache und gruppieren Wörter, deren Bedeutung Überschneidungen aufweisen. Morphologie und die konkrete syntaktische Form des Wortes werden von den Lexemen abstrahiert. Beispielsweise gehören „(ich) *prüf*e“, „(er) *prüf*t“ und „ge*prüf*t“ zum selben Lexem. Flexion und Syntax spielen folglich keine Rolle. Die Wortart ist dagegen ein Unterscheidungskriterium für Lexeme. Das Wort „Prüfer“ ist beispielsweise einem anderen Lexem zuzuordnen. Die Grammatik des Bedeutung-Text-Modells ist eine *dependenzorientierte Transformationsgrammatik*. Eine Transformationsgrammatik ist eine Grammatik, welche zur Generierung von Texten verwendet werden kann. Es handelt sich also um eine ge-

nerative Grammatik. Weiterhin wird zwischen beispielsweise phrasenstruktur- und dependenzorientierten Grammatiken unterschieden. Erstere gruppieren die Wörter eines natürlichsprachlichen Satzes in einer hierarchischen Struktur, deren Ebenen von den einzelnen Phrasen des Satzes gebildet werden. Dagegen verknüpft eine dependenzorientierte Grammatik die Wörter eines Satzes auf Basis ihrer wechselseitigen Abhängigkeit zueinander. Ein Beispiel für eine solche Abhängigkeit ist die Beziehung zwischen einem Nomen und seinem Artikel.

*Dependenzorientierte Grammatik*

Mittels der Grammatik des Bedeutung-Text-Modells kann die Transformation eines natürlichsprachlichen Satzes in seine Bedeutung und umgekehrt modelliert werden. Diese Transformation erfolgt in der Regel mehrstufig. Das Bedeutung-Text-Modell unterscheidet prinzipiell sieben Ebenen, wovon die oberste die *Semantische Repräsentation* und die unterste die *Phonologische* oder *Orthografische Repräsentation* genannt wird. Da für die germanischen Sprachen, zu denen beispielsweise Deutsch und Englisch gehören, nicht alle Ebenen relevant sind, ist die Darstellung der Ebenen in Abbildung 3.16 auf die vier relevanten begrenzt [94].

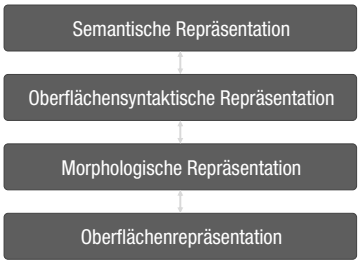



Abbildung 3.16: Relevante Repräsentationsebenen des Bedeutung-Text-Modells

Auf Ebene der *Semantischen Repräsentation* wird die Bedeutung einer Menge synonymer natürlichsprachlicher Sätze als semantisches Netzwerk abgebildet. Die Knoten dieses Netzwerks repräsentieren atomare semantische Einheiten, die *Semanteme* genannt werden. Die bereits vorab genannten Lexeme setzen sich aus einem oder mehreren Semantemen zusammen. Beispielsweise ist die Bedeutung des Lexems „Mutter“ unter anderem durch die Kombination der Semanteme „menschlich“, „erwachsen“, „weiblich“ und „hat Kind“ zu beschreiben. Diese Semanteme werden in der semantischen Repräsentation durch gerichtete Kanten verknüpft, welche ihre Abhängigkeiten voneinander charakterisieren. Die Quelle jeder gerichteten Kante wird als Prädikat und das Ziel wird jeweils als Argument aufgefasst. Beispielsweise nimmt in der Phrase „mehrfache Prüfung“ das Wort „mehrfach“ die Rolle des Prädikats und „Prüfung“ die Rolle des zugehörigen Arguments ein. Das Modell gestattet, dass Argumente von mehreren Prädikaten verwendet werden und, dass Prädikate selbst Argumente für andere Prädikate sein dürfen. Durch die Reduktion natürlichsprachlicher Sätze auf Semanteme und semantische Beziehungen zwischen diesen wird automatisch von syntaktischen und morphologischen Modifikationen abstrahiert. Auf diese Weise lassen sich beispielsweise die bedeutungsgleichen Sätze „Es erfordert eine mehrfache Prüfung“ und „Es ist eine mehrfache Prüfung erforderlich“ mit derselben semantischen Struktur beschreiben. Dadurch wird es möglich, unabhängig davon zu entscheiden, in welcher grammatikalischen Form der natürlichsprachliche Satz materialisiert wird.

*Semantische Repräsentation*

Die Transformation in die Repräsentation der nächst niedrigeren Ebene beinhaltet diese Entscheidung und erfolgt mittels der erwähnten Transformationsgrammatik.

*Oberflächensyntaktische Repräsentation*

Die Ebene der *Oberflächensyntaktischen Repräsentation* ist die für die vorliegende Arbeit wichtigste Repräsentation eines natürlichsprachlichen Satzes, da sie im Konzept des in selbiger vorgestellten NLG-Ansatzes (Abschnitt 6.2) in allen drei Schritten der NLG-Pipeline eine zentrale Rolle spielt. Die Oberflächensyntaktische Repräsentation bildet Abhängigkeitsbeziehungen zwischen Lexemen oder zwischen Lexemen und lexikalischen Funktionen in einer Baumstruktur ab, die als *Deep Syntactic Representation (DSyntR)* bezeichnet werden. Lexikalische Funktionen ermöglichen eine generische Repräsentation von aus grammatikalischen Gründen anzugebenden Wörtern, wie beispielsweise Hilfsverben. Sowohl Lexeme als auch lexikalische Funktionen werden als Knoten in den Abhängigkeitsbäumen abgebildet. Die Beziehungen zwischen diesen werden dementsprechend durch beschriftete Kanten hergestellt, die wie folgt kategorisiert sind:

*Deep Syntactic Representation*

- *I - VII:* Prädikat-Argument-Beziehungen unterschiedlichen Ranges,
- *ATTR:* Modifikatoren für Phrasen, z.B. Adjektive, Adverbien und Relativsätze,
- *COORD:* Koordinierende Konjunktionen, beispielsweise „oder" und „aber".

Die Beziehungen zwischen Prädikaten und Argumenten ist im wesentlichen aus der semantischen Repräsentation übernommen. Ein wichtiger Unterschied ist hier jedoch, dass Prädikate und Argumente keine Semanteme, sondern Lexeme oder lexikalische Funktionen sind. Die Rangfolge der Beziehungen beschreibt die grammatikalische Funktion des Satzteils. Die drei ersten Ränge sind die am häufigsten anzutreffenden und beschreiben entweder eine Subjektfunktion (*I*), eine Akkusativ-Objekt-Funktion respektive andere Phrasentypen mit Objekten (*II*) oder die Funktion eines Dativ-Objekts (*III*). Zwei dieser Beziehungstypen sind in der in Abbildung 3.17 dargestellten Oberflächensyntaktischen Repräsentation verwendet worden. Die Wurzel bildet das Hauptverb des Satzes. Von selbigem geht eine Kante mit der Beschriftung *I* aus, welche das Wort „Professor" als Subjekt des Satzes ausweist. Dagegen ist die Beziehung zu „Antrag" durch die Beschriftung *II* als Akkusativ-Objekt-Funktion festgelegt. Das Adverb „immer" wird entsprechend seiner Funktion als Phrasenmodifikator mit einer Kante der Kategorie *ATTR* verknüpft. Da der Satz lediglich aus einem Hauptsatz besteht, enthält die Oberflächensyntaktische Repräsentation keine koordinierende Konjunktion.

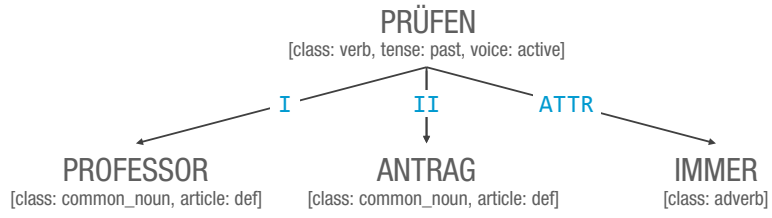



Abbildung 3.17: DSyntR für den Satz „Der Professor genehmigte den Antrag immer."

*Grammeme*

Ein weiterer Aspekt der DSyntR-Strukturen sind die *Grammeme*, die atomaren grammatikalischen Eigenschaften der in der Struktur enthaltenen Lexeme und lexikalischen Funktionen. Das als Hauptverb bezeichnete Lexem „PRÜFEN" ist über

die Grammeme für die Wortart („class"), die Zeitform („tense") sowie das Genus verbi („voice") grammatikalisch näher bestimmt. Die Kombination der Grammeme ergibt in diesem Fall, dass das Wort „prüfen" ein Verb ist, welches in aktiver Vergangenheitsform in den Satz eingegliedert werden soll.

Die Rangfolge der Beziehungen zwischen den einzelnen mit Grammemen versehenen Lexemen der Oberflächensyntaktischen Repräsentation wird in der *Morphologischen Repräsentation* verwendet, um die lineare Struktur des Zielsatzes zu formen. Grundbestandteil der Repräsentation sind hier die sogenannten *Morpheme*. Morpheme sind die kleinsten bedeutungstragenden Einheiten eines Satzes. Im Unterschied zu Lexemen sind diese nicht abstrakt und fasst nicht verschiedene Flexionsformen zusammen, sondern ist eine dieser Flexionsformen. Aus dem DSyntR-Baum, welcher in Abbildung 3.17 dargestellt ist, würde die folgende lineare Struktur entstehen: *Der Professor*$_{[\mathtt{sg}]}$ *genehmigen*$_{[\mathtt{past,3,sg,active}]}$ *den Antrag*$_{[\mathtt{sg}]}$ *immer.*

*Morphologische Repräsentation*

*Morpheme*

Bei der Transformation in die *Oberflächenrepräsentation* wird die lineare Struktur der Morphologischen Repräsentation übernommen. Folglich ist die Hauptaufgabe in diesem Schritt die Realisierung der Flexionsformen gemäß der in den Grammemen angegebenen Informationen. Insgesamt beschreibt die in Abbildung 3.17 dargestellte syntaktische Struktur damit den Satz „Der Professor genehmigte den Antrag immer."

*Oberflächenrepräsentation*

Mit dem in diesem Abschnitt beschriebenen Bedeutung-Text-Modell lassen sich NLG-Systeme einerseits mehrsprachig und andererseits flexibler hinsichtlich der unterstützten Ausdrucksmöglichkeiten gestalten. Beispielsweise kann über die Grammeme zwischen Aktiv- und Passivkonstruktionen sowie verschiedenen Numeri und Tempora variiert werden, ohne die dafür notwendigen Informationen über Syntaxmodifikationen, Flexion und Morphologie explizit in textuellen Schablonen angeben zu müssen. Das ermöglicht eine von grammatikalischen Merkmalen getriebene Spezifikation des Zieltexts. Zudem erleichtert die formale Repräsentation eines Satzes als DSyntR-Struktur die Kombination von Sätzen mittels Konjunktion und Schachtelung. Grund dafür ist, dass die DSyntR-Struktur die Beziehungen zwischen den einzelnen Satzteilen spezifiziert, was in einer Repräsentation als Zeichenkette nicht gegeben wäre. Die Transformation der DSyntR-Repräsentationen in einen natürlichsprachlichen Satz wird in zahlreichen Ansätzen untersucht und durch Werkzeugunterstützung modularisiert. Da sich die Beschreibung des Bedeutung-Text-Modells auf die für die vorliegende Arbeit relevanten Aspekte beschränkt, wird für weiterführende Informationen auf die Quellen [125] und [94] verwiesen.

### 3.6.4 *Theorie rhetorischer Strukturen*

Die *Theorie rhetorischer Strukturen (RST)* (engl. rhetorical structure theory) ist eine Methode zur Beschreibung von Dokumentstrukturen auf Basis rhetorischer Relationen [119]. Für die Generierung natürlichsprachlicher Texte aus formalen Beschreibungen bietet sie ein linguistisches Fundament zur schrittweisen Diskursplanung auf Basis dieser Relationen [79]. Der in Abschnitt 6.2 vorgestellte Ansatz zur Generierung natürlichsprachlicher Prozessbeschreibungen auf Basis deklarati-

*Theorie rhetorischer Strukturen*

ver Prozessmodelle nutzt dieses Grundprinzip im Rahmen der Dokumentplanung. Aus diesem Grund wird die RST nachfolgend auszugsweise vorgestellt.

| Relationsname: | Elaboration |
|---|---|
| Beschränkungen für N: | keine |
| Beschränkungen für S: | keine |
| Beschränkungen für die Kombination N + S: | Der Satellit S präsentiert zusätzliche Details über eine Situation oder ein Element der Thematik, welche im Nukleus N präsentiert wurde oder durch selbigen per Inferenz zugänglich ist. Die Arten derartiger N:S-Beziehungen können sein:<br>1. Menge: Element<br>2. Abstrakt: Instanz<br>3. Ganzes: Teil<br>4. Prozess: Prozessschritt<br>5. Objekt: Attribut<br>6. Generalisierung: Spezialisierung |
| Effekt: | Der Leser erkennt, dass die in S beschriebene Situation zusätzliche Details für N bereitstellt. Der Leser ist weiterhin in der Lage, das Element der Thematik zu erkennen, für welches die Details bereitgestellt werden. |
| Ort des Effekts: | N und S |

Abbildung 3.18: Spezifikation der Elaborationsrelation der Theorie rhetorischer Strukturen

*Rhetorischer Strukturbaum*

Grundlegend basiert die Theorie auf der Dekomposition eines Textes in einzelne Segmente und einer Beschreibung der Beziehungen zwischen einzelnen Segmenten mittels Diskursrelationen. Eine solche Verknüpfung kann wiederum mittels einer Diskursrelation mit anderen Segmenten verknüpft werden. Die vollständige Beschreibung eines Textes nach diesem Prinzip entspricht folglich einer Baumstruktur, welcher als *Rhetorischer Strukturbaum* bezeichnet wird und deren innere Knoten jeweils eine Diskursrelation zwischen den zugehörigen Kindknoten beschreiben. Die Blätter des Baumes repräsentieren dagegen in der Regel die einzelnen Sätze, da diese als elementare Aussagen des Textes betrachtet werden. Die Verknüpfung dieser elementaren Aussagen basiert auf der vielfach bestätigten Annahme, dass jeder Text kohärent ist, also inhaltlich vollständig zusammenhängt. Die Art des Zusammenhangs wird über die bereits erwähnten Diskursrelationen angegeben. Empirische Untersuchungen ergaben, dass die Kohärenz natürlichsprachlicher Texte mit einer kleinen Menge dieser Relationen beschrieben werden kann [154, S. 102]. Die meisten dieser Diskursrelationen sind binär, was bedeutet, dass sie zwei Segmente miteinander verknüpfen. Ein Beispiel dafür kann anhand der folgenden zwei natürlichsprachlichen Satzpaare (*A1* und *A2* respektive *B1* und *B2*) deutlich gemacht werden.

- *A1:* Der Prozess umfasst fünf Schritte.
- *A2:* Einer dieser Schritte ist die Aktivität *Flug buchen*.
- *B1:* Der Prozess umfasst fünf Schritte.
- *B2:* Normalerweise umfasst ein Prozess etwa 10 Schritte.

Bereits rein intuitiv können Unterschiede in der Beziehung zwischen den Satzpaaren abgeleitet werden. Der zweite Satz liefert zusätzliche Informationen über den im ersten Satz nur grob beschriebenen Prozess. Die beiden Sätze *B1* und *B2* transportieren dagegen konkurrierende Informationen. Die zugehörigen Relationen der RST sind in diesem Zusammenhang *Elaboration* für *A1* und *A2* respektive *Contrast* für *B1* und *B2*. Die Sätze, also die zu verknüpfenden Segmente, sind dabei die Argumente dieser Relation. In der RST wird dabei zwischen den zwei Argumenttypen *Nukleus (N)* und *Satellit (S)* unterschieden. Als Nukleus fungiert der dominantere Relationspartner, wohingegen der untergeordnete die Rolle des Satelliten einnimmt. Bei beiden Satzpaaren ist jeweils der erste Satz der Nukleus und der zweite ist folgerichtig der Satellit. In nicht-binären Relationen können auch mehrere Nuklei und Satelliten auftreten.

*Nuklei und Satelliten*

Im Kontext der Natural Language Generation werden in der Regel die Nachrichten der Dokumentplanungsphase als Argumente der Relationen angesehen. Die in der NLG am häufigsten verwendeten Diskursrelationen der Theorie rhetorischer Strukturen sind:

*Gängige Diskursrelationen*

- *Sequence:* Eine Menge von Nachrichten, die zeitlich oder räumlich einer bestimmten Reihenfolge folgen,
- *Elaboration:* Eine Nachricht oder eine Gruppe von Nachrichten (S) verfeinert die Informationen anderer Nachrichten (N),
- *Contrast:* Eine Nachricht oder eine Gruppe von Nachrichten (S) bietet gegensätzliche Informationen in Bezug auf andere Nachrichten (N),
- *Result:* Eine Nachricht oder eine Gruppe von Nachrichten (S) beschreibt die Konsequenz der Informationen anderer Nachrichten (N) und
- *Background:* Eine Nachricht oder Gruppe von Nachrichten (S) beschreiben Hintergrundinformationen zu Informationen anderer Nachrichten (N).

Diese informale Beschreibung birgt Interpretationsspielraum. Die Entwickler der RST geben jedoch in [119] für jede Relation eine semi-formale Spezifikation an. Eine solche ist in Abbildung 3.18 dargestellt. Wie die Abbildung zeigt, gliedert sich die Definition einer Relation in vier Teile, namentlich die Beschränkungen für Nukleus, Satellit und ihre Kombination sowie der Effekt ihrer Kombination. Am oben angegebenen *A1-A2*-Beispiel ist die Art der Kombination von Nukleus und Satellit eine Menge-Element-Beziehung, da ein Prozess als Menge von fünf Schritten beschrieben wird, von denen einer die Aktivität *Flug buchen* ist.

Die rhetorischen Relationen werden in zahlreichen NLG-Systemen in Form von *Planungsoperatoren* verwendet [154, S. 103]. Dabei bilden die Beschränkungen für die zu verknüpfenden Textsegmente die Vorbedingungen und der Effekt die Nachbedingungen der Anwendung des Planungsoperators. Durch die Typisierung der einzelnen rhetorischen Relationen ist es zusätzlich möglich, den Lesefluss des Textes zu verbessern. Beispielsweise können für die *Contrast*-Relation Signalwörter oder -phrasen, wie beispielsweise „dagegen“ oder „im Gegensatz dazu“, als implizite Referenz des Satelliten auf den zugehörigen Nukleus eingesetzt werden. Im Beispiel des Satzpaares *B1* und *B2* nimmt das Wort „Normalerweise“ die Funkion des Signalworts ein.

*Planungsoperatoren*

# 4

# SIMULATION MULTI-PERSPEKTIVISCHER, DEKLARATIVER PROZESSMODELLE

Traditionell dient die Simulation von Geschäftsprozessen vor allem der Validierung von Prozessmodellen zum Designzeitpunkt, der Kostenprognose für eine Implementierung des beschriebenen Prozesses und auch der operativen Entscheidungsunterstützung zur Ausführungszeit. Neuere Anwendungen nutzen Simulationswerkzeuge beispielsweise auch, um eine Grundlage für die Evaluation von Process-Mining-Werkzeugen zu schaffen. Sowohl die Validierung von Prozessmodellen als auch entsprechender Process-Mining-Werkzeuge basieren oft auf der Analyse des vom Prozessmodell erlaubten Verhaltens. Die in Abschnitt 3.3 eingeführten Ereignisprotokolle können das Verhalten der Prozessmodelle exemplarisch beschreiben. Simulatoren werden daher in diesem Zusammenhang hauptsächlich als Generatoren für *künstliche* Ereignisprotokolle verwendet.

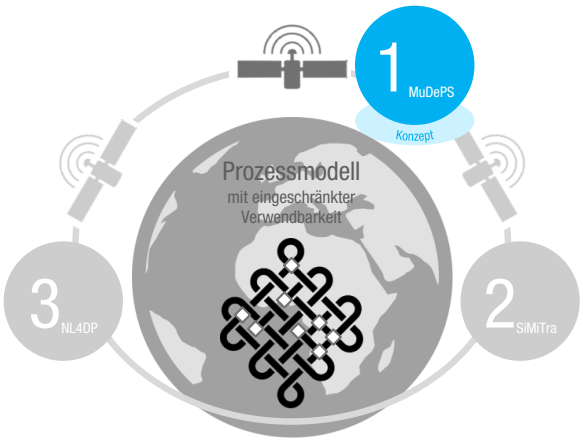



Abbildung 4.1: MuDePS: Spurgenerierung für multi-perspektivische, deklarative Prozessmodelle

Entsprechend des Fokus der vorliegenden Arbeit werden zunächst existierende Ansätze zur Spurgenerierung diskutiert (Abschnitt 4.1). Diese gestatten die Identifikation von Anforderungen für die Entwicklung *neuer* Spurgeneratoren. Keiner unterstützt jedoch die Generierung von Prozessausführungsspuren für multi-perspektivische, deklarative Prozessmodelle. In Konsequenz dazu bildet die Beschreibung eines dafür geeigneten, innovativen Ansatzes den Kern dieses Kapitels (Abschnitt 4.2). Dieser Ansatz ist einer der Kernbeiträge der vorliegenden Arbeit (Abbildung 4.1), welcher bereits im Überblick (Abschnitt 2.1) unter dem Akronym MuDePS skizziert wird.

## 4.1 VERWANDTE ARBEITEN

In der Literatur finden sich zahlreiche Beschreibungen von Simulationskonzepten für Prozessmodelle. Um eine Vergleichsbasis zu schaffen, werden daher nachfolgend ausschließlich Simulationsansätze betrachtet, welche künstliche Ereignisprotokolle produzieren. Folgende Liste gibt einen Überblick über die diskutierten Ansätze:

*Existierende Spurgeneratoren*

- *Imperativ:* Petri-Netze [36, 97, 168, 173], CPN [124], BPMN [34], Process Trees [92], YAWL [156]
- *Deklarativ:* Declare [41], DCR [122]
- *Hybrid:* Declare & CPN [193], Declare & DCR & CPN [194]

In Abschnitt 3.2.5 wird eine spezielle Klasse von Prozesssimulatoren eingeführt, die als Spurgeneratoren bezeichnet werden. Der im späteren Verlauf dieses Kapitels vorgestellte Spurgenerator basiert auf deklarativen, multi-perspektivischen Prozessmodellen. Da die deklarativen Prozessmodellierungssprachen gegenüber den imperativen vergleichsweise jung sind, ist es nachvollziehbar, dass die Anzahl der Spurgeneratoren für deklarative Prozessmodelle vergleichsweise gering ist. Aus diesem Grund werden auch Ansätze aus dem Bereich der *imperativen* Prozessmodellierung analysiert, um zu prüfen, inwiefern sich Erkenntnisse und Prinzipien auf die Spurgenerierung für deklarative Prozessmodelle übertragen lassen.

*Imperative vs. deklarative Prozessmodelle*

Die Simulation von Prozessen auf Basis *imperativer* Prozessmodelle beinhaltet das Erschließen und Protokollieren der *explizit* modellierten Pfade durch den beschriebenen Prozess. *Deklarative* Prozessmodelle beinhalten jedoch keine explizit modellierten Pfade, sondern gestatten *implizit* jeden Pfad, der nicht gegen eine Modellregel verstößt (Abschnitt 1.1.4). Damit sind Verfahren für die Simulation imperativer Modelle ad-hoc nicht auf deklarative übertragbar. Andererseits kann zum aktuellen Zeitpunkt eine Übertragbarkeit der deklarativen Modelle auf imperative Repräsentationen nicht ausgeschlossen werden [147]. Zudem ist der in Abschnitt 5.2 beschriebene Ansatz zur induktiven Übersetzung von Prozessmodellen von der Verfügbarkeit entsprechender Spurgeneratoren abhängig. Damit ist die Berücksichtigung der Simulatoren für imperative Prozessmodelle im aktuellen Kontext sinnvoll.

*Übersicht über verwandte Ansätze*

Zum späteren Vergleich mit dem in Abschnitt 4.2 beschriebenen Spurgenerator gibt Tabelle 4.1 einen Überblick über die wesentlichen Eigenschaften und Funktionen der bereits existierenden und vergleichbaren Techniken.

Die vertikale Trennung der im Tabellenkopf aufgelisteten Ansätze separiert imperative von deklarativen Spurgeneratoren. *PN* steht hier verkürzt für Petri-Netz, *PT* für Process Tree und *Hybrid* bezeichnet eine Kombination aus Declare, DCR[1] und CPN. In den inneren Zellen der Tabelle erfolgt die Bewertung, ob der jeweilige Ansatz (Spalte) die jeweilige Eigenschaft oder Funktion (Zeile) aufweist. Das ✓-Symbol repräsentiert die Unterstützung der jeweiligen Funktion respektive die Gültigkeit der Eigenschaft. Mit (✓) werden die Eigenschaften und Funktionen markiert, welche die jeweilige Technik nur teilweise aufweist. Mit „0" sind die Fälle

1 DCR = Dynamic Condition Response (Graphs)

| | PN [36] | PN [97] | PN [173] | PN [168] | PT [92] | CPN [124] | BPMN [33, 34] | YAWL [156] | Declare [41] | DCR [122] | Hybrid [193] | Hybrid [194] |
|---|---|---|---|---|---|---|---|---|---|---|---|---|
| Unterstützte Prozessperspektiven | | | | | | | | | | | | |
| – Funktionale | ✓ | ✓ | ✓ | ✓ | ✓ | ✓ | ✓ | ✓ | ✓ | ✓ | ✓ | ✓ |
| – Verhaltensorientierte | ✓ | ✓ | ✓ | ✓ | ✓ | ✓ | ✓ | ✓ | ✓ | ✓ | ✓ | ✓ |
| – Datenorientierte | 0 | 0 | (✓) | 0 | 0 | ✓ | ✓ | ✓ | 0 | ✓ | (✓) | (✓) |
| – Organisatorische | 0 | 0 | (✓) | 0 | 0 | ✓ | – | ✓ | 0 | (✓) | (✓) | (✓) |
| – Operationale | 0 | 0 | 0 | 0 | 0 | – | – | 0 | 0 | 0 | 0 | 0 |
| Ereignistypen: Start/Ende | – | – | – | ✓ | ✓ | ✓ | ✓ | ✓ | – | – | – | – |
| Dynamische Beziehungen | – | – | (✓) | – | – | – | – | – | 0 | – | – | – |
| Perspektivenübergr. Beziehungen | 0 | 0 | 0 | 0 | 0 | – | – | – | 0 | – | – | – |
| Konfigurierbare Entscheidungen | ✓ | – | – | (✓) | – | ✓ | – | ✓ | – | – | ✓ | ✓ |
| Nativ garantierte Terminierung | ✓ | – | ✓ | – | ✓ | – | ✓ | (✓) | – | ✓ | – | (✓) |
| Wahlweise künstliches Rauschen | | | | | | | | | | | | |
| – Zusätzliche Aktivitäten | ✓ | – | ✓ | ✓ | (✓) | – | ✓ | – | – | – | – | – |
| – Überspringen von Aktivitäten | ✓ | – | ✓ | ✓ | ✓ | – | ✓ | – | – | – | – | – |
| – Verzögerung von Aktivitäten | – | – | ✓ | – | – | – | – | – | – | – | – | – |
| – Vertauschen von Aktivitäten | – | – | – | ✓ | – | – | ✓ | – | – | – | – | – |
| – Unautorisierte Ressourcen | – | – | ✓ | – | – | – | – | – | – | – | – | – |
| – Verfälschen von Datenwerten | – | – | – | – | – | – | ✓ | – | – | – | – | – |
| – Rauschprofile | – | – | – | – | – | – | ✓ | – | – | – | – | – |
| Konfigurierbare Prozessumgebung | – | – | ✓ | – | – | – | – | ✓ | – | – | – | – |
| Konfigurierbarer Startzustand | – | – | – | – | – | – | – | ✓ | – | – | – | – |
| Ressourcen-Bias | – | – | – | – | – | – | – | – | – | ✓ | – | – |
| Export in Standardformat | ✓ | ✓ | ✓ | ✓ | ✓ | ✓ | ✓ | ✓ | ✓ | – | (✓) | (✓) |

Tabelle 4.1: Eigenschaften und Funktionen existierender Spurgeneratoren

markiert, in denen die Funktion oder Eigenschaft für die zugrundeliegende Prozessmodellierungssprache irrelevant ist. Beispielsweise unterstützten Petri-Netze lediglich den Kontrollfluss, wodurch nicht der Simulationsansatz, sondern die verwendete Sprache limitiert ist. Schließlich werden mit „–“ die Fälle markiert, in denen eine Eigenschaft oder Funktion im jeweiligen Ansatz nicht gegeben ist.

Die betrachteten Eigenschaften und Funktionen dienen auch gleichzeitig als Anforderungskatalog, da diese im Kontext bereits existierender, vergleichbarer Techniken als wichtig erachtet werden. Die aus den verwandten Arbeiten identifizierten Eigenschaften und Funktionen sind: *Anforderungskatalog*

- *Unterstützte Prozessperspektiven:* Vollständigkeit der Simulation der interagierenden Entitäten des Prozesses hinsichtlich der fünf Prozessperspektiven (Abschnitt 1.1.2),
- *Ereignistypen: Start/Ende:* Berücksichtigung eines einfachen Aktivitätslebenszyklus,
- *Dynamische Beziehungen:* Gültige Instanzen der Beziehungen nur zur Laufzeit ermittelbar (beispielsweise *binding of duties* in Abschnitt 3.1.5),
- *Perspektivenübergreifende Beziehungen:* Beziehungen und Regeln, die nur für Teilmengen der einzelnen Entitätsklassen gelten (Abschnitt 3.1.5),
- *Konfigurierbare Entscheidungen:* Kontrolle für Entscheidungen an z.B. XOR-Verzweigungen (imperativ) oder für Aktivitätspriorisierung (deklarativ),
- *Nativ garantierte Terminierung:* Kein künstliches Kriterium für garantierte Terminierung notwendig, besonders für deklarative Prozessmodelle relevant,

- *Wahlweise künstliches Rauschen:* Gezielte Generierung von zum Prozessmodell nonkonformen Ausführungsspuren. Rauschprofile bilden eine vorkonfigurierte Kombination einzelner Rauschparameter,
- *Konfigurierbare Prozessumgebung:* Kontrolle über z.B. Ankunftsraten und Durchlaufzeiten von simulierten Prozessinstanzen,
- *Konfigurierbarer Startzustand:* Wahlweise Vorgabe einer partiellen Ausführungsspur und Simulation möglicher Fortsetzungen,
- *Ressourcen-Bias:* Unterschiedliche Spuren basierend auf Ressourcenprofilen, die unterschiedliche Verhaltensneigungen im Prozess beschreiben,
- *Export in Standardformat:* Generierung der Spuren in gängigen Standardformaten MXML oder XES.

Zwar werden einige der genannten Eigenschaften von mehreren der aufgeführten, verwandten Techniken unterstützt, eine fundierte Anforderungsanalyse ist in allen Fällen jedoch nicht erkennbar. Folglich muss der Übernahme der Eigenschaften und Funktionen zumindest eine grobe kritische Betrachtung vorausgehen.

*Kritische Betrachtung der Eigenschaften*

Möchte man jedes mögliche Verhalten eines Modells zur Laufzeit simulieren, so müssen alle Aspekte desselben berücksichtigt werden können. Der wesentliche Grund ist, dass mittels Simulation mögliche Prozessverläufe zum Vorschein gebracht werden sollen. Je nach Fragestellung können sich diese Verläufe beispielsweise in der Reihenfolge von Aktivitäten, in Datenwerten, in der Zuweisung von Aktivitäten zu Ausführenden oder auch der Verwendung von IT-Systemen unterscheiden. Aus diesem Grund ist die Unterstützung von zumindest den fünf vorab beschriebenen, auf Erfahrungswerten basierenden Prozessperspektiven (Abschnitt 1.1.2) sinnvoll. Eine Prüfung der Vollständigkeit der Abdeckung jeder *einzelnen* Perspektive ist ebenfalls problematisch. Rein qualitativ ist die Abdeckung durch einen Spurgenerator dann vollständig, wenn selbiger die volle Ausdrucksmächtigkeit der zur Prozessmodellierung verwendeten Sprache ausschöpft. Dies kann folglich nur für jede Prozessmodellierungssprache individuell geprüft werden. Der im Rahmen dieser Arbeit entwickelte Spurgenerator operiert auf DPIL-Modellen. Da DPIL Regelschachtelungen beliebiger Tiefe gestattet, erfolgt die Prüfung der Vollständigkeit der Abdeckung der Perspektiven durch den Spurgenerator auf Basis gängiger Regelschablonen (Abschnitt 4.2.4.3). Verwandte Ansätze werden dagegen jeweils auf Basis der Ausdrucksmächtigkeit der Prozessmodellierungssprache bewertet, da diese Analyse auch für den später vorgestellten Translationsansatz relevant ist (Abschnitt 5.2).

*Unterstützte Prozessperspektiven*

Die Unterscheidung der Ereignistypen ermöglicht eine detailliertere Wiedergabe möglicher Prozessverläufe und ist somit ebenfalls eine nützliche Funktionalität. Dynamische und perspektivenübergreifende Beziehungen können nicht in jeder Prozessmodellierungssprache abgebildet werden, in DPIL allerdings schon. Eine Simulationswerkzeug für DPIL muss demnach auch in der Lage sein, die Regeln für diese Beziehungen zu interpretieren.

*Ereignistypen: Start/Ende*

*Dynamische/ perspektivenübergreifende Beziehungen*

Konfigurierbare Entscheidungen bedeuten im DPIL-Kontext die Priorisierung von Aktivitäten. Für manche Anwendungsfälle ist eine solche Priorisierung notwendig, um realitätsnähere Ereignisprotokolle zu generieren, was mit einer Gleichver-

*Konfigurierbare Entscheidungen*

teilung aller Aktivitäten nicht immer möglich ist. Imperative und deklarative Prozessmodelle enthalten zudem potentiell Möglichkeiten der unbegrenzten Wiederholung von Aktivitäten. Für die Terminierung der Simulation ist dann ein künstliches Terminierungskriterium obligatorisch.

*Nativ garantierte Terminierung*

Ausschließlich aus einem Ereignisprotokoll selbst zu schlussfolgern, welches beobachtete Verhalten relevant respektive irrelevant ist, ist fraglich. Die Relevanz wird in der einschlägigen Literatur vordergründig von der *Häufigkeit* der Beobachtung eines bestimmten Prozessverhaltens abhängig gemacht. Diese rein quantitative Sichtweise gibt jedoch weder Aufschluss über die Wichtigkeit noch über die Korrektheit des jeweiligen Prozessverlaufs. Für einen solchen Schritt ist ein detailliertes Wissen über die Qualität der Ereignisprotokolle obligatorisch. Andernfalls ist die Qualität der Schlussfolgerungen diffus, da sie von beispielsweise Umfang, Vollständigkeit und Herkunft der Protokolle abhängig ist. Zudem entsteht durch das Ausblenden von selten auftretenden Prozessverläufen bereits ein Bias hinsichtlich der Motivation des Schließens von beobachtetem Verhalten auf allgemeingültige Verhaltens*vorgaben*. Ist die Motivation der Interpretation der Protokolle die Modellierung der bisher am häufigsten beobachteten Prozessverläufe, dann kann ein solches Ausblenden aber auch sinnvoll sein. In diesem Zusammenhang werden selten beobachtete Prozessverläufe im Kontext von historischen Ereignisprotokollen häufig als Rauschen bezeichnet (Abschnitt 3.3.2). Für künstliche Ereignisprotokolle hat dieses Rauschen jedoch eine andere Relevanz. Rauschen bedeutet hier eine gewünschte, nachvollziehbare Abweichung vom Modell oder das Ausblenden bestimmter Modellbestandteile. Diese Funktionalität kann beispielsweise verwendet werden, um die Wirksamkeit der Rauschfilter von Process-Mining-Verfahren zu testen. Aus Sicht des Autors ist aber die Anwendung im Bereich der Konformitätsprüfung (Abschnitt 3.4.1) plausibler. Künstlich erzeugtes Rauschen kann hier verwendet werden, um die Korrektheit der Funktionsweise von Werkzeugen für die Konformitätsprüfung zu bewerten.

*Wahlweise künstliches Rauschen*

Eine konfigurierbare Prozessumgebung, wie beispielsweise die Ankunftsrate neuer Prozessinstanzen ist von weit geringerer Relevanz. Dies ist in erster Linie für Simulationen zur Messung bestimmter KPIs wichtig, die jedoch für den Zweck der Spurgenerierung im Rahmen der vorliegenden Arbeit irrelevant sind. Da jedoch zum aktuellen Zeitpunkt nicht ausgeschlossen werden kann, dass das in Abschnitt 4.2 vorgestellte Simulationskonzept später auch für diese Zwecke verwendet werden könnte, wird diese Funktionalität exemplarisch berücksichtigt.

*Konfigurierbare Prozessumgebung*

Ein konfigurierbarer Startzustand ist beispielsweise für eine Anwendung des Spurgenerators im Bereich der Vorhersage von Konsequenzen der Entscheidungen in einem Prozess von zentraler Bedeutung.

*Konfigurierbarer Startzustand*

Um Prozessverläufe in Abhängigkeit der Charakteristika bestimmter organisatorischer Ressourcen bei der Spurgenerierung beeinflussen zu können, ist die Konfigurationsmöglichkeit für einen Ressourcen-Bias sinnvoll. Beispielsweise kann ein Nutzer mit geringer Arbeitsmotivation modelliert werden, welcher versucht, bestimmte Aufgaben zu vermeiden oder allgemein danach strebt, jeden Prozess schnellstmöglich zu beenden.

*Ressourcen-Bias*

Der Export der generierten Prozessausführungsspuren in ein Standardformat ist von zentraler Bedeutung für die Verwendung dieser in anderen Werkzeugen. Bei-

*Export in Standardformat*

spiele dafür sind die bereits erwähnten Process-Mining-Technologien und Werkzeuge für die Konformitätsprüfung.

In den nachfolgenden Abschnitten werden die in Tabelle 4.1 dargestellten Techniken hinsichtlich ihrer Eigenschaften und Funktionen zusammengefasst. Der Schwerpunkt wird hier auf eine detaillierte Betrachtung der Spurgeneratoren für deklarative Prozessmodellierungssprachen gelegt.

### 4.1.1 *Prozesssimulation auf Basis imperativer Prozessmodelle*

Der aktuelle Abschnitt beschränkt sich auf einen knappen Überblick über Spurgeneratoren für unterschiedliche *imperative* Prozessmodellierungssprachen. Ziel ist, die Funktionsweise soweit zu analysieren, dass eine Anwendung der Techniken im später beschriebenen Translationsprinzip für Prozessmodelle (Abschnitt 5.2) möglich wird. Weiterhin wird geprüft, inwiefern sich Konzepte dieser Techniken für den im Rahmen der vorliegenden Arbeit entwickelten Simulationsansatz (Abschnitt 4.2) wiederverwendet werden können.

#### 4.1.1.1 *Petri-Netze*

*Spurgenerierung aus Petri-Netzen*

Der mit Abstand größte Anteil an Spurgeneratoren operiert auf Petri-Netzen[2] und beschränkt sich für die Generierung von Ereignisprotokollen auf die Funktionale und die Verhaltensorientierte Perspektive. Burratin und Sperduti [36] beschreiben eine Technik zur parametrierbaren Erzeugung von Petri-Netzen und deren Ausführung zur Generierung künstlicher Ereignisprotokolle. Da die automatische Generierung von Prozessmodellen außerhalb des Fokus der vorliegenden Arbeit liegt, beschränkt sich die folgende Erläuterung auf den Spurgenerierungsanteil. Der Spurgenerator folgt schrittweise jedem Pfad durch das Modell und protokolliert dabei die besuchten Aktivitäten. Mehrere Pfade existieren nur dann, wenn das Modell Verzweigungen enthält. Da Petri-Netze grundsätzlich nur zwischen *exklusiven Entscheidungen (XOR)* und *Parallelisierung (AND)* des Kontrollflusses unterscheiden können[3], muss der Spurgenerator lediglich an XOR-Verzweigung entscheiden, welchem Pfad zu folgen ist. Der Ansatz unterstützt hierfür eine zufallsbasierte Entscheidung auf Basis einer Gleich-, Standardnormal- oder Beta-Verteilung (Tabelle 3.3). Eine Besonderheit ist, dass für AND-Verzweigungen auch jeweils eine dieser Wahrscheinlichkeitsverteilungen angegeben werden kann. Damit wird jedoch lediglich eine zeitliche Reihenfolge festgelegt, in der die parallelisierten Pfade durchlaufen werden. Eine zusätzliche Restriktion ist, dass jedem von einer Verzweigung ausgehenden Pfad eine Wahrscheinlichkeit von $> 0$ zugewiesen wird. Einerseits vermeidet das „tote" Pfade. Bei Schleifen andererseits – modelliert mittels XOR-Verzweigungen – läuft die Wahrscheinlichkeit für endlose Wiederholungen gegen $0$, wodurch der Algorithmus garantiert terminiert. Die besuchten

*Probabilistische Modellierung von Entscheidungen*

2 In einigen Fällen handelt es sich um sogenannte Workflow-Netze, eine spezielle Form der Petri-Netze. Da diese sich im Wesentlichen nur durch definierte Anfangs- und Endplätze von allgemeinen Petri-Netzen unterscheiden, werden die beiden Formen im Rahmen dieser Arbeit als identisch angesehen.

3 *Inklusive (OR)* Entscheidungen können aber durch eine Kombination aus XOR- und AND-Verzweigungen nachgebildet werden.

Aktivitäten jeder Ausführung werden schließlich mit einem Zeitstempel versehen und in einem MXML-Ereignisprotokoll abgelegt. Der Algorithmus ist zusätzlich in der Lage, Aktivitäten zufallsbasiert zu löschen oder als künstliches Rauschen ins Protokoll aufzunehmen. Künstliches Rauschen dient dazu, die generierten Spuren realitätsnäher zu gestalten, indem auf Wunsch Unregelmäßigkeiten eingefügt werden. Da diese Simulationstechnik vollständig von der angenommenen Petri-Netz-Struktur ausgeht, ist sie nicht auf Simulationsprobleme für multi-perspektivische, deklarative Prozessmodelle anwendbar. Das Prinzip, gültige Ereignisprotokolle auf Wunsch mit nicht-modellkonformen Prozessausführungsspuren anzureichern, wird in Abschnitt 4.2 aufgegriffen.

*Anreicherung um invalide Prozessausführungsspuren*

Der in [97] beschriebene Ansatz erzeugt auf Basis von gegebenen Petri-Netzen neue Petri-Netze sowie dazu konforme Ereignisprotokolle. Die Generierung künstlicher Prozessmodelle ist jedoch nicht im Fokus der vorliegenden Arbeit, weswegen auch hier auf eine Erläuterung dieses Teils der Funktionalität verzichtet wird. Die Simulationsdurchführung basiert auf Graphersetzungsregeln. Die wichtigste Regel beschreibt die Aktivierung von Transitionen und den damit verbundenen Fluss der Markierungen. Ein Simulationsfortschritt wird dadurch erzielt, dass ein unmarkierter Platz durch einen ein- oder mehrfach markierten ersetzt wird. Welche Plätze ersetzt werden, entscheidet der aktuelle Markierungszustand und die dadurch ermöglichten Aktivierungen von Transitionen. In der Beschreibung des Ansatzes wird jedoch keine Aussage getroffen, inwiefern die entwickelten Graphersetzungsregeln auf Petri-Netze mit AND- respektive XOR-Verzweigungen anwendbar sind. Weiterhin wird nicht eindeutig erwähnt, in welchem Serialisierungsformat die erzeugten Ereignisprotokolle exportiert werden. Der Ansatz ist von den Autoren selbst als unfertig beschrieben, sodass keine Rückschlüsse auf die Vorteile der Verwendung von Graphgrammatiken zur Simulationsdurchführung gegenüber alternativen Verfahren gezogen werden können. Aus diesem Grund wird das vorgestellte Simulationsprinzip als nicht relevant für den Rahmen der vorliegenden Arbeit betrachtet.

*Graphersetzungsregeln*

In [173] wird zwar allgemein von sogenannten Prozessmodellen mit Blockstruktur ausgegangen, die zugehörige Implementierung ist jedoch auf Petri-Netze beschränkt. Diese Beschränkung wird insofern gelockert, als die zugehörige Simulationssoftware gestattet, den sogenannten Kontext der Prozessausführung zu modellieren. Je nach gewähltem Modus wird ist Kontext entweder leer (SIMPLE-Modus) oder wird mit einer Kombination aus Nutzer- und Zugriffsmodell sowie einem Modell für die Datenverwendung von und in Aktivitäten befüllt. Folglich enthalten die generierten Spuren entweder nur Aktivitätsnamen und zugehörige Zeitstempel oder zusätzlich die ausführende Ressource und Datenelemente. Im Unterschied zu vergleichbaren Ansätzen werden Ressourcenzuweisung auf Basis von Autorisierungsfunktionen vorgenommen. Die derzeit unterstützten Autorisierungsfunktionen beschränken sich auf ein einfaches Rollenkonzept und eine explizite Zugriffsliste. Hierarchische organisatorische Beziehungen, wie beispielsweise die Angestellter-Vorgesetzter-Beziehung, lassen sich hier nicht umsetzen. Wie in [36] können auch mit der Technik aus [173] Ausführungsspuren erzeugt werden, welche einen Verstoß gegen das gegebene Modell und zusätzlich gegen den spezifizierten Kontext darstellen. Dies geschieht nach der eigentlichen Simulation des Eingabe-Petri-Netzes in Form von Spurtransformationen. Beispiele für die kon-

*Spurgenerierung aus Prozessmodellen mit Blockstruktur*

*Autorisierungsfunktionen*

trollierte Generierung von Abweichungen sind zeitliche Verzögerungen, das Überspringen von Aktivitäten und die nicht autorisierte Zuweisung einer Ressource zu einer Aktivität. Als Besonderheit können zusätzlich auch Verstöße gegen Regeln wie *binding* oder *separation of duties* simuliert werden (Abschnitt 3.1.5). Die Ausgabe der Ereignisprotokolle erfolgt entweder als reiner Text oder in MXML-Form; eine XES-Exportfunktion ist geplant. Die Simulationsdurchführung basiert erneut ausschließlich auf der Petri-Netz-Notation. Damit ist auch diese Technik nicht auf die Simulation multi-perspektivischer, deklarativer Prozessmodelle übertragbar. Das Konzept der Autorisierungsfunktionen ist jedoch wiederverwendbar und wird in Abschnitt 4.2 durch die Verwendung der Prozessmodellierungssprache DPIL implizit berücksichtigt. Eine nachträgliche, gezielte Verfälschung berechneter Ausführungsspuren ist unnötig, da der in Abschnitt 4.2 beschriebene Ansatz zur Generierung von Spuren mit künstlichem Rauschen eigenständig in der Lage ist.

Shugurov und Mitsyuk [168] beschreiben ebenfalls einen Ansatz für die Generierung künstlicher Ereignisprotokolle auf Basis von rein kontrollflussbasierten Petri-Netzen. Wie in [36] und [173] ermöglicht der Spurgenerator auch die Erzeugung von Spuren, welche gezielt künstliches Rausche beinhalten. Dies können zusätzliche Aktivitäten, Missachtungen der Reihenfolge von Aktivitäten oder übersprungene Aktivitäten sein. Im Unterschied zu [36] können die Anzahl der Spuren sowie die maximale Anzahl an Ereignissen pro Spur parametriert werden. Letzteres stellt ein künstliches Abbruchkriterium dar, welches für den gewählten Ansatz notwendig ist, falls ein Prozessmodell Endlosschleifen gestattet. Das trifft im aktuell diskutierten Ansatz potentiell zu. Hier, wie auch in [36], werden Wahrscheinlichkeiten verwendet, um die nächste Transition auszuwählen und diese Wahrscheinlichkeiten auch 0 sein dürfen. In [36] müssen keine künstlichen Abbruchkriterien definiert werden, da Wahrscheinlichkeiten von 0 explizit verboten sind. Die im aktuell betrachteten Ansatz verwendete Wahrscheinlichkeitsverteilung für die Zufallszahlen wird nicht näher spezifiziert. Eine weitere Besonderheit ist die Berücksichtigung des Aktivitätslebenszyklus. So kann im Ereignisprotokoll zwischen dem Start und dem Ende der Bearbeitung einer Aktivität unterschieden werden. Die Technik ist als Erweiterung der bereits erwähnten ProM-Plattform implementiert und erlaubt somit beispielsweise den Export des erzeugten Ereignisprotokolls im XES-Format. Auch dieser Ansatz enthält, obwohl er ebenfalls auf imperativen Modellen basiert, einen relevanten Gesichtspunkt. Die Berücksichtigung des Aktivitätslebenszyklus erlaubt die Berechnung der *Dauer* der Durchführung einer Aktivität. Das wird von dem in Abschnitt 4.2 beschriebenen eigenen Ansatz ebenfalls unterstützt.

*Künstliches Abbruchkriterium: Länge der Spuren*

*Berücksichtigung des Aktivitätslebenszyklus*

#### 4.1.1.2 *Prozessbäume*

Wie in den Arbeiten von Burattin, Sperduti [36] und Kataeva [97] bietet auch die in [92] beschriebene Technik die Möglichkeit, sowohl Prozessmodelle als auch zufällige Ereignisprotokolle für diese Prozessmodelle zu generieren. Prozessmodelle werden hier in Form von Prozessbäumen repräsentiert und weisen demnach eine Blockstruktur auf. Die Prozessbäume werden auf Basis von den in [160] identifizierten kontrollflussbasierten Workflow-Mustern wie Schleifen und verschiedenen Verzweigungsformen generiert. Im Unterschied zu den anderen in diesem Abschnitt beschriebenen Ansätzen können hier zusätzliche Kontrollflussmuster realisiert wer-

den. Ein Beispiel sind die sogenannten *langfristigen Abhängigkeiten*, welche eine Abhängigkeit zwischen zwei XOR-Entscheidungen darstellt. Diese können aufgrund der Blockstruktur von Prozessbäumen eigentlich nicht modelliert werden, da die einzelnen Zweige per Definition voneinander unabhängig sind. Stattdessen wird auf einen Umweg zurückgegriffen, der Teilbäume gezielt kopiert, sodass die entstehenden Duplikate die gewünschte Abhängigkeit repräsentieren. An dieser Stelle ist anzumerken, dass die Wahl der Prozessmodellierungssprache und das eben angesprochene Alleinstellungsmerkmal der Technik einen unnötigen Widerspruch darstellen. Die Erzeugung der künstlichen Prozessbäume ist wiederum außerhalb des Fokus der vorliegenden Arbeit.

Die in [92] vorgestellte Technik zur Spurgenerierung profitiert insofern von der Eigenschaft der Azyklizität von Prozessbäumen, als keine künstlichen Abbruchkriterien für die Simulation notwendig sind. Die Terminierung ist bereits durch die Baumstruktur garantiert. Somit wird für die Simulationsdurchführung lediglich die Gesamtanzahl der zu simulierenden Instanzen benötigt. Ähnlich wie bei zuvor diskutierten Techniken (beispielsweise [168]), so wird auch hier zwischen Beginn und Ende der Durchführung einer Aktivität unterschieden. Der Prozessbaum wird mit einer festen Präordnung traversiert und in eine Sequenzform überführt. Sequenzen von Aktivitäten werden exakt in der Traversierungsreihenfolge angeordnet. Für Aufteilungen des Kontrollflusses durch XOR-, AND-, OR- und Schleifen-Verzweigungen werden die jeweiligen Nachfolgeknoten entsprechend der Semantik des Operators in eine sequentielle Reihenfolge gebracht. Ob eine Schleife abgebrochen oder erneut durchlaufen wird und welchem Pfad bei einer Entscheidung gefolgt wird, entscheidet der Algorithmus zufällig. Es wird keine Aussage darüber getroffen, auf welcher *Grundlage* diese Entscheidung gefällt wird. Wie einige der anderen diskutierten Techniken auch, können gezielt Ausführungsspuren generiert werden, welche Verstöße gegen das jeweilige Prozessmodell aufweisen. Zu den unterstützten Verstößen zählen das Überspringen oder irregulär-doppelte Ausführungen von Aktivitäten. Die Serialisierung des Ereignisprotokolls erfolgt im XES-Format. Für den in der vorliegenden Arbeit vorgestellten Ansatz zur Spurgenerierung für multi-perspektivische, deklarative Prozessmodelle ist die eben beschriebene, rein imperative und kontrollflussbasierte Technik nicht relevant. Für den in Abschnitt 5.2 beschrieben Ansatz zur Übersetzung von Prozessmodellen stellt sie jedoch die einzige Möglichkeit dar, Prozessbäume induktiv in eine andere Prozessmodellierungssprache zu übersetzen.

#### 4.1.1.3 *CPN-Modelle*

*Generierung aus CPN-Modellen*

Der in [124] vorgestellte Ansatz erzeugt Ereignisprotokolle für gegebene Coloured Petri Nets und baut auf der generellen CPN-Semantik auf [90]. Das Werkzeug CPN Tools bietet für die Simulationsdurchführung einen interaktiven und einen automatischen Modus. Für die automatisierte Erzeugung von Ereignisprotokollen ist lediglich letzterer Modus relevant. Um die Terminierung der Simulation zu garantieren, muss ein künstliches Abbruchkriterium festgelegt werden. Im Gegensatz zu der in [168] diskutierten Simulationstechnik kann dieses Abbruchkriterium nicht nur eine maximale Anzahl durchgeführter Aktivitäten, sondern auch eine maximale Laufzeit des Prozesses sein. Ausgehend von der initialen Markierung

*Künstliches Abbruchkriterium: Maximale Prozesslaufzeit*

und den festgelegten Abbruchkriterien wird auf Basis des globalen Modelltaktes der Fortschritt der Prozessausführung simuliert. In jedem Takt werden auf Basis der entsprechenden Markierungen die aktuell aktivierten Transitionen berechnet. Sollten von einem Platz aus mehrere, aktuell aktive Transitionen ausgehen, wird eine davon zufallsgesteuert ausgewählt. Die Belegung der Markierungen mit entsprechenden Datenwerten erfolgt ebenfalls durch einen Zufallszahlengenerator. Diese Zufallszahlen werden nach einer der möglichen parametrierbaren Wahrscheinlichkeitsfunktion ermittelt. Zu den von CPN Tools unterstützen Funktionen zählen beispielsweise die Gleich-, Standard-Normal-, Erlang- und Exponentialverteilung (vgl. Tabelle 3.3). Der in [124] beschriebene Simulator basiert auf der generellen Simulationsfunktionalität von CPN Tools, welches zwar die beschriebene automatisierte Ausführung von CPN-Modellen, nicht aber die Protokollierung des Ausführungsverlaufs unterstützt. Diese Funktionalität wird in [124] mittels weiterer CPN-ML-Funktionen als Modifikationen der von CPN Tools standardmäßig unterstützten Eingabe-, Ausgabe- und Aktionsbeschriftungen realisiert. Da CPN Tools sowohl Zeitstempel, Aktivitätsdurchführungen und Ressourcenzuweisungen auf Basis eines ebenfalls zu modellierenden Ressource-Rollenmodells erlaubt, ist lediglich eine Überführung dieser Informationen in das gewünschte Format notwendig. Der beschriebene Ansatz erzeugt daher für jeden bearbeiteten Fall eine Prozessausführungsspur im MXML-Format. Das finale Ereignisprotokoll wird durch die Verschmelzung aller dieser Spuren mittels des Werkzeugs ProM generiert. Wie die vorher diskutierten Ansätze beruht auch der aktuell betrachtete Simulationsansatz auf der Verfügbarkeit des Prozessmodells als gerichteter Graph. Aus diesem Grund ist der Ansatz prinzipiell nicht auf deklarative Prozessmodelle anzuwenden. Die Simulation unter Berücksichtigung eines Ressource-Rollenmodells wird jedoch auch von der in Abschnitt 4.2 vorgestellten Simulationstechnik für multi-perspektivische, deklarative Prozessmodelle unterstützt. Zudem ist die aktuell diskutierte Simulationstechnik durch ihren multi-perspektivischen Charakter sowie programmatische Erweiterungspunkte und zahlreiche Möglichkeiten der Parametrierung für die Simulation von imperativen Prozessmodellen im Rahmen der induktiven Translation geeignet (Abschnitt 5.2).

*Ressource-Rollenmodell*

*Programmatische Erweiterungspunkte*

#### 4.1.1.4 *BPMN-Modelle*

In [34] und [33] wird der in [36] vorgestellte Ansatz zur Erzeugung von Petri-Netz-basierten Prozessmodellen und zugehöriger Ereignisprotokolle signifikant erweitert. Eine der wesentlichen Neuerungen ist dabei die Unterstützung der Datenorientierten Perspektive (Abschnitt 1.1.2). Aktivitäten können hier Datenproduzenten oder -konsumenten sein. Weiterhin wird zwischen statischen und dynamischen Datenobjekten unterschieden. Erstere sind einfache Schlüssel-Wert-Paare, die den kompletten Simulationsverlauf über unverändert bleiben. Letztere können zu jedem Zeitpunkt des Simulationsverlaufs einen anderen Wert annehmen. Weiterhin gestattet eine neue interne Struktur mehr Flexibilität, sodass es beispielsweise möglich ist, mit externen Werkzeugen erstellte BPMN-Modelle zu importieren und auf eine Petri-Netz-Repräsentation abzubilden. Die Simulation von Entscheidungen auf Basis konfigurierbarer Wahrscheinlichkeitsverteilungen, wie es im Vorläufer [36] möglich ist, wird derzeit nicht unterstützt. Stattdessen ist an jeder Verzweigung die

*Spurgenerierung aus BPMN-Modellen*

Wahl eines der ausgehenden Pfade gleich wahrscheinlich gegenüber den Alternativen. Weiterhin wird eine Unterscheidung zwischen dem Start und dem Ende der Bearbeitung einer Aktivität unterstützt, ist aber nicht obligatorisch. Dabei wird jedoch von einer rein sequentiellen Ausführung ausgegangen, was bedeutet, dass zwischen Beginn und Ende der Durchführung einer Aktivität in derselben Prozessinstanz kein Ereignis auftreten darf, welches sich auf eine andere Aktivitätsdurchführung bezieht. Statische Datenobjekte werden vom Simulator für jede simulierte Instanz als Konstanten gehandhabt. Für dynamische Datenobjekte kann ein Python-Skript hinterlegt werden, sodass flexibel jede beliebige Belegungsstrategie implementiert werden kann. Dabei kann für jede Instanz eine andere Strategie gewählt werden. Zusätzlich ist es möglich, künstliches Rauschen im Ereignisprotokoll hervorzurufen. Das kann auf Spur-, Ereignis- oder Datenebene geschehen. Auf Spurebene können so künstliche Unregelmäßigkeiten wie „abgeschnittene" Teilspuren oder zufällige, nicht im Modell enthaltenen Aktivitäten generiert werden. Auf Ereignisebene kann beispielsweise der Aktivitätsname und auf Datenebene der Datenwert zufällig verändert werden. Dabei muss vom Nutzer nur die maximale Höhe der Änderung für die Datenwerte angegeben werden. Aufgrund der Vielfalt an Möglichkeiten, wie sich das Rauschphänomen im Ereignisprotokoll manifestieren kann, werden mehrere sogenannte *Rauschprofile* vordefiniert. Diese stellen gebräuchliche Konfigurationen zum Einfügen von Rauschen dar. Der Export der generierten Ereignisprotokolle erfolgt wahlweise im MXML- oder XES-Format. Für den in Abschnitt 5.2 beschriebenen Ansatz ist die hier besprochene Technik insofern relevant, als sie als multi-perspektivischer Spurgenerator für die Übersetzung von BPMN-Modellen dienen kann. Für die in Abschnitt 4.2 beschriebene Technik zur multi-perspektivischen Spurgenerierung für deklarative Prozessmodelle ist hingegen lediglich die Bereitstellung von Rauschprofilen relevant.

*Rauschprofile*

#### 4.1.1.5 *YAWL-Modelle*

Der im imperativen Bereich letzte Ansatz zur Spurgenerierung verwendet die Prozessmodellierungssprache YAWL und ist in [156] beschrieben. Grundlegende Unterschiede zu den vorangegangenen vergleichbaren Ansätzen liegen in der Unterscheidung zwischen der Verfügbarkeit, dem Start der Bearbeitung und dem zugehörigen Ende der Bearbeitung einer Aktivität im Ereignisprotokoll. In den übrigen Ansätzen werden lediglich das Ende und teilweise der Start einer Aktivitätsdurchführung protokolliert. Zusätzlich zu Funktionaler und Verhaltensorientierter werden auch die Organisatorische und die Datenorientierte Prozessperspektive unterstützt, wodurch die Abdeckung der Perspektiven mit [124] identisch ist. Ein weiterer wesentlicher Unterschied liegt jedoch in der Art und Weise, wie das Simulationsmodell initialisiert wird. Anstatt das Simulationsmodell von Grund auf vom Durchführenden der Simulation konfigurieren zu lassen, werden Informationen aus einem WfMS verwendet. Der Ansatz geht davon aus, dass in selbigem *Design-Informationen* in Form eines Prozessmodells mit Abdeckung der angesprochenen Perspektiven, *historische Informationen* in Form früherer Ereignisprotokolle sowie *Zustandsinformationen* über aktuelle Prozessausführungen vorliegen. Das ist mit dem gleichnamigen *YAWL WfMS* gegeben – zumindest wenn dieses bereits früher zur Ausführung des jeweiligen Prozessmodells eingesetzt wurde.

*Spurgenerierung aus YAWL-Modellen*

*Erweiterter Aktivitätslebenszyklus*

*Alternative Initialisierung des Simulationsmodells mit historischen oder zustandsbezogenen Informationen*

Design-Informationen in Kombination mit historischen Informationen sollen das Simulationsmodell potentiell stärker am realen Prozessablauf ausrichten als ein manuell konstruiertes. Dieser Ansatz erscheint zunächst paradox, da Simulatoren im aktuellen Kontext besonders dann eingesetzt werden, wenn die historische Informationsgrundlage *spärlich* oder gar nicht existent ist. Allerdings ist es auch bei Verfügbarkeit einer umfangreichen historischen Informationsgrundlage schwierig abzuschätzen, ob selbige weitestgehend vollständig und fehlerfrei ist. Ein Simulator kann unvollständige historische Ausführungsprotokolle komplementieren. Andererseits bietet er die Möglichkeit, ein Simulationsmodell automatisiert zu parametrieren. Das kann genutzt werden, um die Auswirkungen von Änderungen am Modell vor dem Hintergrund der Status-Quo-Daten abzuschätzen.

*Schätzung der Parameter des Simulationsmodells auf Basis historischer Informationen*

Historische Informationen werden hier zur Vorkonfiguration des Simulationsmodells verwendet [156]. Dazu zählt beispielsweise eine automatisierte, historisch fundierte Schätzung der Wahrscheinlichkeiten für die einzelnen Entscheidungen an den jeweiligen Entscheidungspunkten. Zudem werden sowohl die durchschnittlichen Bearbeitungszeiten von Aktivitäten als auch die Wertebereiche für involvierte Datenobjekte auf historischer Basis ermittelt. Damit greift das Verfahren auf keinerlei „manuelle", sondern mit historischen Daten parametrierte Wahrscheinlichkeitsverteilungen zurück, was ebenfalls einen bisher einzigartigen Unterschied zu den vorab diskutierten Ansätzen darstellt. Der aktuelle Zustand einer laufenden Prozessausführung wird verwendet, um das Simulationsmodell virtuell in denselben Zustand zu versetzen. Der Zweck der Simulation ist hierbei die *Unterstützung der operativen Entscheidungsfindung* durch die Erschließung unterschiedlicher Szenarien der Prozessfortsetzung hinsichtlich ihrer Auswirkungen auf die nahe Zukunft. Informationen, welche das YAWL WfMS hierfür zur Verfügung stellt ist der Fortschritt hinsichtlich der Abarbeitung von Aufgaben sowie die aktuell gebundenen Ressourcen. Zusammenfassend werden die statischen Design-Information zur Definition der Struktur, die dynamischen historischen Informationen zur Parametrierung und die Momentaufnahme der Zustandsinformationen zur Definition des Ausgangspunktes des Simulationsmodells verwendet. Sobald diese Informationen im YAWL-Modell kodiert worden sind, wird das Modell in eine äquivalente CPN-Repräsentation konvertiert. Die Simulationsdurchführung basiert folglich erneut auf CPN Tools [90] und der aktuelle Ausführungszustand eines Prozessmodells wird über eine von CPN Tools angebotene Schnittstelle als Initialzustand für das CPN verwendet. Dieser Initialzustand kann dabei auch leer sein, wenn die Simulation vom im Prozessmodell vorgegebenen Startpunkt aus durchgeführt werden soll. Die eigentliche Simulationsdurchführung erfolgt analog zu der in [124] beschriebenen. Als Export-Format für die erzeugten Ereignisprotokolle wird ebenfalls das MXML-Format gewählt. Das Prinzip, ein Simulationsmodell zur operativen Entscheidungsfindung zu verwenden wird von dem in Abschnitt 4.2 beschrieben Ansatz ebenfalls übernommen. Es findet dabei allerdings eine Übertragung auf das deklarative Modellierungsparadigma statt. Auch die Verwendung historischer Informationen für die Parametrierung des Simulationsmodells kann für die Simulation multi-perspektivischer deklarativer Prozessmodelle wiederverwendet werden.

*Simulation zur operativen Entscheidungsfindung*

### 4.1.2 *Prozesssimulation auf Basis deklarativer Prozessmodelle*

Dieser Abschnitt liefert einen detaillierteren Überblick über die bis dato geringe Zahl an Spurgeneratoren für unterschiedliche *deklarative* Prozessmodellierungssprachen. Im Gegensatz zu den vorab beschriebenen imperativen Spurgeneratoren können deklarative Spurgeneratoren nicht auf explizite Kontrollflussvorgaben zurückgreifen. Stattdessen müssen diese aus den impliziten, im Modell enthaltenen *Einschränkungen* ermittelt werden. Weiterhin sind beispielsweise die Aktivität-Ressourcen-Zuweisungen oder auch die Aktivität-Datenabhängigkeiten tendenziell von einer im Bereich der imperativen Modelle unbekannten Dynamik geprägt (Abschnitt 3.1). Der in Abschnitt 4.2 vorgestellte Ansatz zur Simulation multi-perspektivischer, *deklarativer* Prozessmodelle realisiert durch eine geeignete Sprachwahl diese Dynamik. Die im aktuellen Abschnitt betrachteten Ansätze bilden dabei die unmittelbare Vergleichsbasis für diese Technik und werden daher ausführlicher betrachtet.

#### 4.1.2.1 *Declare-Modelle*

Mit [41] wird eine Technik zur Simulation rein kontrollflussbasierter, deklarativer Prozessmodelle auf Basis der Sprache Declare beschrieben. Die Simulationsdurchführung selbst basiert auf *Endlichen Zustandsautomaten (EZA)*. Da Declare jedoch auf dem Logik-Kalkül LTL basiert (Abschnitt 3.1), ist zunächst eine Transformation des Declare-Prozessmodells in einen semantisch äquivalenten EZA notwendig. Dies geschieht in sechs Schritten, die in Abbildung 4.2 illustriert sind.

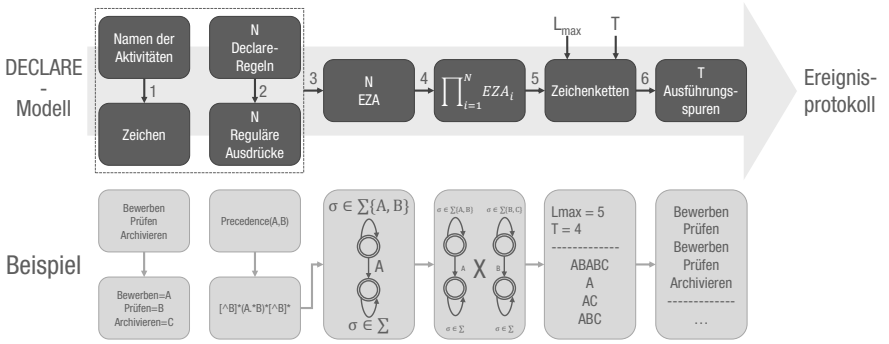



Abbildung 4.2: Ablauf der Spurgenerierung für Declare-Modelle [41]

*Schritt 1* beinhaltet die eindeutige Abbildung der Namen aller Aktivitäten des Declare-Modells auf ein einzelnes Zeichen $\sigma$ eines Alphabets $\Sigma$, beispielsweise also auf Buchstaben des lateinischen Alphabets. Dies ist notwendig, um anschließend die Regeln des Declare-Modells transformieren zu können. *Schritt 1*

In *Schritt 2* werden alle $N$ Declare-Regeln in jeweils einen *regulären Ausdruck* konvertiert. Die Verlustfreiheit dieser Konvertierung ist einerseits durch die eindeutige Abbildung der Aktivitätsnamen auf einzelne Zeichen und andererseits dadurch gewährleistet, dass Declare-Regeln lediglich auf Aktivitäten bezogene Existenz- und Reihenfolgeabhängigkeiten beschreiben. Die ersten beiden Schritte transformieren somit die in Abbildung 4.2 darge- *Schritt 2*

stellte precedence(Bewerben, Prüfen)-Regel in den regulären Ausdruck [^B]* (A.*B)* [^B]*.

Reguläre Ausdrücke beschreiben eine eindeutige Menge von Zeichenketten. Für jeden regulären Ausdruck existiert ein EZA, welcher ausschließlich und vollständig alle Zeichenketten dieser Menge akzeptiert. Für eine Teilmenge aller möglichen Declare-Regelschablonen sind jeweils korrespondierende reguläre Ausdrücke so-

*Schritt 3* wie die zugehörigen EZAs vordefiniert. Somit besteht *Schritt 3* lediglich aus dem „Nachschlagen" des passenden EZA für den gegebenen regulären Ausdruck. Das Ergebnis sind demnach N Zustandsautomaten.

*Schritt 4* *Ein* EZA steht für lediglich *eine* Declare-Regel. Eine valide Prozessausführungsspur muss jedoch *alle* Declare-Regeln befolgen, weswegen in *Schritt 4* alle N EZAs mittels des Produktoperators miteinander verknüpft werden. Akzeptiert ein Automat eine gegebene Zeichenkette, dann ist das Ergebnis der Auswertung 1 und andernfalls 0. Somit ist bei einem einzelnen Regelverstoß das Gesamtergebnis ebenfalls 0, was der Semantik des Declare-Modells entspricht. Das Ergebnis dieses Schrittes ist somit ein kompositer Automat.

*Schritt 5* In *Schritt 5* wird der komposite Automat mehrfach auf einem zufällig gewählten Pfad durchlaufen, wobei jeder Durchlauf in einem definierten Endzustand terminiert. Jede Transition wird durch die Emission eines Zeichens des Automatenalphabets eingeleitet. In einem Durchlauf wird jedes emittierte Zeichen an eine Zeichenkette angehängt, die so die chronologische Reihenfolge der Emissionen protokolliert. Die Anzahl der Durchläufe T muss ebenso wie die maximale Länge $L_{max}$ parametriert werden. Letzteres ist ein künstliches aber notwendiges Abbruchkriterium. Das Ergebnis dieses Schrittes sind somit T Zeichenketten mit der maximalen Länge $L_{max}$. Entgegen der meisten imperativen Ansätze ist es nicht möglich, die Wahrscheinlichkeit bestimmter Ausführungspfade zu beeinflussen.

*Schritt 6* Im *sechsten* und letzten *Schritt* werden die Zeichen aller Zeichenketten durch die Namen der in Schritt 1 zugeordneten Aktivitäten ersetzt, ohne die Reihenfolge zu verändern. Jede Zeichenemission wird somit mit einem Ereignis einer Aktivitätsausführung gleichgesetzt. Dabei wird nicht zwischen Start- und Endereignissen unterschieden. Zusätzlich wird für jede Aktivitätsausführung ein zufälliger Zeitstempel berechnet. Das Ergebnis wird schließlich im XES-Format serialisiert.

Der beschriebene Ansatz ist trotz des scheinbar aufwendigen Transformationsprozesses in der Lage, für *kontrollflussbasierte*, deklarative Prozessmodelle umfangreiche, künstliche Ereignisprotokolle effizient zu erzeugen. Die Technik ist jedoch nicht auf *multi-perspektivische* deklarative Prozessmodellierungssprachen

*Hindernis: Eindimensionalität regulärer Ausdrücke* übertragbar. Der Grund dafür ist die *Eindimensionalität* regulärer Ausdrücke und zugehöriger Zeichenketten. Der Ansatz geht davon aus, dass jedes verwendete Zeichen eines Alphabets als Kodierung für den Namen einer Aktivität verwendet wird. Betrachtet man beispielsweise die Organisatorische oder die Operationale Perspektive (Abschnitt 1.1.2), dann muss der Spurgenerator aber auch in der Lage sein, Akteure und Software-Werkzeuge mit diesen Aktivitäten zu assoziieren. Da die Menge möglicher Akteure und die Menge zu verwendender Software-Werkzeuge diskret und endlich ist, ist es hier theoretisch möglich, dies auf den eindimensionalen Raum abzubilden. Das Zeichen A kodiert dann nicht mehr nur eine Aktivität, also beispielsweise Bewerben, sondern steht dann

für ein Tupel wie (Bewerben, MaxMustermann, FancyTextProgram). Das Zeichen B kann dieselbe Konstellation aber mit einem anderen Akteur kodieren (z.B. (Bewerben, MaraMusterfrau, FancyTextProgram)). Allgemein lässt sich die Anzahl $N_c$ der so notwendigen Zeichen über folgende Formel beschreiben:

$$N_c = \prod_{p \in P} |p| \tag{24}$$

Dabei bezeichnet P eine geschachtelte Menge, bestehend aus den Mengen der Elemente der unterschiedlichen Perspektiven. Aus der Anzahl möglicher Kombinationen jedes Elements einer Menge mit denen der anderen ergibt sich die Gesamtzahl notwendiger Kodierungszeichen. Diese entspricht folglich dem Produkt der Elementanzahl aller Perspektiven. Für die Kodierung eines Prozessmodells mit beispielsweise zehn Aktivitäten, drei möglichen Akteuren und zwei Software-Werkzeugen würden bereits sechzig Zeichen benötigt. Diese Komplexität allein verursacht bisher nur Schwierigkeiten in der praktischen Anwendung, ist jedoch in der Theorie noch kein Ausschlusskriterium für die vorgestellte Technik – die Daten-orientierte Perspektive hingegen schon. Grund ist, dass die „Menge" möglicher Datenwerte – also der Wertebereich – potentiell nicht diskret, sondern kontinuierlich ist. Damit ist die Anzahl der Elemente in der Menge der Datenwerte und folglich auch die Anzahl der benötigten Zeichen potentiell unendlich. Sollte die Festlegung auf ein beschränktes Intervall im Bereich der natürlichen Zahlen möglich sein, so sind die Anzahl möglicher Datenwerte und dafür notwendiger Kodierungszeichen endlich. Bezüglich der Leistungsfähigkeit endlicher Zustandsautomaten stellt dies dennoch ein bekanntes Problem dar, welches als *Zustandsexplosionsproblem* (engl. *state explosion problem*) bezeichnet wird. Jedes emittierte Zeichen führt zu einem neuen Zustand. Die dadurch potentiell große Anzahl an Zuständen senkt die Leistungsfähigkeit des Spurgenerators deutlich. Dies wird bereits im eindimensionalen Raum der Kontrollfluss-Regeln von Declare erkannt [147].

*Problem der Zustandsexplosion*

Abschließend ist somit festzustellen, dass sich das in [41] beschriebene Simulationsverfahren für kontrollflussbasierte, deklarative Prozessmodelle begrenzter Größe eignet. Für die Simulation *multi-perspektivischer*, deklarativer Prozessmodelle ist es jedoch ungeeignet. Das Detail der nachträglichen Generierung der Zeitstempel auf Basis einer Wahrscheinlichkeitsverteilung wird allerdings von dem in Abschnitt 4.2 beschriebenen Simulationsansatz übernommen.

#### 4.1.2.2 *Hybride Declare-CPN-Modelle*

Ein weiterer Ansatz für die Simulation von Declare-Modellen ist in [193] beschrieben. Der Ansatz greift dabei auf die Flexibilität des bereits in Abschnitt 4.1.1 beschriebenen CPN Tools zurück. Die aus den allgemeinen Petri-Netzen übernommenen Transitionen werden dazu mit Aktivitäten aus der Sprache Declare gleichgesetzt. Ein von CPN Tools angebotener Erweiterungsmechanismus bietet einerseits die Möglichkeit der Erweiterung der Funktionalität durch Dritte und andererseits die Kommunikation der jeweiligen Erweiterung mit dem CPN-Tools-Simulator und der grafischen Nutzerschnittstelle. Der Simulationsansatz aus [193] implementiert einen Filter für die Kommunikation zwischen Simulator und Nutzer-

*Verknüpfung von Declare und CPN Tools*

schnittstelle, welcher auf Basis der Auswertung eines Declare-Modells Transitionen entweder aktiviert oder deaktiviert. Die Auswertung selbst basiert auf dem bereits existierenden Ausführungsansatz für Declare [145]. Dieser bildet das jeweilige Declare-Modell zunächst auf eine Menge an LTL-Formeln ab, aus denen schließlich semantisch äquivalente endliche Zustandsautomaten abgeleitet werden. Die Simulationsausführung übernimmt hier CPN Tools.

*Aktivierung von Transitionen*

CPN Tools entscheidet beim Start der Simulation und nach jeder Transition, welche Transitionen jeweils aktiviert sind. Diese Prüfung erfolgt in zwei Schritten. Im ersten Schritt wird gemäß der CPN-Semantik anhand des jeweiligen Markierungsszustands des Modells geprüft, welche Transitionen aktiviert werden können [90]. Generell ist in CPN Tools automatisch jede aktiviert, die keinerlei eingehende Kanten aus entsprechenden Plätzen aufweisen. In einem Declare-Modell gilt das für alle Aktivitäten, da in selbigem keine Entsprechung für Petri-Netz-Plätze existiert. Damit sind aus Sicht des CPN *alle* Transitionen aktiviert, sodass ausschließlich Schritt zwei für eine Deaktivierung sorgen kann. Dieser beinhaltet eine Prüfung der Declare-Regeln jeder Transition, welche in EZA-Form vorliegen.

*Keine konfigurierbaren Entscheidungen*

Die Simulation kann hier nicht wie in [124] mittels Wahrscheinlichkeitsverteilungen bestimmte Teilpfade des Prozesses priorisieren. Grund ist das Fehlen von expliziten Entscheidungspunkten. Diese ergeben sich lediglich implizit durch die Kombination aller Declare-Regeln. In [193] wird keine Aussage darüber getroffen, wie CPN Tools in diesem Fall zu einer Entscheidung gelangt, welche aktive Transition jeweils ausgeführt wird. Ein Export der Simulationsergebnisse in einem der beiden üblichen XML-basierten Formate MXML respektive XES ist zwar standardmäßig nicht möglich, kann aber durch den modularen Aufbau von CPN-Tools jederzeit integriert werden. Beispielsweise kann hierfür der in [124] entwickelte Serialisierungsmechanismus wiederverwendet werden.

*Verstärkte Auswirkungen des Problems der Zustandsexplosion*

Wie die in [41] vorgestellte Technik greift auch der aktuell diskutierte Ansatz auf endliche Zustandsautomaten für die Simulationsausführung zurück. Im Gegensatz zu ersterem wird allerdings beim Simulationsstart und nach jeder Transition für alle Transitionen geprüft, ob sie aktiviert werden können. Das bedeutet eine Verstärkung der Auswirkungen des Problems der Zustandsexplosion, welches aus der enormen Größe der aus realistischen Declare-Modellen generierten EZAs resultiert. Für die Simulation *multi-perspektivischer* Prozessmodelle ist der CPN-Tools-basierte Ansatz dennoch ein Fortschritt. Der Grund dafür ist, dass hier Declare als Erweiterung der weiterhin verfügbaren reinen CPN-Modellierungssprache integriert ist. Das ermöglicht beispielsweise die Berücksichtigung der Datenorientierten Perspektive bei der Simulation eines Declare-Modells. Da diese aber nicht von der rein kontrollflussorientierten Sprache Declare selbst unterstützt wird, kann von der Simulation multi-perspektivischer, *deklarativer* Prozessmodelle nicht gesprochen werden. Elemente anderer Perspektiven (Abschnitt 1.1.2) können dann möglicherweise auf Bestandteile der CPN-Notation abgebildet werden. Bislang ist jedoch kein Verfahren bekannt, welches *beliebige* deklarative Prozessmodelle auf eine äquivalente Repräsentation transformiert. Die wenigen verfügbaren Techniken dieser Art reduzieren Declare auf eine kleine Menge von Regelschablonen (z.B. [147]). Folglich ist der hier vorgestellte CPN-Tools-basierte Ansatz für die Spurgenerierung für multi-perspektivische, deklarative Prozessmodelle ungeeignet.

#### 4.1.2.3 *DCR-Graphen*

Im Gegensatz zu Declare unterstützen DCR-Graphen auch die Datenorientierte Perspektive [169]. Die Organisatorische Perspektive wird lediglich durch ein einfaches Rollenkonzept zur Gruppierung von Ressourcen und darauf basierenden Einschränkungen der möglichen Ausführenden einer Aktivität unterstützt. In [122] wird ein Ansatz zur kollaborativen Simulation von DCR-Graphen beschrieben. Mit Simulation ist hier allerdings in erster Linie eine interaktive, rollenspielbasierte Ausführung gemeint, welche wörtlich auf das „Durchspielen" des Prozesses zur Erforschung der möglichen Verläufe orientiert ist. Der Simulator bietet allerdings zwei *automatisierte Nutzer* an, die in unterschiedlichem Umfang den Prozess vollautomatisiert abarbeiten können. Dafür müssen die im Modell vorkommenden Rollen einem dieser beiden zugeordnet werden. Der eine automatisierte Nutzer bearbeitet dabei lediglich Aufgaben, die vom Prozessmodell *gefordert* werden, während der zweite alle verfügbaren und nicht bereits ausgeführten Aktivitäten durchführt. Die Simulationsdurchführung erfolgt dann entsprechend der Ausführungssemantik der DCR-Graphen. Bei mehreren zur Bearbeitung verfügbaren Aufgaben wird zufällig entschieden, in welcher Reihenfolge diese durchgeführt werden. In [122] wird keine Aussage darüber getroffen, welchem Modell diese Zufallsentscheidung folgt.

*Interaktive Simulation*

*Verschiedene Nutzermodelle*

Der Simulator protokolliert pro Ereignis den Aktivitätsnamen, einen zugehörigen Zeitstempel des Beginns der Bearbeitung derselben sowie die ausführende Ressource. Ein Export der produzierten künstlichen Ausführungsspuren in ein übliches Protokollformat – wie beispielsweise XES oder MXML – existiert nicht.

*Kein Export in standardisiertes Format*

DCR-Graphen sind, im Gegensatz zu Declare, multi-perspektivisch, da sie einerseits die Datenorientierte Perspektive und in Ansätzen auch Organisatorische Perspektive unterstützen. Die Reduzierung letzterer auf ein starres Ressourcen-Rollen-Konzept bedeutet, dass DCR-Graphen gegenüber anderen deklarativen Sprachen – wie beispielsweise DPIL – keine *dynamischen* Regeln unterstützen kann, in welcher die organisatorische Perspektive eine Rolle spielt. Beispielsweise sind die beiden Konzepte der *separation of duties* respektive *binding of duties* (Abschnitt 3.1.5) mit DCR-Graphen nicht umzusetzen. Weiterhin ist die Spezifikation des Simulationsansatzes nicht aussagekräftig genug, um diese Technik zu adaptieren. Das Konzept der Simulation unterschiedlicher Nutzertypen, welche entsprechend ihrer Charakteristika im Prozess unterschiedlich agieren, wird in Abschnitt 4.2 jedoch aufgegriffen und erweitert.

*Keine Unterstützung dynamischer Regeln*

#### 4.1.2.4 *Hybride Modelle mit Declare-, CPN- und DCR-Graph-Anteilen*

Der vierte und letzte Ansatz zur Spurgenerierung für deklarative Prozessmodelle ist in [194] beschrieben und basiert erneut auf CPN Tools. In der nachfolgend beschriebenen Technik ist eine kombinierte Nutzung von CPNs, Declare-Modellen und DCR-Graphen möglich. Die Transitionen der CPNs werden dazu mit den Regeln aus Declare und den Ereignissen aus der DCR-Graph-Notation gleichgesetzt. Somit ist die Aktivierung einer Transition nicht nur vom Markierungszustand des CPN, sondern auch von den Ergebnissen der Auswertungen aller Regeln der Declare- und DCR-Graph-Anteile abhängig. Um eine effiziente Simulation zu gewährleisten, werden die Modellanteile *aller* drei Sprachen auf sogenannte

*Kombination von CPN-Transitionen, Declare-Regeln und DCR-Graph-Ereignissen*

*Präfixautomaten* abgebildet. Dieser ist nichts anderes als ein endlicher Zustandsautomat, welcher alle Präfixe einer bestimmten Zeichenkette akzeptiert. Da die Ausführungssemantik von sowohl Declare als auch DCR-Graphen mittels EZAs beschrieben werden kann, ist garantiert, dass ein solcher Präfixautomat in jedem Fall existiert. Jeder dieser Automaten erlaubt die Aktivierung einer Transition nur dann, wenn sie zu einem Endzustand in selbigem führt. Für jeden einzelnen Automaten ist dies auch entscheidbar, nicht aber für ihr Produkt, welches gebildet werden muss, um sicherzustellen, dass eine Sequenz von Ereignissen die Regeln und Vorgaben *aller drei* Modellanteile befolgt. Zudem existiert für jedes CPN, welches Schleifen oder Datenobjekte mit kontinuierlichem, unbegrenztem Wertebereich enthält, kein Präfixautomat. Die Anzahl der Zustände wäre unendlich groß. Eine Garantie für die Validität der Aktivierung einer Transition kann folglich im Allgemeinen nicht gegeben werden. Aus diesem Grund wird bei der Simulation des kombinierten Modells nicht nur zwischen den Zuständen *valide* und *invalide*, sondern auch zwischen den zwei schwächeren Zuständen *temporär valide* respektive *temporär invalide* unterschieden.

*Simple Simulation*

Die Simulationsdurchführung kann auf drei verschiedene Arten erfolgen. Die sogenannte *Simple Simulation* transformiert ähnlich wie in [41] jede einzelne Declare-Regel in einen EZA. Damit ist garantiert, dass sich die Simulation nie in einem *permanent* invaliden Zustand befindet. Für den DCR-Graph-Anteil wird in diesem Modus keine Information gespeichert, welche Regeln derzeit invalide oder valide sind. Stattdessen wird mittels des aktuellen Markierungsszustands vor der Ausführung einer Transition geprüft, ob diese auch gemäß der DCR-Graph-Anteile aktiviert ist. In diesem ersten Modus können zwar keine Declare-Regeln verletzt, dafür aber temporär invalide sein. Deswegen ist die Terminierung in einem gültigen Endzustand nicht garantiert. Im zweiten Modus, der sogenannten *Smart Simulation*, wird das explizit ausgeschlossen. Dazu wird für das vollständige Declare-Modell ein Präfixautomat abgeleitet. Zusätzlich führt dieser Modus nur Transitionen aus, welche keine der Regeln permanent oder temporär verletzt. Damit wird seitens des Declare-Anteils zu jedem Zeitpunkt die Terminierung in einem gültigen Endzustand garantiert. Mit den DCR-Graph-Anteilen wird analog verfahren, was insgesamt die Entscheidbarkeit garantiert. Hierbei ist allerdings kritisch anzumerken, dass mit diesem Verfahren beispielsweise die Declare-Regel *response(A,B)* eine Ausführung von Aktivität *A* vollkommen unmöglich macht. Grund hierfür ist, dass die Ausführung von A die Regel temporär verletzt, da erst ab diesem Zeitpunkt eine Ausführung von B gefordert wird. Ein invalider Zustand kann nicht nur durch die Missachtung der kontrollflussbasierten Regeln der deklarativen Modellanteile hervorgerufen werden. Vielmehr kann seitens der deklarativen Anteile die Ausführung einer Transition gestattet sein, während eine nicht erfüllte Datenabhängigkeit im CPN-Anteil diese verbietet. Aus diesem Grund ist der dritte und letzte Modus, die sogenannte *Datenbewusste Simulation* problematisch. Für abgeschlossene, kleine Beispiele kann dieser Modus zunächst den vollständigen Zustandsraum erzeugen und typischerweise in Minuten oder Stunden die Aktivierung oder Deaktivierung einer Transition veranlassen. Für größere Model-

*Smart Simulation*

*Datenbewusste Simulation*

le[4] werden mittels Smart Simulation zunächst viele Durchläufe simuliert. In einem Nachbearbeitungsschritt werden dann die Durchläufe verworfen, die nicht in einem validen Endzustand terminieren.

Wie in [193] ist auch hier kein direkter Export der erzeugten Prozessausführungsspuren in ein Standard-Protokollformat möglich. Dafür kann aber der in [124] beschriebene Serialiserungsmechanismus wiederverwendet werden.

Spurgeneratoren werden im Kontext der vorliegenden Arbeit einerseits dazu verwendet, Beispiele für Prozessausführungen zu liefern, welche das Verständnis *eines* Modells verbessern sollen. Andererseits werden sie zur Generierung künstlicher Ereignisprotokolle für die induktive Translation von Prozessmodellen *einer* Sprache verwendet. Die Erweiterung der Ausdrucksmächtigkeit einer Sprache durch die Komposition mit anderen Sprachen ist daher nicht im Fokus der Arbeit. Folglich bietet der aktuell diskutierte Ansatz keine Vorteile für die Simulation der Modelle der *einzelnen* Sprachen. Besonders die Problematik, dass nicht in jedem Zustand der Simulation eine valide Terminierung garantiert werden kann, ist als großer Nachteil anzurechnen. Der in Abschnitt 4.2 beschriebene Ansatz basiert auf einer Sprache größerer Ausdrucksmächtigkeit, vermeidet aber diesen Nachteil.

*Hybride Ansätze nicht im Fokus*

Ziel von Abschnitt 4.1 ist es, einen Überblick über existierende Spurgeneratoren zu gewinnen. Dies dient einerseits dazu, eine Vergleichsbasis für den in dieser Arbeit entwickelten Spurgenerator (Abschnitt 4.2) zu schaffen und andererseits dazu, Simulationskomponenten für den ebenfalls in dieser Arbeit vorgestellten Translationsansatz (Abschnitt 5.2) zu identifizieren. Dabei zeigt sich, dass im Bereich der imperativen Prozessmodellierungssprachen – besonders für Petri-Netze – eine größere Anzahl an Spurgeneratoren zur Verfügung steht. Die einzigen zwei für deklarative Prozessmodelle eignen sich entweder nicht für *multi-perspektivische* Prozessmodelle [41] oder decken nur einen geringen Teil der geforderten Funktionen und Eigenschaften (Tabelle 4.1) ab [122]. Somit ist die Entwicklung einer Technik mit größerem Funktionsumfang zur Spurgenerierung für multi-perspektivische, deklarative Prozessmodelle naheliegend und sinnvoll (Abschnitt 4.2).

## 4.2 LOGIKBASIERTE GENERIERUNG VON PROZESSAUSFÜHRUNGSSPUREN FÜR MULTI-PERSPEKTIVISCHE, DEKLARATIVE PROZESSMODELLE

In diesem Abschnitt wird eine Technik zur Erzeugung künstlicher Ereignisprotokolle für multi-perspektivische, deklarative Prozessmodelle am Beispiel von DPIL vorgestellt. Diese stellt einen der Kernbeiträge der vorliegenden Arbeit dar (Abschnitt 2.1). Das generelle Vorgehen des Ansatzes (Abbildung 4.3) umfasst die Translation des zu simulierenden Prozessmodells von DPIL auf Alloy als Sprache zur Beschreibung des Simulationsmodells mittels einer Modell-zu-Text-Transformation (*Modell-zu-Text*). Das Simulationsmodell besteht aus einem statischen Teil, der in der Abbildung als *Metamodell für Ereignisketten* bezeichnet ist. Dieser beschreibt im Wesentlichen die in Abschnitt 3.3.1 formal definierten Prozessausführungsspuren mit Alloy-Sprachmitteln. Der auf dem statischen basierende dynamische Teil ist die Alloy-Repräsentation des zu simulierenden

*Logikbasierte Simulation*

4 Die Größe wird hier weniger durch die Anzahl der sichtbaren Modellelemente, sondern mehr durch die Größe des Wertebereichs von Datenobjekten beeinflusst.

Prozessmodells. Nach der Transformation erfolgt die Suche nach Instanzen, die das gegebene Modell entweder erfüllen oder dagegen verstoßen – je nachdem, ob Positiv- oder Negativbeispiele bezüglich des Modells angefordert werden. In einem *Nachbearbeitung*sschritt werden die erzeugten Prozessausführungsspuren beispielsweise mit Zeitstempeln versehen und ins Ereignisprotokollformat XES transformiert, was auch das Ausgabeformat des Ansatzes ist.

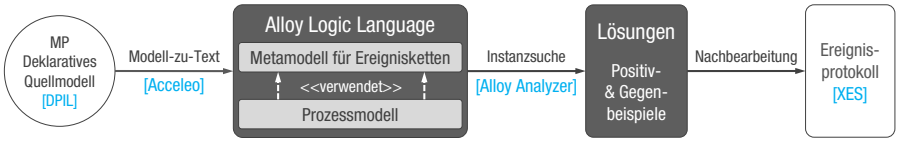



Abbildung 4.3: Prinzip der logikbasierten Generierung von Prozessausführungsspuren für multi-perspektivische, deklarative Prozessmodelle

In Abschnitt 4.2.1 wird die zur Definition des Simulationsmodells verwendete Sprache Alloy detailliert betrachtet. Darauf folgt ein Abschnitt (Abschnitt 4.2.2), der am Beispiel von Alloy erläutert, wie die Transformation eines deklarativen Prozessmodelle in ein semantisch äquivalentes Erfüllbarkeitsproblem möglich ist und inwiefern dies zur Spurgenerierung verwendet werden kann. Die Sektionen 4.2.3 bis 4.2.8 erläutern im Detail die in Abbildung 4.3 dargestellten Schritte, Zwischen- und Endresultate sowie verschiedene Konfigurationsmöglichkeiten.

### 4.2.1 *Alloy*

*Alloy*

In diesem Abschnitt wird die Modellierungssprache *Alloy* für die Erzeugung von auf DPIL-Modellen basierenden Simulationsmodellen überblicksweise eingeführt. Alloy wird also verwendet, um die Semantik von DPIL-Modellen zur Spurgenerierung verwenden zu können. In den einzigen vergleichbaren Ansätzen zur Spurgenerierung wird die Semantik der deklarativen Prozessmodelle entweder mittels regulärer Ausdrücke oder in Form spezieller CPN-Modelle repräsentiert. Beide Repräsentationen eignen sich jedoch nicht für die Simulation auf Basis *multi-perspektivischer* deklarativer Prozessmodelle oder werden an sich als problematisch angesehen (Abschnitt 4.1.2). Die Wahl von Alloy wird wie folgt begründet:

*Begründung der Wahl*

- Alloy basiert auf der Prädikatenlogik erster Stufe, ist also ausdrucksmächtiger als die in einem verwandten Ansatz verwendeten regulären Ausdrücke. Dies erlaubt die Berücksichtigung beliebig vieler Prozessperspektiven.
- Auch DPIL basiert auf der Prädikatenlogik erster Stufe, was sich im Sprachdesign widerspiegelt und so eine Abbildung auf Alloy erleichtert.
- Alloy bezeichnet sowohl eine Sprache als auch ein Analysewerkzeug, das als Ausführungssystem zur Spurgenerierung verwendet werden kann.
- Alloy bietet die Möglichkeit, sowohl Positiv- als auch Gegenbeispiele für respektive gegen valide Prozessausführungen zu erzeugen (Anforderung A2).
- Alloy ist flexibel und erleichtert die Umsetzung vieler der in Tabelle 4.1 zusammengefassten, nützlichen Funktionen vergleichbarer Ansätze.

Die nachfolgenden Abschnitte beschreiben die sprachlichen Grundkonzepte und Charakteristika von Alloy und begründen darauf aufbauend die Wahl dieser Sprache zur Beschreibung von Simulationsmodellen.

#### 4.2.1.1 *Grundidee und Funktionsweise*

Alloy ist eine deklarative Modellierungssprache zur Beschreibung von Software-Strukturen unter Berücksichtigung gewünschter oder geforderter Restriktionen. Die Motivation für die Entwicklung einer solchen Sprache beginnt in [88, S. 1] mit den folgenden Worten:

> "Software is built on abstractions. Pick the right ones, and programming will flow naturally from design; [...]"

*Alloy: Sprache zur Softwareabstraktion*

Abstraktionen trennen das Design von der programmatischen Umsetzung. Dies bietet den Vorteil, einen problemorientierten Entwurf anfertigen zu können, ohne an die Ausdrucksmittel der Lösungssprache gebunden zu sein, welche ein Design häufig durch problemferne „technische Details" unleserlich machen. Eine eindeutige und präzise Möglichkeit der Abstraktion mit garantierter technischer Realisierbarkeit sind die sogenannten *formalen Spezifikationen*. Formale Spezifikation wird jedoch selten als Designmethode gewählt, da viele existierende Notationen zwar nicht durch technische Details verwirren, jedoch durch ihre mathematische Syntax ähnliche Schwierigkeiten verursachen. Eine ausreichende Werkzeugunterstützung würde dieses Problem abschwächen. Es mangelt jedoch an Werkzeugen, die Unterstützung über einfache Typprüfung und Syntaxhervorhebungen hinaus anbieten. Besonders Theorembeweiser wären zu Verifikationszwecken hilfreich.

*Formale Spezifikation*

Mit Alloy wird eine kompakte und robuste Sprache beschrieben, welche einerseits die Idee der formalen Spezifikation aufgreift. Andererseits wird in der zugehörigen Werkzeugunterstützung auf Theorembeweiser *verzichtet*. Stattdessen setzt man auf eine vollautomatische Analyse, sogenanntes *Constraint Solving*, das zwar nicht den kompletten Lösungsraum analysiert, dafür aber schneller Rückmeldung für einen begrenzten Lösungsraum anbietet. Allerdings sind die Anzahl der möglichen zu analysierenden Lösungen üblicherweise enorm. Der Hauptvorteil jedoch ist, dass keine zusätzliche Testspezifikation notwendig ist. Stattdessen formuliert man die zu prüfende Eigenschaft. Der Constraint Solver überprüft die gewählte Eigenschaft für die gegebene formale Spezifikation.

*Constraint Solving*

*Zusammenspiel von Logik, Sprache und Analysewerkzeug*

Alloy ist genau genommen eher ein Rahmenwerk[5] und *beinhaltet* eine Logik, eine Sprache und ein Analysewerkzeug. Die Logik liefert die Bausteine für die Softwarestrukturen, die vollständig mittels Relationen und wenigen, mächtigen Operatoren beschrieben werden können. Zustände und Ausführungen werden mittels *Constraints* – als Formeln im Logikkontext und Boolesche Ausdrücke im Programmierumfeld bezeichnet – beschrieben. Die Sprache bietet ein paar wenige, syntaktische Erweiterungen zur Strukturierung wie beispielsweise Module. Auch ein Typsystem ist enthalten. Der Analyseteil beinhaltet die bereits erwähnten Constraint Solver. Diese können Beispielinstanzen generieren, welche eine gegebene Eigenschaft *erfüllen* (Simulation) und andererseits können Gegenbeispiele

*Positiv- und Gegenbeispiele*

5 Nachfolgend wird der Einfachheit halber jedoch häufig zu Alloy als Sprache Bezug genommen.

ermittelt werden, welche die Eigenschaft *verletzen* (Checking). In beiden Fällen werden die Instanzen aus einem begrenzten Lösungsraum ermittelt, haben also einen anzugebenden maximalen Umfang.

#### 4.2.1.2 *Logisches Fundament*

*Alloy-Logik*

Der *Logikteil* von Alloy basiert kombiniert drei Logiken, die *Prädikatenlogik* erster Stufe, die *relationale Logik* und eine Logik für *Navigationsausdrücke*. Das wesentliche Fundament ist dabei die relationale Logik, welche mit den Quantoren der Prädikatenlogik angereichert werden. Navigationsausdrücke bilden Mengen durch die Traversierung von Relationen zwischen quantifizierten Variablen. In vielen Fällen lässt sich somit ein Constraint in jedem der drei Formalismen formulieren. Zur Illustration im Prozesskontext nehmen wir an, dass es eine Relation `performer` gibt, welche eine Abbildung von einer Aktivitätsausführung (`Event`) auf den oder die Ausführenden (`Performer`) bezeichnet. Angenommen, man möchte die Anzahl der Ausführenden auf $\leqslant$ `1` begrenzen, dann bestehen die in Codeausschnitt 4.1 dargestellten drei Möglichkeiten.

*Drei Logiken*

```
∀ e: Event, p, p': Performer | e->p in performer and e->p' in performer → p=p'

∀ e: Event | lone e.performer

no ~performer.performer - iden
```

Codeausschnitt 4.1: Ein Constraint, drei logische Beschreibungsmöglichkeiten

Die Frage, welche Notation die „beste" ist, lässt sich pauschal nicht beantworten. Für das genannte Beispiel lautet die Antwort: keine. Alloy bietet hierfür die Kurzschreibweise `performer in Event -> lone Performer` an. Auch aufgrund dieser großen Flexibilität hat folgende Aussage ihre Berechtigung [88, S. 1]

> "Since the same idea can be reduced to different forms, abstractions are always, in a sense, inventions, even if the ideas they reduce existed before [...]"

*Atome und Relationen*

Entitäten und Beziehungen werden in Alloy mittels *Atome* und *Relationen* beschrieben. Ein Atom repräsentiert eine primitive Entität und ist *unteilbar*, *unveränderlich* sowie *uninterpretiert*. Letzteres bedeutet, dass Atome – im Gegensatz zu z.B. Zahlen – keine Eigenschaften besitzen. In der Realität sind die wenigsten Dinge tatsächlich atomar. *Relationen* ermöglichen daher die Beschreibung kompositer, veränderlicher und interpretierter Strukturen. Relationen setzen Atome in Beziehung zueinander. Sie bestehen aus einem Satz von Tupeln, wobei jedes eine Sequenz von Atomen ist. Eine Relation ist daher mit einer Tabelle vergleichbar, in der jeder Eintrag ein Atom ist. Die *Größe* einer Relation ist definiert als die Anzahl der Zeilen. Die Anzahl der Spalten dagegen bezeichnet die sogenannte *Arität*. Die in Codeausschnitt 4.1 eingeschränkte Relation `performer` hat die Arität 2 und wird als *binäre* Relation bezeichnet. Relationen mit den Aritäten 1 respektive 3 werden als *unäre* beziehungsweise *ternäre* Relationen bezeichnet. Unäre Relationen beschreiben im Wesentlichen Skalare. Relationen ohne zugehörige Tupel werden als *leer* und Relationen mit höchstens einem Tupel werden als *Option* bezeichnet.

*Größe und Arität einer Relation*

In der Arithmetik kann man Konstanten mittels Operatoren verknüpfen. Alloy verfügt über drei eigene Konstanten und einen kompakten Satz ausdrucksstarker Operatoren. Die drei Konstanten sind die leere Menge (`none`), die Universalmenge (`univ`) und die Identität (`iden`). Sowohl `univ` als auch `iden` sind nur für genau *ein* Modell konstant. Bezüglich eines Modells, bestehend aus zwei disjunkten Mengen `Event` = `{(Push1),(Push2),(Pull)}` und `Performer` = `{(Max),(Mary)}`, sind die Konstanten wie folgt definiert:

*Drei Konstanten*

- `univ` = `{(Push1),(Push2),(Pull),(Max),(Mary)}`
- `iden` = `{(Push1,Push1),(Push2,Push2),(Pull,Pull),(Max,Max),(Mary,Mary)}`

Die *Operatoren* werden in zwei Klassen unterteilt: *Mengenoperatoren*, für die die Struktur der Tupel irrelevant ist und *relationale Operatoren*, für welche die Tupelstruktur essentiell ist. Die Mengenoperatoren beinhalten typische Operatoren, wie beispielsweise Vereinigung (+), Schnittmenge (&) und Differenz (−). Relationale Operatoren sind beispielsweise Produkt (→), Verbund (.) oder auch die transitive Hülle (^).

*Zwei Klassen von Operatoren*

`CompleteEvent` und `Performer` sind die *Namen* zweier Relationen, die mittels einer *Deklaration* eingeführt werden. Eine Deklaration folgt immer der Schreibweise ⟨Relationsname⟩ : ⟨Ausdruck⟩ und somit ist `performer: Event -> Performer` die Deklaration der `performer`-Relation. Deklarationen können zudem *Multiplizitätsbeschränkungen* beinhalten. In Alloy sind dafür vier Multiplizitäten definiert: Beliebig (`set`), genau Eins (`one`), höchstens Eins (`lone`) und mindestens Eins (`some`). Somit schränkt `lone in performer in TaskEvent -> lone Performer` die Anzahl der Ausführenden pro `Event` auf höchstens Eins. Auch grundlegende arithmetische Operationen auf Basis ganzer Zahlen ist in Alloy möglich. Damit könnte man die Einschränkung auf höchstens einen Ausführenden auch mit dem Ausdruck ∀ `e TaskEvent | #e.performer` ⩾ `1` festlegen oder dies dynamisch berechnen.

*Multiplizität*

#### 4.2.1.3 *Notation*

Der *Sprachteil* von Alloy beinhaltet eine Syntax, um die oben beschriebenen Logikausdrücke kompakter und lesbarer zu beschreiben. Dazu zählen Kurzschreibweisen für beispielsweise Deklarationen aber auch ein Modularisierungskonzept. Die Modellstruktur besteht aus drei Blöcken: Der *Modulkopf* dient der Benennung des Moduls (`module`) und dem Import anderer Module (`import`). Der *Spezifikationsteil* umfasst die Strukturbeschreibung selbst, bestehend aus Deklarationen, Constraints und Behauptungen. Im *Kommandoteil* erfolgt – darauf aufbauend – die Definition des Scopes und des gewünschten Analysemodus. Für die Strukturbeschreibung des Alloy-Modells stehen die folgenden Sprachelemente zur Verfügung:

*Alloy-Notation*

- *Signaturen* (`sig`) beschreiben jeweils eine Menge von Atomen und ähneln einer Klasse in objektorientierten Programmiersprachen (OOP). So kann sie beispielsweise Felder beinhalten, selbst abstrakt sein und Eigenschaften polymorph an andere Signaturen vererben. Zusätzlich hat jede Signatur eine Multiplizität.

*Signaturen ≈ Klassen*

*Fakten ≈ Invarianten*

- *Fakten* (`fact`) sind mit Invarianten aus der *Object Constraint Language (OCL)* [188] vergleichbar und ermöglichen die Spezifikation von Constraints, von denen angenommen wird, dass sie immer wahr sind.

*Funktionen = Funktionen*

- *Funktionen* (`fun`) sind, wie auch in OOP, wiederverwendbare Code-Teile, die parametriert werden können, einen Rückgabetyp aufweisen und auf Basis der Parameterwerte Berechnungen durchführen.

*Prädikate = Boolesche Funktionen*

- *Prädikate* (`pred`) sind spezielle Funktionen mit der Einschränkung, dass ihr Rückgabewert immer ein Wahrheitswert – also `true` oder `false` – ist. Der zweite große Unterschied ist, dass Alloy Prädikate „ausführen" kann und daraufhin Instanzen des Alloy-Modells ermittelt, die das Prädikat erfüllen – genannt Positivbeispiele.

*Behauptungen ≈ Testfälle*

- *Behauptungen* (`assert`) beinhalten Behauptungen. Der Körper einer Behauptung kann derselbe sein, wie bei einem Fakt. Im Gegensatz zu Fakten dienen Behauptungen jedoch dazu, die Eigenschaften zu formulieren, die getestet werden sollen. Dazu ermittelt Alloy Instanzen, welche die Behauptung verletzen – genannt Gegenbeispiele.

*Run und Check Scope*

Im Kommandoteil der Spezifikation kommt eines der beiden Analyse-Kommandos `run` oder `check` zum Einsatz. Zu beiden gehört eine entsprechende *Scope*-Definition zur Einschränkung der maximalen Größe der Lösungen. Der Scope definiert einen multidimensionalen Raum aus Testfällen, wobei jede Dimension die Begrenzung genau einer Signatur darstellt. Ein Beispiel für ein gültiges run-Kommando ist: `run myPredicate for 3 Event, 2 Performer`. So produziert Alloy für das Prädikat (myPredicate) und unter Beachtung der vollständigen, gegebenen Alloy-Spezifikation gültige Beispielinstanzen mit maximal 3 `Event`- und 2 `Performer`-Atomen. Die Definition eines check-Kommandos erfolgt analog, wobei anstelle des Prädikatnamens der Name der zu prüfenden Behauptung angegeben wird. Das Ergebnis sind entsprechend Gegenbeispiele gleicher Größe.

*Typsystem*

Der Sprachanteil von Alloy beinhaltet zusätzlich ein *Typsystem*. Dieses fängt Typfehler ab, bevor die eigentliche – meist teure – Analyse durchgeführt wird. Andererseits wird es auch eingesetzt, um *überladene* Felder aufzulösen. Haben verschiedene Signaturen Felder mit demselben Namen, dann wird über den Typ des zugehörigen Ausdrucks ermittelt, welches Feld jeweils gemeint ist. Typen werden in Alloy implizit mit Signaturen assoziiert. Jede Top-Level-Signatur und jede erbende Signatur hat einen *Basistyp*. Bei einer Vererbungshierarchie, bei der Signatur $A1$ die Signatur $A$ erweitert, ist der mit $A1$ assoziierte Typ ein Subtyp des mit $A$ assoziierten Typs. Ausdrücke haben dagegen einen *relationalen Typ*, der dem Typ der enthaltenen Tupel entspricht und mittels der Vereinigung der Produkte ermittelt wird. Folglich hat jeder Produktterm exakt so viele Basistypen, dass die Anzahl der Arität der Relation entspricht. Binäre Relationen haben also Produkte mit zwei Basistypen.

#### 4.2.1.4 *Ausführung durch Analyse*

*„Ausführung"*

Das Analysewerkzeug des Alloy-Rahmenwerks kommt dann zum Einsatz, wenn ein check- oder ein run-Kommando „ausgeführt" wird. Was Ausführung im Alloy-

Kontext bedeutet, wird im verbliebenen Teil dieses Abschnitts erläutert.

Im Wesentlichen bedeutet Ausführung bei beiden Kommandos das Zuweisen von Relationen zu Variablen zur Erfüllung der Constraints. Aus diesem Grund wird das Erfüllen eines Prädikats oder einer Behauptung im verbleibenden Teil des Abschnitts unter dem Begriff „Lösung" zusammengefasst. Alloys relationale Logik ist allgemein *unentscheidbar*. Das bedeutet, dass es nicht möglich ist, ein automatisches Werkzeug anzubieten, welches für *jede* mögliche Belegung eine Aussage treffen kann, ob selbige eine Lösung darstellt. Grund ist, dass die Anzahl möglicher Belegungen potentiell unendlich ist. Folglich führt nur eine Kompromissbildung zum Ziel.

*Unentscheidbarkeit*

Der traditionelle Kompromiss aus dem Bereich der Theorembeweise ist, dass ein automatischer Theorembeweiser versucht, die Existenz einer Lösung zu beweisen. Ist dieser Versuch erfolgreich, dann existiert trivialerweise auch eine Lösung. Schlägt der Versuch fehl, kann das einerseits bedeuten, dass keine Lösung existiert. Andererseits kann dennoch eine existieren, was jedoch durch eine falsche Behauptung, Limitationen des Beweisers selbst oder fehlerhafte Verwendung durch den Nutzer nicht bewiesen werden konnte. Zwischen diesen verschiedenen Gründen zu unterscheiden, ist potentiell sehr schwierig. Aus diesem Grund wird für Alloy ein anderer Weg gewählt. Anstatt Beweise zu konstruieren, versucht Alloy *Beispiele* und *Gegenbeispiele* einer begrenzten Größe zu finden. Dazu wird eine große Menge von *Testbelegungen* erzeugt und mit dem Prädikat oder der Behauptung abgeglichen. Der Unterschied zwischen den beiden Modi ist die Aussage des Resultats. Liefert Alloy für die Ausführung eines Prädikats eine Lösung, ist die Spezifikation erfüllbar. Für Behauptungen wird dagegen nach einem Gegenbeispiel gesucht. Wird keines gefunden, bedeutet das nicht, dass die Behauptung in jedem Fall korrekt ist. Im umgekehrten Fall gilt bei Nichtauffinden eines Beispiels für ein Prädikat, dass dieses dennoch erfüllbar sein könnte. Wird ein Gegenbeispiel für eine Behauptung gefunden, dann ist die der Behauptung zugrundeliegenden Annahme über eine Eigenschaft falsch. Folglich ist die Möglichkeit, eine hilfreiche Aussage zu treffen, von der gewählten Maximalgröße – dem bereits genannten Scope – einer Lösung abhängig. Diese gegensätzliche Aussagekraft der Alloy-Resultate ist zwar ein Kompromiss, allerdings kann für den gewählten Scope eine Garantie für die Richtigkeit einer Aussage in Form von Beispielen für ein Prädikat oder Gegenbeispielen gegen eine Behauptung gegeben werden.

*Instanzsuche statt Theorembeweis*

#### 4.2.1.5 *Warum Alloy?*

Alloy ähnelt in gewissen Punkten anderen Ansätzen wie *B*, *OCL*, *VDM* und *Z* und unterscheidet sich von diesen gleichzeitig signifikant [88, S. 301 ff.]. Alle genannten Ansätze, inklusive Alloy, bieten eine Notation zur Abstraktion von Software. Alle basieren auf den gleichen mathematischen Strukturen wie beispielsweise Mengen und Relationen. Außerdem sind sie alle deklarativ und Modelle werden folglich mittels Constraints spezifiziert. Ein signifikanter Unterschied ist, dass vor allem B und VDM aber auch Z und OCL auf endlichen Zustandsautomaten orientiert sind. In Abschnitt 4.1 wird diskutiert, dass endliche Zustandsautomaten für deklarative Prozessmodelle potentiell das Problem der Zustandsexplosion aufweisen und damit nicht leistungsfähig genug sind. Zudem fokussiert sich Alloy stärker auf eine

*Warum Alloy?*

automatische Analyse, was für die Anwendung zur automatischen Spurgenerierung wichtig ist. Alle anderen Ansätze benötigen eine Testspezifikation, also manuell festgelegte Eingaben, Bedingungen und erwartete Ausgaben. Alloy erschließt dagegen einen begrenzten Zustandsraum *vollständig* und *eigenständig*. Auch Alloys klare Trennung zwischen Spezifikation und Kommandos ist nicht in jedem Ansatz gegeben. Zudem ist Alloy kompakter und gleichzeitig flexibler als andere Ansätze, was besonders durch die Auffassung, dass alles eine Relation ist, ermöglicht wird. Ein weiterer wichtiger Grund für die Wahl von Alloy als „Simulationssprache" ist, dass sie im Gegensatz zu den anderen Sprachen strikt auf der Prädikatenlogik erster Stufe beruht – demselben Fundament, auf dem auch DPIL basiert. Das erleichtert eine Transformation von DPIL-Modellen in Alloy-Spezifikationen.

*Automatische, vollständige Lösung für begrenzten Zustandsraum*

*Logisches Fundament: Prädikatenlogik erster Stufe*

Für weitere Informationen bezüglich des Alloy-Rahmenwerks verweist der Autor auf die dedizierte Literatur [88] und bei Bedarf auf ein anschauliches Tutorial[6].

### 4.2.2 *Umformung des Simulationsproblems in ein Erfüllbarkeitsproblem*

In Abschnitt 3.2 wird das Prinzip der Ereignisorientierten Simulation vorgestellt. Die Simulationsdurchführung beinhaltet die *inkrementelle* Erzeugung einer künstlichen Prozessausführungsspur. Alloy erzeugt dagegen Prozessausführungsspuren immer stapelweise – also in Form *vollständiger* Prozessausführungsspuren. Im aktuellen Abschnitt wird daher kurz darauf eingegangen, wie eine Ereignisorientierte Prozesssimulation auf ein Erfüllbarkeitsproblem abgebildet werden kann. Dabei beschränken sich die Erläuterungen für diesen Abschnitt auf die zeitliche Reihenfolge der Durchführung von Prozessaktivitäten und auf gegenüber dem Prozessmodell konformen Ausführungsspuren.

*Batch-Simulation statt inkrementeller*

In klassischen Simulationsansätzen werden Ereignisprotokolle in diskreten Zeitschritten sukzessive aufgebaut. Grundlage dafür ist ein globaler Systemtakt. Übertragen auf die Prozesssimulation bedeutet das, dass für jeden Zeitschritt die Ausführung einer Aktivität protokolliert wird, bis das Prozessziel erreicht wird oder der Prozess abbricht. Die Entscheidung, welche Aufgabe im jeweiligen Zeitschritt simuliert wird, fällt lokal und basiert auf den Vorgaben des Prozessmodells. Bei imperativen Prozessmodellen ist es möglich, eine zeitliche Reihenfolge der Aktivitäten *direkt* abzuleiten, da die zugehörige Ordnung im Modell *explizit* kodiert ist. Das trifft auf deklarative Prozessmodelle nicht zu, da in diesen die vorgegebene Ordnung nur *implizit* in Form von Regeln angegeben ist. Folglich muss die zeitliche Reihenfolge der Ausführung der Prozessaktivitäten innerhalb einer Instanz so gewählt werden, dass sie alle dieser Regeln befolgt. Das Ermitteln solcher beispielhafter Instanzen entspricht der Ausführungssemantik des im vorherigen Abschnitt angesprochenen run-Kommandos. Dieses startet zunächst einen Transformationsprozess, welcher die Alloy-Spezifikation schrittweise in eine Boolesche Formel umwandelt. Die Ermittlung von Instanzen, die diese Formel erfüllen wird *Boolesches Erfüllbarkeitsproblem* (engl. *boolean satisfiability problem*), kurz *SAT*, genannt. Sogenannte SAT-Löser (engl. *SAT solver*) sind Werkzeuge, die vollautomatisch gültige Lösungen für Boolesche Formeln ermitteln, falls diese existieren. Die Existenz einer solchen Lösung besagt, dass die Formeln erfüllbar

*Globale statt lokale Entscheidungen*

*SAT: Boolesche Erfüllbarkeitsprobleme*

6 http://alloy.mit.edu/alloy/tutorials/online/, zuletzt aufgerufen: 02.01.2017

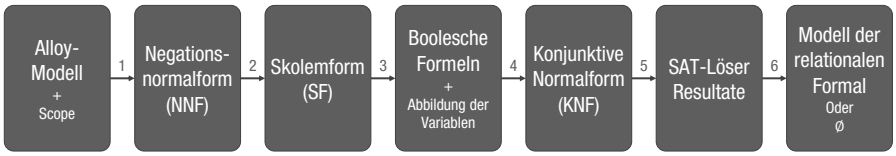



Abbildung 4.4: Transformation eines Alloy-Modells und „Lösen" mittels SAT-Löser

sind. Im Kontext der Prozesssimulation sind diese Lösungen die gesuchten künstlichen Ausführungsspuren eines Prozessmodells. Alloy selbst enthält eine Auswahl verschiedener SAT-Löser.

Wie bereits angedeutet, wird eine Alloy-Spezifikation in mehreren Schritten (Abbildung 4.4) in Boolesche Formeln übersetzt [87].

*Transformation in Boolesche Formeln Alloy →NNF*

Im ersten Schritt werden die relationalen Formeln der Alloy-Spezifikation in die *Negationsnormalform (NNF)* überführt. In dieser Form kommen Negationsoperatoren nur über atomaren Aussagen vor. Die Überführung in die Skolemform eliminiert mit einem Existenzquantor versehene Variablen. Diese beiden ersten Schritte vereinfachen die relationalen Formeln stark, dienen also der Berechnungsoptimierung. Die nachfolgenden Transformationen sind hingegen notwendig, da SAT-Löser eine bestimmte Art Boolescher Formeln erwartet. Daher werden in Schritt 3 alle relationalen Skolem-Formeln in eine einzelne Boolesche Formel übersetzt, wobei letztere genau dann eine Lösung hat, wenn auch die ursprüngliche Formel für den gegebenen Scope gelöst werden kann. Die Formeln sind demnach semantisch äquivalent. Die Übersetzung beinhaltet zusätzlich eine direkte Abbildung relationaler auf Boolesche Variablen, wobei diese Abbildung zurückzuverfolgen ist. Diese Abbildung ist notwendig, da Boolesche Formeln beispielsweise keine Mengen enthalten dürfen. Im vierten Schritt wird die entstandene Boolesche Formel in die *Konjunktive Normalform (KNF)* überführt. Diese Form ist das Eingabeformat, das von SAT-Lösern im Allgemeinen erwartet wird. Auf Algorithmen zum Lösen von Erfüllbarkeitsproblemen sei an dieser Stelle nicht weiter eingegangen, da die existierenden Implementierungen unverändert wiederverwendet werden können. Das Ergebnis des fünften Schrittes ist entweder eine leere Lösungsmenge oder eine Menge von „Modellen" (*SAT-Lösungen*). Jedes dieser Modelle weist jeder Booleschen Variable einen Wert zu. Schließlich erfolgt im sechsten Schritt die *Rekonstruktion der relationalen Modelle*. Dazu wird die in Schritt 3 verwendete Abbildung relationaler auf Boolesche Variablen zurückverfolgt. Falls für die Ausführung das run-Kommando verwendet wird, ist das Ergebnis folglich eine Menge relationaler Modelle, welche die relationale Alloy-Spezifikation erfüllen. Für das check-Kommando erfolgen alle Schritte analog, mit dem Unterschied, dass die Booleschen Formeln negiert werden. Die Menge der durch diese Analyse generierten Lösungen ist bezüglich des gewählten Scopes vollständig. Jede Lösung besteht aus Tupeln von Atomen, deren Semantik durch die ursprüngliche Alloy-Spezifikation festgelegt ist.

*Skolemisierung*

*Boolesche Formeln →KNF*

*Rekonstruktion relationaler Modelle*

### 4.2.3 *Statischer Teil des Simulationsmodells: Metamodell für Prozessausführungsspuren in Alloy*

In der einleitenden Beschreibung des Simulationsansatzes anhand von Abbildung 4.3 wird erwähnt, dass das Simulationsmodell in Alloy aus einem statischen und einem dynamischen Teil besteht. Der statische Teil wird als *Metamodell für Prozessausführungsspuren* bezeichnet und wird nun beschrieben.

*Metamodell für Prozessausführungsspuren*

Der nachfolgend vorgestellte Ansatz unterstützt derzeit vier Prozessperspektiven, namentlich die *(i)* Funktionale, *(ii)* die Verhaltensorientierte, *(iii)* die Organisatorische und die *(iv)* Datenorientierte Perspektive. Die Operationale Perspektive wird hier nicht betrachtet, da die dafür notwendigen Konzepte mit denen der übrigen Perspektiven nahezu deckungsgleich sind und somit keine weiteren Erkenntnisse erbringen. Auch der Lebenszyklus von Aktivitäten wird vorerst nicht berücksichtigt.

Prozessausführungsspuren werden in Abschnitt 3.3.1 formal definiert. Die Beschreibung in Codeausschnitt 4.2 basiert auf dieser Definition. In der ersten Zeile wird das aktuelle Modul benannt und in Zeile 2 und 3 werden zwei weitere „geöffnet". Das bedeutet, dass alle darin enthaltenen Signaturen, Funktionen und andere referenzierbare Konstrukte auch im aktuellen Modul bekannt sind. Die Signatur `PEvent` definiert eine abstrakte „Klasse", welche das Rückgrat der Kettenstruktur der Prozessausführungsspuren bildet. Jedem `PEvent` ist dabei eine eineindeutige Position (`pos`) innerhalb der Kette zugeordnet. Eine intuitive, alternative Umsetzung ist eine einfach oder doppelt verkettete Liste. Allerdings haben Leistungstests ergeben, dass die hier vorgeschlagene Variante deutlich performanter ist. Die Signatur ist als abstrakt gekennzeichnet. Dieses Konzept ist aus der OOP bekannt und verhindert eine Instanziierung. In Alloy ist es zwar trotzdem möglich, Instanzen für abstrakte Signaturen zu erzeugen, was aber verhindert wird, sobald andere Signaturen die abstrakte Signatur erweitern.

*Rückgrat der Kettenstruktur*

Die `TaskEvent`-Signatur erweitert `PEvent` und kapselt Ereignisse, die mit der Ausführung einer Aktivität assoziiert sind. Diese Assoziation wird über eine generische Schnittstelle – ebenfalls eine Signatur – namens `AssociatedElement` realisiert. Diese Schnittstelle stammt aus dem Modul `traceCommons`, welches in Codeausschnitt 4.3 dargestellt ist. Der Ausdruck `#(Task ∩ assoEl) = 1` ist ein direkt mit einer Signatur assoziierter Fakt und enthält eine relationale Operation, die dafür sorgt, dass ein *TaskEvent* mit genau *einer* Aktivität – hier als `Task` bezeichnet – assoziiert wird. Auch diese Signatur ist abstrakt, was die tatsächlich später in den Instanzen auftretenden Ereignisse derzeit auf die letzte Erweiterung der Ereignissignaturen beschränkt – die `HumanTask`. Diese beschreibt die Ausführung einer Aktivität durch genau eine Person. Alternativ ist es auch möglich, für jedes gewünschte Attribut eines Ereignisses ein eigenes Feld innerhalb der Task-Signaturen zu definieren. Die hier vorgeschlagene Variante entspricht jedoch des in Abschnitt 3.3.1 beschriebenen generischen Aufbaus eines Ereignisses, das sich aus Schlüssel-Wert-Paaren zusammensetzt. Dadurch lassen sich die mit den Ereignissen assoziierten Informationen ebenso generisch anfragen, was in Abschnitt 4.2.4 deutlich wird. Das einzige benannte Feld ist die von `PEvent` vererbte Positionsangabe.

*Kapselung von Ereignissen*

```
module traceMM
open traceCommons
open orgMM

abstract sig PEvent { pos: disj Int }
abstract sig TaskEvent extends PEvent{ assoEl: some AssociatedElement }
{
    #(Task ∩ assoEl) = 1
}
sig HumanTaskEvent extends TaskEvent{}{
    #(Identity ∩ assoEl) = 1
}
abstract sig Task extends AssociatedElement{}
abstract sig DataObject extends AssociatedElement{}
abstract sig Variable extends DataObject{
    value: TaskEvent -> lone univ
}{
    value[TaskEvent] in (String + Int)
}
abstract sig Document extends DataObject{}

fact{
    ∀ e: PEvent | e.pos < (int[Int] + #PEvent)
}
fact{ ∀ e: TaskEvent | #(e.assoEl ∩ Group) = 0 }

fact {
    ∀ ae: (AssociatedElement - Group) | ae in TaskEvent.assoEl
}
fact {
    ∀ e: TaskEvent, v: Variable | v in e.assoEl ↔ v.value[e] ≠ none
}

fun search(asso: AssociatedElement): set TaskEvent {
    { e: TaskEvent | asso in e.assoEl }
}
fun searchBefore(cur: TaskEvent, asso: AssociatedElement): set TaskEvent{
    { e: TaskEvent | e.pos < cur.pos and asso in e.assoEl }
}
fun searchAfter(cur: TaskEvent, asso: AssociatedElement): set TaskEvent{
    { e: TaskEvent | e.pos > cur.pos and asso in e.assoEl }
}
fun searchB(a,c: TaskEvent, ae: AssociatedElement): set TaskEvent {
    { e: TaskEvent | e.pos > a.pos and e.pos < c.pos and ae in e.assoEl }
}
fun searchAtPos(asso: AssociatedElement, givenPos: Int): set TaskEvent {
    { e: TaskEvent | e.pos = (int[givenPos]) and asso in e.assoEl }
}
fun searchLastBefore(cur: TaskEvent, ae: AssociatedElement): lone TaskEvent{
    { e: TaskEvent | ∀ o: TaskEvent | ae in e.assoEl and e.pos < cur.pos and (ae in
        o.assoEl → e = o or o.pos > e.pos) }
}
fun eventAtPos(givenPos: Int): one PEvent{
    { e: PEvent | e.pos = givenPos }
}
```

Codeausschnitt 4.2: Metamodell für Prozessausführungsspuren

```
module traceCommons

abstract sig AssociatedElement{}
```

Codeausschnitt 4.3: Modul `traceCommons`

*Elemente der Datenorientierten Perspektive*

Die verbleibenden drei abstrakten Signaturen `DataObject`, `Variable` und `Document` ermöglichen die Berücksichtigung der Datenorientierten Perspektive. Die hier getroffene Unterscheidung zwischen einer Variable und einem Dokument entstammt der abstrakten DPIL-Syntax (Abschnitt 3.1.5). Variablen können hier vorerst nur ganze Zahlen oder Zeichenketten als Werte besitzen, da Alloy keine reellen Zahlen unterstützt. Da Alloy keine explizite Mehrfachvererbung unterstützt, muss für die Möglichkeit, Atome sowohl aus `String` als auch aus `Int` als Werte annehmen zu können, das `value`-Feld vom Typ `univ` sein. Die Beschränkung auf *ausschließlich* diese beiden Mengen wird durch den direkt an die Signatur angehängten Fakt (Zeile 18) realisiert. Die Zuweisung eines Datenobjekts zu einem Ereignis bedeutet hier, dass das Datenobjekt in der angegebenen Form innerhalb der Durchführung der ebenfalls referenzierten Aufgabe produziert wurde. Differenzierter interpretiert beschreibt das Ereignis entweder die Zuweisung eines Werts zu einer Variable oder das Erzeugen eines Dokuments.

Sowohl die aktivitäts- als auch die datenbezogenen Signaturen sind abstrakt und werden auch von keiner anderen Signatur des Moduls spezialisiert. Die Erweiterung dieser Signaturen findet stattdessen im dynamischen Teil statt und ist im Sinne der OOP als Instanziierung zu verstehen.

*Fakten zur weiteren Beschreibung der Kettenstruktur*

Nachfolgend enthält das Metamodell drei Fakten. Der erste beschreibt die Kettenstruktur der Prozessausführungsspur genauer, indem er festlegt, dass die Positionen aller `PEvent`s fortlaufend nummeriert sind und bei der niedrigsten zur Verfügung stehenden Zahl (`int[Int]`) beginnen. Der zweite Fakt sorgt dafür, dass Gruppenzugehörigkeiten von beispielsweise Identitäten nicht in der Ereignisaufzeichnung selbst widergespiegelt werden, da diese bereits durch die im organisatorischen Metamodell festgelegten Strukturen repräsentiert werden. Der letzte Fakt schließlich sorgt dafür, dass alle `AssociatedElement`-Atome nur in Verbindung mit einem Ereignis auftreten.

*Hilfsfunktionen zur Modellabfrage*

An die Fakten schließen sich mehrere Funktionen an, die als Hilfsfunktionen für die Formulierung der Prozessregeln im dynamischen Teil verwendet werden. Dies erlaubt später eine prägnante, lesbare Schreibweise der Regeln in Alloy. Jede der Funktionen „sucht" innerhalb der Prozessausführungsspur nach dem als Parameter übergebenen `AssociatedElement`. Die erste Funktion ermittelt alle prinzipiell existierenden Vorkommen des fraglichen Elements. Die darauffolgenden zwei Funktion liefern nur die Teilmenge davon, die ausgehend von einem bestimmten Ereignis *vor* respektive *nach* selbigem auftreten. Auch das Vorkommen zwischen zwei spezifischen Ereignissen oder an einer bestimmten Position kann ermittelt werden, wofür die Funktionen `searchB` und `searchAtPos` verwendet werden können. Die letzte Hilfsfunktion liefert für eine gegebene Position das zugehörige Ereignis.

Das bereits angesprochene und in Codeausschnitt 4.4 dargestellte Metamodell für Organisationsstrukturen ist die Umsetzung des von Bussler entworfenen

```
module orgMM

open traceCommons

abstract sig Relation {
    subject: one Element,
    object: one Element,
    predicate: one RelationType
}

abstract sig Element extends AssociatedElement{}
abstract sig Identity extends Element{}
abstract sig Group extends Element{}
abstract sig RelationType{}

fact{
    ∀ g: Group | g in Relation.object
}

fun groupOf(id:Identity, rt:RelationType):set Group{
    { g:Group | some r:Relation | r.predicate = rt and r.subject = id and r.
        object in Group }
}
fun relationOf(e1,e2:Element):set RelationType{
  { rt:RelationType | some r:Relation | r.predicate = rt and r.subject = e1
      and r.object = e2 }
}
```

Codeausschnitt 4.4: Modul `orgMM`: Organisatorisches Metamodell nach Bussler [38]

kompakten aber ausdrucksstarken organisatorischen Metamodells [38]. Da dieses Metamodell auch in DPIL integriert ist (Abbildung 3.6), ist es naheliegend, auch in Alloy auf dieselbe Repräsentation organisatorischer Strukturen zurückzugreifen. Die bereits bekannte Schnittstelle `AssociatedElement` wird auch hier mittels `open traceCommons` zur Verfügung gestellt. Jede der enthaltenen Signaturen ist atomar – mit Ausnahme der Signatur `Relation`. Diese enthält drei Felder, mit denen die bekannte Subjekt-Prädikat-Objekt-Beziehung beschrieben werden kann. Dabei können sowohl Identitäten als auch Gruppen die Subjekt- bzw. Objektrolle einnehmen. Das wird über die zusätzliche Abstraktionsstufe `Element` sichergestellt. Folglich ist sowohl eine Beziehung wie `Max (Subjekt) hasRole (Prädikat) Admin (Objekt)` als auch eine Relation wie `Admin (Subjekt) memberOf (Prädikat) Employee (Objekt)` mit dieser Form realisierbar. Die Möglichkeit zur Assoziation mit Ereignissen wird dadurch realisiert, dass `Element` die `AssociatedElement`-Signatur erweitert. Anzumerken ist hier, dass für die aktuelle Implementierung nur `Identity` von `AssociatedElement` erben müsste. Dieses erbt aber bereits von `Element` und Mehrfachvererbung wird von Alloy nicht direkt unterstützt.

*Organisatorisches Metamodell in Alloy*

Der einzige Fakt des organisatorischen Metamodells legt fest, dass eine Gruppe nur dann Teil einer simulierten Prozessinstanz sein kann, wenn andere organisatorische Einheiten Teil dieser Gruppe sind.

Auch dieses Modul verfügt über Hilfsfunktionen, wobei die erste für eine gegebene Identität und ein ebenfalls anzugebendes Prädikat die Gruppen ermittelt, zu denen die Identität gehört. Die zweite ermittelt die Beziehung zwischen zwei

Elementen. Beispielsweise kann so die Information beschafft werden, dass eine Identität der Mentor einer anderen ist.

P14

Da die Modellierung der mit einem Prozess assoziierten Organisationsstruktur nicht Teil der Modellierungssprache DPIL ist, müssen Informationen über organisatorische Ressourcen sowie deren Gruppierung und Beziehungen aus externen Quellen bezogen werden. Da hierfür bislang kein Konzept existiert, empfiehlt es sich, das Organisationsmodell zum aktuellen Zeitpunkt als Konfigurationsparameter zu konzipieren. Die Markierungen P1, P2, etc. verknüpfen das Konzept des jeweiligen Konfigurationsparameters mit der zugehörigen Implementierung in Abschnitt 7.3.

### 4.2.4 *Dynamischer Teil: Transformation von DPIL-Modellen in Alloy-konforme Simulationsmodelle*

Ausgangspunkt für die Simulation ist ein DPIL-Modell. Die Sprache zur Erstellung von Simulationsmodellen ist jedoch nicht DPIL, sondern Alloy, sodass eine Abbildung notwendig ist. Diese basiert entgegen der üblichen Vorgehensweise auf einer Modell-zu-Text-Transformation, was in Abschnitt 4.2.4.1 begründet ist. Für die Beschreibung der Transformation selbst wird die Struktur der DPIL-Modelle in Prozessentitäten (z.B. Aktivitäten und Ressourcen), Prozessregeln und als Meilensteine formulierte Prozessziele aufgeteilt. In eben diese Unterabschnitte gliedert sich der verbliebene Teil dieses Abschnitts.

#### 4.2.4.1 *Allgemeines Prinzip*

In der Regel werden Abbildungen auf Basis der Metamodelle der fraglichen Sprachen als *Modell-zu-Modell-Transformationen* definiert (Abschnitt 3.5). Das setzt aber voraus, dass jeweils ein Metamodell, welches die abstrakte Syntax der Sprache beschreibt, für Quell- und Zielmodell explizit definiert ist. Das trifft zwar auf DPIL (Abschnitt 3.1.5), nicht aber auf Alloy zu. Alloy wird lediglich mittels einer EBNF[7]-Grammatik beschrieben. Diese definiert jedoch nur die konkrete Syntax einer Sprache. Zwar gibt es Methoden, die aus einer EBNF-Grammatik automatisch eine passende abstrakte Syntax ableiten können (z.B. [195]), allerdings stellt dies für das aktuelle Problem einen unnötigen Umweg dar. Einige wichtige Rahmenbedingungen für die Abbildung von DPIL auf Alloy sind:

*Rahmenbedingungen für die Abbildung von DPIL-Regeln auf Alloy*

- Die Translation erfolgt *ausschließlich* von DPIL nach Alloy, ist also unidirektional.
- DPIL und Alloy ähneln sich in Syntax und Semantik aufgrund der gemeinsamen Basis der FOL.
- Die Zielsprache ist textuell.
- Die Abbildung kann als Codegenerierung betrachtet werden, da sie stattfindet, um nicht-maschinenausführbare DPIL-Modelle in maschinenausführbare Repräsentationen zu übersetzen. Dieser Prozess wird auch als Kompilierung bezeichnet und kann mehrstufig sein.

7 EBNF = Erweiterte Backus-Naur-Form

*Modell-zu-Text-Transformation*

In vielen Fällen, wie auch in diesem, ist die *Modell-zu-Text-Transformation (M2T)* eine Alternative zur Modell-zu-Modell-Transformation. Die Literatur gibt nur vage Aufschluss darüber, welche der Alternativen in welchen Fällen anzuwenden ist. Das Fehlen des Metamodells der Zielsprache und die textuelle Repräsentation derselben sind jedoch vergleichsweise starke Indikatoren für die Eignung eines M2T-Ansatzes [68].

P1

Im aktuell diskutierten Ansatz wird die von der OMG standardisierte *MOF Model to Text Transformation Language (MOF2Text)* [140] verwendet, um DPIL-Modelle in textuelle Alloy-Spezifikationen[8] zu übersetzen. Voraussetzung ist, dass das Quell-Metamodell als Instanz des MOF[9]-Meta-Metamodells vorliegt. Das DPIL-Metamodell ist als *Ecore*-Modell realisiert, wobei Ecore auf einer Teilmenge des MOF-Meta-Metamodells basiert. Damit ist MOF2Text in der Lage die Abbildung von DPIL-Modellen auf Alloy-Spezifikationen zu realisieren. Der verbliebene Teil dieses Abschnitts beschäftigt sich mit eben dieser Abbildung.

Auf Basis des Metamodells für Prozessausführungsspuren aus Abschnitt 4.2.3 wird nachfolgend die Transformation eines DPIL-Prozessmodells in eine Alloy- und zum besagten Metamodell konforme Repräsentation[10] beschrieben. Dabei werden zunächst die Entitäten des Prozesses transformiert und danach erfolgt die Abbildung der Prozessregeln. Wie in Abschnitt 3.5 erläutert, werden die Abbildungsvorschriften als Regeln auf Ebene des DPIL-Metamodells definiert, sodass auch bei verschiedenen konkreten Syntaxen die Transformation möglich ist. Zum besseren Verständnis wird das zu transformierende DPIL-Konstrukt mittels der textuellen *konkreten* Syntax dargestellt. Die Transformation selbst ist jedoch auf Basis von DPILs *abstrakter* Syntax definiert, was im späteren Verlauf des Abschnitts am Beispiel des DPIL-Makros `sequence` erläutert wird. Die Abbildung der DPIL-Konstrukte auf Alloy erfolgt auf Basis textueller Schablonen, wobei die dynamischen Inhalte durch $-Symbole eingeschlossen sind. Der in diesen Symbolen stehende Ausdruck beschreibt einen Zugriffspfad auf Informationen aus dem abstrakten Syntaxbaum des DPIL-Konstrukts. Die Befüllung mit den dynamischen Inhalten erfolgt durch Ersetzung der so markierten Bestandteile durch das Ergebnis des Zugriffs für die konkrete Instanz, also zum Zeitpunkt der Ausführung der Transformationsregeln.

#### 4.2.4.2 *Abbildung der Prozessentitäten*

*1:1-Abbildung für Entitäten*

Die Transformation der Prozessentitäten beinhaltet die notwendigen Abbildungen von Aktivitäten, Dokumenten, Variablen sowie der organisatorischen Ressourcen und Gruppen (Tabelle 4.2). Somit wird jede in einem Prozessmodell enthaltene Aktivität (`task`) mit dem Namen `a` in eine Erweiterung der `Task`-Signatur transformiert. Der Name der Erweiterung ist dabei der Name der Aktivität selbst. Analog wird mit allen übrigen Entitäten verfahren, für die im Metamodell für Prozessausführungsspuren jeweils eine spezifische Signatur enthalten ist (Abschnitt 4.2.3).

8 Die Markierungen P1, P2, etc. werden zur Assoziation mit konfigurierbaren Parametern aus dem Implementierungsteil verwendet.

9 MOF = Meta Object Facility

10 Diese ist zwar kein Modell im eigentlich Sinne, sondern eine textuelle Spezifikation. Allerdings wird vereinfacht dennoch der Begriff „Zielmodell" verwendet.

| DPIL | Alloy |
|---|---|
| `task a` | `one sig $a.name$ extends Task{}` |
| `document d` | `one sig $d.name$ extends Document{}` |
| `variable v` | `one sig $v.name$ extends Variable{}` |
| `use identity i` | `lone sig $i.name$ extends Identity{}` |
| `use group g` | `lone sig $g.name$ extends Group{}` |

Tabelle 4.2: Abbildung von DPIL-Entitäten auf Alloy-Signaturen

*Namenseindeutigkeit*

Eine Grundvoraussetzung für die Validität des Abbildungsergebnisses ist die Eindeutigkeit der abgeleiteten Namen aller erzeugten Signaturen. Dies ist jedoch dadurch sichergestellt, dass DPIL typübergreifend die Eindeutigkeit der Namen aller Prozessentitäten fordert. Das Ergebnis der Abbildung ist eine Menge von Signaturen, welche bereits eine gültige Alloy-Spezifikation darstellen. Eine Simulationsdurchführung an dieser Stelle würde beliebige Reihenfolgen der Durchführung von Aktivitäten sowie beliebige Assoziationen zwischen diesen und allen anderen Prozessentitäten generieren.

#### 4.2.4.3 *Abbildung der Prozessregeln*

Auf Basis der Abbildungen aller Prozessentitäten werden nun die Prozessregeln des DPIL-Modells transformiert. Da sowohl DPIL als auch Alloy auf Prädikatenlogik erster Stufe basieren, ist eine generische Abbildung denkbar. Die Transformation selbst liegt jedoch nicht im Fokus der vorliegenden Arbeit, weswegen hier lediglich eine vereinfachte Abbildung auf Basis der in Abschnitt 3.1.5 vorgestellten Bibliothek an Modellierungsmustern – Makros im DPIL-Kontext – beschrieben wird. Dies bedeutet, dass nur die Prozessregeln eines DPIL-Modells in eine Alloy-Repräsentation überführt werden können, die den Aufruf eines der in Tabelle 4.3 aufgelisteten Makros darstellen. Im Wesentlichen wird jedes dieser Makros in ein Alloy-Prädikat übersetzt. Eine Ausnahme hierbei bildet das Makro `serializeAll`, welches für das in Abschnitt 4.2.3 vorgestellte Metamodell für Prozessausführungsspuren keinerlei Relevanz hat. Grund ist, dass dieses Makro eine rein sequentielle Bearbeitung aller Aktivitäten des Prozesses vorschreibt, was jedoch nur Auswirkungen hat, wenn hinsichtlich des Aktivitätslebenszyklus zwischen verschiedenen Zuständen einer Aktivität unterschieden werden kann. Das im Rahmen der vorliegenden Arbeit beschriebene Spurmetamodell unterstützt diese Unterscheidung jedoch bewusst nicht.

*1:1 Abbildung für Makros*

*Strukturelle Mustersuche*

Für die Identifikation der passenden Abbildungsregel wird die zu einem Makro gehörende Regel in ihrer abstrakten Syntax als Muster verwendet. Der Abgleich von Muster und Instanz wird als *strukturelle Mustersuche* (engl. *structural pattern matching*) bezeichnet. Das Prinzip ist unter anderem aus dem Bereich der deklarativen Programmierung bekannt. Es bietet hier den Vorteil, dass die Namensgebung der Makros irrelevant ist. Für jede DPIL-Regel, welche einem der definierten Muster gleicht, wird im Zielmodell der zu diesem Muster gehörende Alloy-Ausdruck materialisiert. Wie bereits eingangs erwähnt, sind die Muster in Tabelle 4.3 mittels DPILs konkreter Syntax formuliert, was aber lediglich der Verbesserung der Verständlichkeit dient. Die Abbildung eines textuellen DPIL-Ausdrucks auf die zugehö-

| DPIL-Makro | Alloy |
|---|---|
| `sequence(a,b)` | `pred sequence[a,b: Task]{`<br>`  ∀ e: TaskEvent \| b in e.assoEl → #searchBefore[e,a]>0`<br>`}` |
| `directSequence(a,b)` | `pred directSequence[a,b: Task]{`<br>`  ∀ e: TaskEvent \| b in e.assoEl → #searchAtPos[e,int[e.pos-1]]>0`<br>`}` |
| `response(a,b)` | `pred response[a,b: Task]{`<br>`  ∀ e: TaskEvent \| a in e.assoEl → #searchAfter[e,b]>0`<br>`}` |
| `once(a)` | `pred once[a: Task]{`<br>`  ∀ e: TaskEvent \| b in e.assoEl → #searchBefore[e,a]=0`<br>`}` |
| `serializeAll` | keine Entsprechung in Alloy notwendig |
| `local(a,o)` | `pred local[a: Task,o: DataObject]{`<br>`  ∀ e: TaskEvent \| o in e.assoEl → a in e.assoEl`<br>`}` |
| `produces(a,o)` | `pred produces[a: Task,o: DataObject]{`<br>`  ∀ e: TaskEvent \| a in e.assoEl → o in e.assoEl`<br>`}` |
| `consumes(a,o)` | `pred consumes[a: Task,o: DataObject]{`<br>`  ∀ e: TaskEvent \| a in e.assoEl → #searchBefore[e,o]>0`<br>`}` |
| `direct(a,i)` | `pred direct[a: Task,i: Identity]{`<br>`  ∀ e: TaskEvent \| a in e.assoEl → i in e.assoEl`<br>`}` |
| `role(a,r)` | `one sig hasRole extends RelationType{}`<br>`pred role[a: Task,r: Group]{`<br>`  ∀ e: TaskEvent \| a in e.assoEl →`<br>`  #(r ∩ groupOf[Identity ∩ e.assoEl,hasRole]) > 0`<br>`}` |
| `separate(a,b)` | `pred separate[a,b: Task]{`<br>`  ∀ e1,e2: TaskEvent \| a in e1.assoEl and b in e2.assoEl →`<br>`  #(Identity ∩ e1.assoEl ∩ e2.assoEl) > 0`<br>`}` |
| `binding(a,b)` | `pred binding[a,b: Task]{`<br>`  ∀ e1,e2: TaskEvent \| a in e1.assoEl and b in e2.assoEl →`<br>`  #(Identity ∩ e1.assoEl ∩ e2.assoEl) = 0`<br>`}` |
| `caseHandling` | `pred caseHandling{`<br>`  ∀ e1,e2: TaskEvent \| #(Identity ∩ e1.assoEl ∩ e2.assoEl) = 0`<br>`}` |
| `bindingRole(a,b)` | `pred bindingRole[a,b: Task]{`<br>`  ∀ e1,e2: TaskEvent \| a in e1.assoEl and b in e2.assoEl`<br>`  → groupOf[Identity ∩ e1.assoEl,hasRole] = groupOf[Identity ∩ e2.assoEl,`<br>`  hasRole]`<br>`}` |
| `capability(a,rt,g)` | `pred capability[a: Task,rt: RelationType,g: Group]{`<br>`  ∀ e: TaskEvent \| a in e.assoEl → groupOf[Identity ∩ e.assoEl,rt] = g`<br>`}` |
| `orgDistMulti(a,b,rt)` | `pred orgDistMulti[a,b: Task,rt: RelationType]{`<br>`  ∀ e1,e2: TaskEvent \| a in e1.assoEl and b in e2.assoEl →`<br>`  #relationOf[Identity ∩ e1.assoEl,Identity ∩ e2.assoEl,rt]`<br>`}` |
| `roleSequence(a,b,g)` | `one sig hasRole extends RelationType{}`<br>`pred roleSequence[a,b: Task,g: Group]{`<br>`  ∀ e: TaskEvent \| b in e.assoEl and groupOf[Identity ∩ e.assoEl,hasRole] = g`<br>`  → #searchBefore[e,a] > 0`<br>`}` |
| `resourceSequence(a,b,i)` | `one sig hasRole extends RelationType{}`<br>`pred resourceSequence[a,b: Task,i: Identity]{`<br>`  ∀ e: TaskEvent \| b in e.assoEl and (Identity ∩ e.assoEl) = i`<br>`  → #searchBefore[e,a] > 0`<br>`}` |
| `dataSequence(a,b,o,v)` | `pred dataSequence[a,b: Task,o: Variable,val: univ]{`<br>`  ∀ e: TaskEvent \| b in e.assoEl and let v = o ∩ e.assoEl \|`<br>`  v.value[e] = val → #searchBefore[e,a]>0`<br>`}` |
| `dataOrdering(o1,o2)` | `pred dataOrdering[o1,o2: DataObject]{`<br>`  ∀ e: TaskEvent \| o2 in e.assoEl → #searchBefore[e,o1]>0`<br>`}` |
| `mpSequence(o,val,r1,r2,b)` | `pred mpSequence[o: Var,val: univ,r1,r2: Group,b: Task]{`<br>`  ∀ e: TaskEvent \| let ae = e.assoEl \| b in ae and let v = o ∩ ae \|`<br>`  v.value[e] = val and groupOf[Identity ∩ ae,hasRole]=r1 → let suc =`<br>`  searchAfter[e,b] \| #suc>0 and groupOf[Identity ∩ suc.assoEl,hasRole]=r2`<br>`}` |

Tabelle 4.3: Abbildung von DPIL-Makros auf Alloy-Prädikate

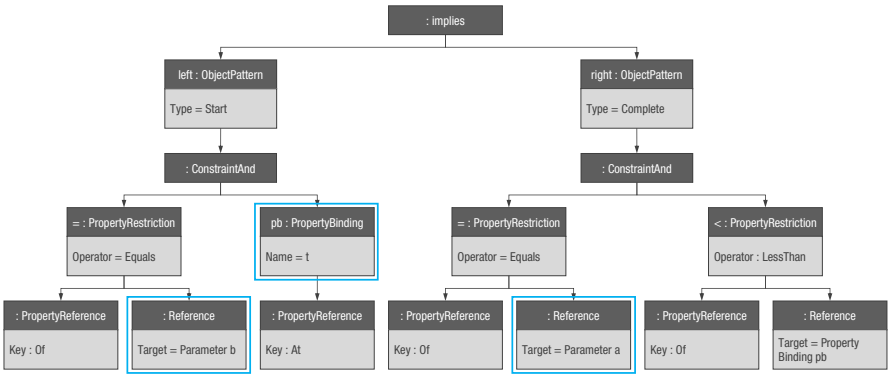



Abbildung 4.5: Abstrakter Syntaxbaum (AST) der DPIL-Regel `sequence`

rige abstrakte Syntax ist immer eindeutig, weswegen diese Vereinfachung zulässig ist. Aus diesem Grund wird die musterbasierte Abbildung von DPIL-Makros auf Alloy-Sprachmittel lediglich exemplarisch anhand *eines* der Makros in seiner abstrakten Syntax erläutert. Das Beispiel (Abbildung 4.5) repräsentiert die abstrakte Syntax des Regelteils des bereits aus Tabelle 4.3 bekannten `sequence`-Makros.

*Abgleich auf Basis des ASTs*

Der Musterabgleich besteht lediglich aus einer simultanen Traversierung der entsprechenden abstrakten Syntaxbäume des zu transformierenden Makros und eines der Muster. Die Traversierung kann vom Wurzelknoten aus schrittweise in beliebiger Reihenfolge durchgeführt werden. In jedem Schritt werden die aktuell besuchten Knoten beider abstrakter Syntaxbäume verglichen. Diese beiden Knoten werden nachfolgend als *korrespondierend* bezeichent. Unterscheiden die korrespondierenden Knoten sich hinsichtlich ihres Typs oder enthaltener Eigenschaften, dann gelten Muster und zu transformierendes Makro als unterschiedlich und es wird dieser Prozess mit dem nächsten Muster wiederholt. Unterscheiden sich die aktuell untersuchten Knoten nicht, dann werden in der gewählten Traversierungsreihenfolge die nächsten Knoten ausgewählt und verglichen. Dies wird solange fortgesetzt, bis jeder Knoten von Muster und Makro besucht wurde. Enthält einer der beiden abstrakten Syntaxbäume weniger Knoten als der andere, dann gelten beide Bäume von vorn herein als nicht deckungsgleich. Somit müssen einerseits Struktur und andererseits die Inhalte von Makro und Muster übereinstimmen.

*Parametrierbare Muster*

Parametrierbare Muster unterstützten in Bezug auf die Benennung ihrer Parameter respektive Eigenschaftsbindungen (`PropertyBinding`) auch Variabilität. Die in Abbildung 4.5 farblich eingerahmten Knoten beinhalten Eigenschaften, deren Benennung als variabel eingestuft ist. Für die Eigenschaftsbindung `pb`, welche in diesem Fall einer Variable `t` den aktuellen Zeitstempel zuordnet, ist die Namensgebung der Variable irrelevant. Der Knoten eines zu transformierenden Makros gilt bezüglich `pb` als gleich, wenn er ebenfalls vom Typ Eigenschaftsbindung ist. Der Attributwert für `Name` kann sich dabei vom Musterknoten unterscheiden. Ähnlich verhält es sich mit den zwei linken Blattknoten vom Typ `Reference`. Diese referenzieren die Parameter des Makrokopfes (Tabelle 4.3), die in diesem Fall `a` und `b` genannt werden. Auch hier dürfen sich für den Knotenabgleich die Namen der Parameter unterscheiden. Nicht unterscheiden dürfen sich jedoch der Typ der Referenz – hier `Parameter` – und die Reihenfolge der Parameter im Makro-Kopf. Das Muster ist zudem auch so generisch, dass es auch für die Definition für

*Generizität der Muster*

Prozessregeln ohne Makrodefinition verwendet werden kann. In diesem Fall existieren keine Parameter als Referenzziele. Stattdessen würden direkt Aktivitäten referenziert werden, wodurch sich lediglich der Typ von `Target` in den beiden linken Referenzen auf `Task` verändert. Damit kann auch eine `sequence`-Regel ohne Makrodefinition auf Alloy abgebildet werden – wobei das Ergebnis wiederum die Definition und Verwendung eines Prädikats mit dem Namen `sequence` ist.

*Beschränkung auf verpflichtende Regeln*

Makros sind lediglich Regelschablonen, die der kompakteren und lesbareren Darstellung von DPIL-Modellen dienen. Ihre Verwendung für Regeldefinitionen folgen dem Schema `ensure makro-name(paramVal1,paramVal2)`. DPIL unterscheidet zwischen verpflichtenden Regeln (`ensure`) und Empfehlungen (`advise`). Da diese Unterscheidung in Alloy aber nicht möglich ist, wird im aktuellen Abschnitt lediglich die Transformation verpflichtender Regeln erläutert[11]. Beispielsweise ist `ensure sequence(AntragStellen,AntragPrüfen)` eine verpflichtende Regel. Dies bedeutet die Zuordnung der jeweiligen Aktualparameter zu den im Makro angegebenen Formalparametern und intern die Instanziierung der im Makro kodierten Regelschablone mit den eingesetzten Aktualparametern. Folglich kann, wie zuvor bereits für die Makros beschrieben, auf den abstrakten Syntaxbaum der Regel zugegriffen werden, um auch in Alloy den Formalparametern des jeweiligen Prädikats die entsprechenden Aktualparameter zuzuordnen. Diese Zuordnung erfolgt durch die Verwendung des Prädikats in einem Alloy `fact` oder einem `assert`. Die genaue Verwendung wird in den Abschnitten Abschnitt 4.2.5 respektive Abschnitt 4.2.6 beschrieben.

*Flexibilität durch generisches Metamodell und Hilfsfunktionen*

Die in Tabelle 4.3 aufgelisteten Makros zeigen nicht nur exemplarisch die Flexibilität von DPIL selbst, sondern auch die Eignung von Alloy als Modellierungssprache für aus DPIL-Modellen abzuleitende Simulationsmodelle. Rein auf den Kontrollfluss orientierte Regelvorlagen, wie beispielsweise `sequence)`, lassen sich ebenso in Alloy formulieren, wie perspektivenübergreifende Restriktionen (beispielsweise `produces` oder `role`). Grundlage für die Transformationen sind in vielen Fällen die generischen Hilfsfunktionen wie beispielsweise `searchBefore` und `searchAfter` und die ebenso generischen Beziehungen Assoziationen der Prozessentitäten durch die Signatur `AssociatedElement` und die Relation `assoEl` der Signatur `TaskEvent`. Das ebenfalls in Abschnitt 4.2.3 beschriebene Metamodell für organisatorische Strukturen ermöglicht unter anderem die Umsetzung eines Rollenkonzepts für Prozessressourcen. Hierfür wird eine zusätzliche Signatur mittels `sig hasRole extends RelationType{}` eingeführt, welche beispielsweise vom `role`-Makro genutzt wird. Sind im konkreten DPIL-Modell mehrere Makros enthalten, welche in ihrer Alloy-Entsprechung diese Signatur benötigen, dann wird diese jedoch nur einmal im Zielmodell erzeugt. Das Makro `dataSequence` beschreibt die Abhängigkeit der Einhaltung einer Aktivitätsreihenfolge von einem konkreten Datenwert. Da das Spur-Metamodell für Variablen derzeit nur Zeichenketten und ganzzahlige Werte gestattet, ist auch die Transformation dieses Makros nur möglich, wenn für `v` eine Zeichenkette oder ein ganzzahliger Wert angegeben wird.

11 Die Berücksichtigung von Empfehlungen wird gesondert in Abschnitt 4.2.8 diskutiert.

```
fact{
  milestones[PEvent]
  ∀ e1,e2: PEvent|milestones[e1 + searchBefore[e1,none]] → e2.pos ⩽ e1.pos
}

pred milestones[es: set PEvent]{
  /*milestone 1*/
  /*milestone 2*/
  /* Beispiel: */
  some e: es|C in e.assoEl or D in e.assoEl
}
```

Codeausschnitt 4.5: Alloy-Schablone für DPIL-Meilensteine

#### 4.2.4.4 *Abbildung der Prozessmeilensteine*

*1:1-Abbildung von Meilensteinen*

Die bloße Einhaltung der Prozessregeln seitens der Simulation produziert noch keine Prozessausführungsspuren, welche auf die Erreichung der als Meilensteine (`milestone`) definierten Prozessziele orientiert sind. Aus diesem Grund ist auch die Transformation der DPIL-Meilensteine in eine Alloy-Repräsentation notwendig. Diese ist in Codeausschnitt 4.5 und Tabelle 4.4 dargestellt. Ersteres beschreibt die Grundstruktur für die Abbildung der Meilensteine in Alloy. Der Fakt wird unabhängig von der Anzahl der im DPIL-Modell enthaltenen Meilensteine nur einmal in der Alloy-Spezifikation materialisiert und besteht lediglich aus statischen Inhalten. Die erste Zeile des Fakts legt fest, dass die Meilensteine in jeder simulierten Prozessinstanz erreicht werden müssen. In der zweiten Zeile wird festgelegt, dass eine Prozessinstanz endet, sobald alle Meilensteine erreicht sind. Das zugehörige Prädikat `milestones[es: set PEvent]` sammelt alle transformierten Meilensteine aus dem DPIL-Modell. Für jeden wird im Prädikat eine eigene Zeile nach dem Schema `some e: es|$msbody$` erzeugt. Mit `some e: es` wird sichergestellt, dass mindestens ein Ereignis der abgeschlossenen Prozessausführungsspur den jeweiligen Meilenstein erfüllt. Letzterer wird durch die Ersetzung des Platzhalters `$msbody$` beschrieben. Im derzeitigen Konzept kann `$msbody$` durch Makroverwendungen (Tabelle 4.3) oder Beschreibungen des Auftretens bestimmter Ereignisse (Tabelle 4.4) ersetzt werden. Auch die Verknüpfung mehrerer der genannten Ersetzungen mittels Disjunktion („Veroderung") ist möglich. Damit können sowohl atomare Meilensteine, wie beispielsweise `milestone complete(C)`, aber auch komplexe Meilensteine, wie `milestone complete(C) or complete(D)`, transformiert werden. Jede einzelne Zeile des `milestones`-Prädikats ist mit den übrigen durch Konjunktion verknüpft, was sicherstellt, dass jede Prozessausführungsspur *alle* Meilensteine erfüllen muss.

*Teildynamisierung der Makroabbildung*

*Exemplarische Schablonen für Prozessziele*

Die in Tabelle 4.4 beschriebenen Inhalte von Meilensteinen können derzeit das Auftreten eines Ereignisses sein, welches die Vollendung einer Aufgabe signalisiert. Zudem kann auch das Ereignis der Erzeugung eines bestimmten Datenobjektes, beispielsweise einer Messung oder eines bestimmten Dokuments als Prozessziel definiert werden. Die letzte Abbildung beschreibt diesbezüglich eine differenziertere Form, bei der nicht nur die Existenz eines Wertes für eine Variable von Bedeutung ist, sondern auch der Wert selbst. Diese Auflistung möglicher Inhalte für DPIL-Meilensteine ist ebensowenig vollständig wie die diesbezüglichen

| DPIL-Meilensteininhalte (msbody) | Alloy |
|---|---|
| `complete(of $a$)` | `#search[$a.name$]>0` |
| `write(of $o$)` | `#search[$o.name$]>0` |
| `variablewrite(of $o$ value $v$)` | `let f = search[$o.name$]\|#f>0 and f.value[e]=$v$` |

Tabelle 4.4: Abbildung von DPIL-Meilensteinen auf Alloy-Sprachmittel

Ausdrucksmöglichkeiten in Alloy. Die dient erneut nur der Veranschaulichung der Möglichkeiten und ist, nach Erfahrungen in unterschiedlichen Projekten mit Partnern aus Industrie und Wirtschaft, in den meisten Fällen ausreichend.

Ziel von Abschnitt 4.2.4 ist die Konzeption einer Abbildungsvorschrift von DPIL-Modellen auf eine gegenüber Alloy konforme Spezifikation. Dies wird mit generischen strukturellen DPIL-Mustern ermöglicht, welche an ebenso generische Schablonen für Alloy-Konstruktionen gekoppelt sind. Eine zentrale Einschränkung des aktuellen Abbildungsprinzips ist, dass DPIL-Regeln nur dann transformiert werden können, wenn sie dem Aufruf eines Makros entspricht und dieses Teil einer beschriebenen Bibliothek von Modellierungsmustern ist. Durch die Prädikatenlogik als gemeinsames Fundament für DPIL und Alloy ist ein generisches Abbildungskonzept bei zukünftigen Weiterentwicklungen erstrebenswert und realistisch.

### 4.2.5 *Deterministische Konfiguration und Ausführung*

Nachdem der dynamische Teil des Simulationsmodells durch eine Abbildung des DPIL-Eingabemodells auf eine gültige Alloy-Spezifikation definiert ist, können darauf basierend nun – ebenfalls mittels Alloy – künstliche Prozessausführungsspuren erzeugt werden. Alloy generiert *alle* Lösungen für eine gegebene Begrenzung des Lösungsraums (Abschnitt 4.2.1). Damit ist die Spurgenerierung insofern deterministisch, als bei gleichem DPIL-Eingabemodell und gleicher Begrenzung des Lösungsraums auch die gleichen Prozessausführungsspuren generiert werden. Der aktuelle Abschnitt beschäftigt sich mit den Konfigurationsmöglichkeiten der Simulationsdurchführung. Nicht-deterministisches Verhalten kann ebenfalls simuliert werden, was im Folgeabschnitt gesondert betrachtet wird.

Nachfolgend werden nacheinander die Möglichkeiten zur Konfiguration der Simulation beschrieben. Diese sind zum Teil durch die in Abschnitt 4.1 vorgestellten verwandten Ansätze motiviert (auch Tabelle 4.1). Die hier besprochenen Konfigurationsmöglichkeiten sind im Einzelnen:

*Deterministische Konfigurationsmöglichkeiten*

- Wahl des Simulationsmodus *(O)*,
- (Maximale) Länge der Prozessausführungsspuren *(O)*,
- Maximale Anzahl der Prozessausführungsspuren *(O)*,
- Konfiguration von Entscheidungen (deterministisch),
- Konfiguration des Ressourcen-Bias,
- Differenzierte Konfiguration künstlichen Rauschens.

Die mit *(O)* markierten Anforderungen sind obligatorisch. Die ersten beiden Konfigurationen sind notwendig, um den Lösungsraum einzuschränken, was von Alloy gefordert wird. Der Simulationsmodus unterscheidet zwischen der Generierung von Positiv- und Gegenbeispielen für respektive gegen eine gegebene Alloy-Spezifikation.

*Wahl des Simulationsmodus*

Die Wahl des gewünschten Simulationsmodus erfolgt in zwei Schritten:

1. Transformation der im Simulationsmodell enthaltenen Prozessregeln und
2. Wahl des passenden Alloy-Kommandos.

P5

Gemäß der im vorherigen Abschnitt beschriebenen Transformation werden jede DPIL-Regel des zu simulierenden Prozessmodells auf jeweils ein Alloy-Fakt abgebildet. Dies entspricht bereits dem ersten Teil der Konfiguration für die Generierung von Positivbeispielen. Im zweiten Schritt muss zur Ausführung durch Alloy ein leeres Prädikat erzeugt und ein `run`-Kommando definiert werden, welches dieses „ausführt". Sollen stattdessen Gegenbeispiele generiert werden, muss im ersten Schritt der Inhalt aller Prozessregel-Fakten zeilenweise in einen einzelnen `assert`-Block verschoben werden. Jede Zeile der Behauptung ist so per Konjunktion mit allen anderen verknüpft. Als zugehöriges Kommando wird im zweiten Schritt `check` gewählt. Im Falle der Positivbeispiele ermittelt Alloy so Instanzen, die allen Fakten entsprechen, also das Modell erfüllen. Im Falle der Gegenbeispiele überprüft Alloy die Behauptungen und produziert Gegenbeispiele, sollten eine oder mehrere Behauptung falsch sein. Der Simulationsmodus erlaubt in dieser Form lediglich eine *globale* Entscheidung, ob Positiv- oder Gegenbeispiele für Prozessausführungsspuren generiert werden sollen. Erzeugt man in beiden Modi alle möglichen verschiedenen Prozessausführungsspuren aus dem begrenzten Lösungsraum, dann entspricht das Ergebnis der Simulation eines Prozessmodells, in dem keinerlei Regeln enthalten sind. In vielen Fällen ist eine differenziertere Unterscheidung wünschenswert. Diese Möglichkeit wird im letzten Teil des aktuellen Abschnitts beschrieben.

```
fact{ /* Prozessregel 1 */ }
fact{ /* Prozessregel 2 */ }
pred positives{}
run positives for (exactly) p PEvent, (exactly) i Identity, (exactly) d
    DataObject, p' Int

assert negatives{
    /* Prozessregel 1 */
    /* Prozessregel 2 */
}
check negatives for (exactly) p PEvent, (exactly) i Identity, (exactly) d
    DataObject, p' Int
```

Codeausschnitt 4.6: Konfiguration der Länge der Prozessausführungsspuren

P2

*Länge der Prozessausführungsspuren*

Die Länge der Prozessausführungsspuren kann entweder *exakt* oder durch eine *obere Schranke* begrenzt werden. Als Länge einer Prozessausführungsspur wird die Anzahl der protokollierten Ereignisse einer Prozessinstanz verstanden. Auf das Alloy-basierte Simulationsmodell abgebildet ergeben sich hier die in Codeausschnitt 4.6 dargestellten Konfigurationsschablonen.

Wie oben beschrieben, sind die beiden Kommandoschablonen für die Generierung von Positiv- (`run`) respektive Gegenbeispielen (`check`) zu verwenden. Dabei ist `positives` ein leeres Prädikat. Mit `negatives` ist eine Behauptung, also ein `assert`-Block gemeint, welcher alle Prozessregeln mittels Konjunktionen verknüpft. Nach dem Schlüsselwort `for` erfolgt die Scope-Definition, also die Einschränkung des Lösungsraums. Dieser wird für jede Signatur, die nicht bereits eindeutig quantifiziert ist einzeln festgelegt. Ein Beispiel für eine bereits quantifizierte Signatur ist die `Task`-Signatur, die mittels des Schlüsselworts `one` auf exakt ein Vorkommen pro Prozessausführungsspur festgelegt ist. Die in Codeausschnitt 4.6 in den Kommandos aufgelisteten Signaturen können wie folgt quantifiziert werden:

*Quantifizierung der Signaturen*

P6 P7

- `PEvent`: Der Zahlenwert für `p` entspricht der gewünschten maximalen oder exakten Länge der Ausführungsspuren,
- `Identity`: Die Anzahl `i` der für den Prozess zu erzeugenden Identitäten kann beliebig gewählt werden. Als Standardwert wird die Anzahl der im DPIL-Modell deklarierten Identitäten gesetzt.
- `DataObject`: Die Anzahl `d` der Datenobjekte, die in einer Prozessinstanz verwendet werden, kann ebenfalls beliebig gewählt werden. Als Standardwert wird die Gesamtzahl der im Prozessmodell deklarierten Datenobjekte, also die Anzahl aller Variablen und Dokumente gesetzt.
- `Int`: Der Wertebereich `p'` ganzer Zahlen wird automatisch auf Basis der gewünschten Länge der Prozessausführungsspuren sowie der zu simulierenden Datenwerte ermittelt.

P3

Zu jeder im Kommando quantifizierten Signatur kann zusätzlich das Schlüsselwort `exactly` angegeben werden, wodurch die Lösungen nur Beispiele oder Gegenbeispiele von *exakt* der angegebenen Größe beinhalten. Die einzige Ausnahme bildet hier die in Alloy standardmäßig verfügbare Signatur `Int`. Diese wird für die Positionsangabe der Ereignisse in den Prozessausführungsspuren sowie für die Festlegung des Wertebereichs $]-\mathrm{D}, \mathrm{D}]$ benötigt, da sie die Menge an Ganzzahl-Atomen repräsentiert. Durch diese Abhängigkeiten wird die obere Begrenzung `p'` für die `Int`-Signatur nach folgender Vorschrift ermittelt:

$$\mathrm{p}' = \max\left(\lceil \mathrm{ld}\, \mathrm{p} \rceil, \lceil \mathrm{ld}\, 2\mathrm{D} \rceil\right) \tag{25}$$

Grund für die Verwendung des *logarithmus dualis* ist, dass der Scope der `Int`-Signatur als *Bittiefe* angegeben wird. Das bedeutet, dass mit einer Scope-Einschränkung von `p'` ganze Zahlen im Wertebereich $]-2^{\mathrm{p}'}/2, 2^{\mathrm{p}'}/2]$ erzeugt werden. Zusätzlich ist definiert, dass, falls für keine Variable ein ganzzahliger Datentyp benötigt wird oder gar keine Variablen modelliert sind $\mathrm{D} \stackrel{\text{def}}{=} 1$, wodurch $\mathrm{p}' = \mathrm{ld}\, 1 = 0$ gilt.

P4

*Maximale Anzahl an Prozessausführungsspuren*

Die Konfiguration der gewünschten *maximalen Anzahl* an Prozessausführungsspuren wird auf die Selektion der entsprechenden Anzahl an Lösungen für das Erfüllbarkeitsproblem abgebildet. Diese entsprechen Wertebelegungen für das relationale Eingabemodell. Der Konfigurationsparameter ist als obere Schranke modelliert, da so nicht vorausgesetzt werden muss, dass der Nutzer die maximale Anzahl unterschiedlicher Prozessausführungsspuren kennt.

*Konfiguration von Entscheidungen*

Falls die Simulation nicht, wie standardmäßig vorgesehen, *beliebige* Prozessinstanzen simulieren soll, können zusätzlich alle im Prozessverlauf zu treffenden *Entscheidungen* in Form von zusätzlichen Restriktionen eingeschränkt werden. Für imperative Prozessmodelle bedeutet die Konfiguration von Entscheidungen die Selektion von Teilpfaden an explizit definierten Entscheidungspunkten. Da deklarative Prozessmodelle mögliche Pfade durch den Prozess nicht explizit modellieren, sind auch die Entscheidungspunkte nicht explizit kodiert. Entscheidungen können hier nur in Bezug auf einzelne Ereignisse getroffen werden. Daher bedeutet die Konfiguration von Entscheidungen für die hier beschriebene Technik die Zuweisungen der einzelnen Entitäten aus den unterschiedlichen Prozessperspektiven zu Ereignissen. Hierbei ist es gleichermaßen möglich, den Start (engl. *head*), das Ende (engl. *tail*) oder eine Zwischensequenz der simulierten Instanz zu beschränken. Oder es werden Aktivität-Ressource-Zuweisungen und Datenwerte priorisiert. Diese Priorisierungen können beispielsweise auf historischen Daten oder aktuellen Zuständen einer Prozessausführung basieren. Im aktuellen Abschnitt wird der deterministische Konfigurationsanteil betrachtet. Der im Rahmen dieser Arbeit entwickelte Simulationsansatz ist mit deterministischen Mitteln in der Lage, an *beliebiger* Stelle innerhalb der Prozessausführungsspur statische Anteile und andere Restriktionen in beliebiger Kombination einzufügen. Der Vorteil deterministischer Restriktionen ist die Kontrolle über die Inhalte der Prozessausführungsspuren und die Wiederholbarkeit. Der Satz möglicher Restriktionen ist mit Alloy-Sprachmitteln formuliert und kann daher beliebig erweitert werden. Für den hier diskutierten Ansatz werden die folgenden gängigen Einschränkungstypen berücksichtigt:

*Zusätzliche Einschränkungen*

1. Statische Aktivitäten, z.B. „Die ersten drei Aktivitäten sind A, B und C".
2. Statische Ressourcenzuweisungen, z.B. „Die letzten Akteure sind I und J".
3. Statische Datenbelegungen, z.B. „T ist anfangs 10 und nie über 100".
4. Ignorieren von Aktivitäten oder Ressourcen, z.B. „I soll nie A ausführen".
5. Instanziierung aller Modellelemente, z.B. „Jede Ressource soll mindestens eine Aufgabe bearbeiten".
6. Bedingte Restriktionen, z.B. „Wenn T > 50, muss I Aufgabe B ausführen".

Die ersten drei Typen sind für die Herstellung der Konfigurierbarkeit des Startzustandes der Spurgenerierung notwendig (Tabelle 4.1 in Abschnitt 4.1). Alle übrigen sind Beispiele, welche die Flexibilität des Alloy-basierten Ansatzes hervorheben und eine detailliertere Konfiguration des Spurgenerators gestatten.

*Generische Prädikate*

P11

In Codeausschnitt 4.7 sind einerseits die genannten Restriktionstypen und andererseits die dafür angegebenen Beispiele beschrieben. Die einzelnen Restriktionstypen können mittels weniger, generischer Prädikate umgesetzt werden. Das Prädikat `static` ist gültig, wenn an der angegebenen Position in der Prozessausführungsspur ein Ereignis auftritt, welches von der angegebenen Ressource (`i`) ausgelöst wurde, die festgelegte Aktivität (`t`) betrifft und in welchem die Produktion einer bestimmten Menge an Daten (`d`) protokolliert ist. Hierbei sind alle Parameter optional. Möchte man einen Parameter nicht angeben, dann kann dafür,

wie beispielsweise in Zeile 22 dargestellt ist, der Name der betreffenden Signatur eingesetzt werden. Dies referenziert *alle* Atome der zugehörigen Relation und sagt aus, dass *jedes* davon eine gültige Belegung ist. Die einzige Ausnahme bildet der Parameter `d`, für den stattdessen die leere Menge (`none`) angegeben werden muss.

```
pred static[t:Task, idx:Int, i:Identity, d:set DataObject]{
    ∀ te:TaskEvent | te.pos = idx → (t + i + d) in te.assoEl
}
pred ignore[t: Task, idx: Int, i: Identity, d: set DataObject]{
    no te:TaskEvent | te.pos = idx → (t + i + d) in te.assoEl
}
pred restricted[v:Variable, l:lone Int, u:lone Int, e:set TaskEvent]{
    ∀ te:e | let vl = int[v.value[te]] | vl = none or vl ⩾ int[l] and vl ⩽ int
        [u]
}
pred useAll[ae: AssociatedElement]{
    ae in TaskEvent.assoEl
}

fun eventAtPos(idx: Int): one PEvent {
    { e:PEvent | e.pos = idx }
}
fun lastEvent(): one PEvent {
    { e: PEvent | ∀ other: (PEvent - e) | other.pos < e.pos }
}

fact{
    static[A,int[integer/min],Identity,none]
    static[B,int[integer/min + 1],Identity,none]
    static[C,int[integer/min + 2],Identity,none]
    static[Task,int[lastEvent[].pos],J,none]
    static[Task,int[lastEvent[].pos - 1],I,none]
    restricted[T,10,10,eventAtPos[integer/min]]
    restricted[T,100,100,PEvent]
    ignore[A,Int,I,none]
    useAll[Identity]
    !restricted[T,int[integer/min],50,PEvent] → static[B,Int,I,none]
}
```

Codeausschnitt 4.7: Zusätzliche Restriktionen für die Simulationsdurchführung

Das Prädikat `ignore` ist analog aufgebaut, seine Gültigkeit ist allerdings genau entgegengesetzt. Somit kann mittels dieses Prädikats das Auftreten bestimmter Entitäten im gewünschten Kontext verboten werden. In Zeile 26 ist das zugehörige, oben beschriebene Beispiel dargestellt.

Für statische Datenbelegungen für einen gewünschten Kontext ist das Prädikat `restricted` zu verwenden. Dieses ist gültig, wenn der Wert für die angegebene Variable (`v`) innerhalb einer ebenfalls angegeben Ereignismenge (`e`) im Wertebereich [`l`,`u`] liegt. Hierbei ist erneut jeder Parameter optional, sodass einerseits lokale Wertebereichsbeschränkungen (Zeile 27) und andererseits globale (Zeile 28) vorgenommen werden können. P12

Das letzte Prädikat (`useAll`) ist gültig, wenn alle angegebenen Entitäten mindestens einmal in einem beliebigen Ereignis auftreten. Das Prädikat kann somit für Ressourcen (Zeile 30) und beispielsweise auch für Aufgaben verwendet werden. Schließlich ist es zudem möglich, jedes der Prädikate zur Formulierung einer *bedingten* Restriktion einzusetzen, wie es beispielsweise in Zeile 31 am oben beschriebenen Beispiel dargestellt ist. Dazu ist es lediglich notwendig, zwei Prädikat-„Aufrufe“ mittels logischer Implikation zu verknüpfen. Das von der Impli- P10

kation links stehende Prädikat dient dabei als Aktivierungsbedingung. Das andere ist die Konsequenz, die sich bei erfolgter Aktivierung ergibt.

Dem Prädikat `restricted` kommt für das Standardverhalten des Simulators eine zentrale Rolle zu. Variablen können mit Werten aus einem potentiell unendlich großen Wertebereich belegt werden. Für jede Belegung aus diesem Wertebereich würde – nach der Definition der Redundanzfreiheit von Ereignisprotokollen (Definition 3.2) – pro möglichem Datenwert eine gültige Prozessausführungsspur erzeugt werden. Damit beinhaltet das Ereignisprotokoll ebenfalls potentiell unendlich viele Spuren. Um dieses Problem zu umgehen, greift der Simulator auf eine *Heuristik* zurück, welche die Variablen-Wertebereiche dynamisch sehr stark einschränkt. Die Heuristik basiert auf der Annahme, dass ein großer Teil der möglichen Belegungen überflüssig ist. Zweck der Simulation von Wertezuweisungen ist es, beispielhaft zu beschreiben, welche Wertebereiche vom Modell zugelassen sind und welche nicht. Ein solcher Bereich kann mathematisch über Intervallgrenzen angegeben werden. Angenommen in einem bestimmten Prozesszustand gilt für die ganzzahlige[12] Variable $T$ die Regel $T < 10$, dann ist der erlaubte Wertebereich durch das Intervall $[-\infty, 9]$ definiert. Die Heuristik geht hier davon aus, dass für die Simulation lediglich die Intervallgrenzen interessant sind. Folglich würde für das Beispiel die Materialisierung der Datenwerte $-\infty$ und $9$ ausreichen. Allerdings könnte man anhand dieser zwei Materialisierungen nicht unterscheiden, ob sie für die Regel $T < 10$ oder $T = -\infty \vee T = 9$ erzeugt wurden. Aus diesem Grund empfiehlt es sich, nicht nur die Intervallgrenzen, sondern auch $n$ Folgewerte zu simulieren. Bei $n = 2$ wären die erzeugten Werte beispielsweise $-\infty$, $8$ und $9$.

*Begrenzung der Wertebereiche durch Heuristik*

Bisher sind die Intervallgrenzen nur mit diesbezüglich *gültige* Werten beschrieben. Da die eben erwähnten Regeln jedoch das Prozessverhalten beeinflussen können, müssen auch bezüglich des Intervalls *ungültige* Werte materialisiert werden. Für das Beispiel $T < 10$ würden somit – inklusive von $n = 2$[13] Folgewerten – mindestens die Werte $-\infty$, $8$ bis $11$ erzeugt werden.

Im Allgemeinen werden durch den Simulator für die folgenden Regelformen Datenwerte nach der ebenfalls folgenden Heuristik materialisiert ($\rightrightarrows$):

- $T = x \rightrightarrows T = x$
- $\forall x \in \mathbb{Z} | T < x \rightrightarrows T \in \{-\infty, x-n, x-n+1, ..., x, x+1, ..., x+n\}$
- $\forall x \in \mathbb{Z} | T > x \rightrightarrows T \in \{x-n, x-n+1, ..., x, x+1, ..., x+n, +\infty\}$
- $\forall x \in \mathbb{Z} | T \leqslant x \equiv T < (x+1)$
- $\forall x \in \mathbb{Z} | T \geqslant x \equiv T > (x-1)$

Komposite Einschränkungen eines gültigen Wertebereichs werden in atomare Einschränkungen zerlegt. Beispielsweise wird die Regel $x_1 < T \leqslant x_2$ in die atomaren Regelbestandteile $x_1 < T$ und $T \leqslant x_2$ zerlegt, wobei $x_1 < T \equiv T > x_1$ gilt. Zusätzlich wird festgelegt: Alle Beschränkungen, welche *dieselbe* Variable betreffen, werden zur Ermittlung der zu materialisierenden Werte mittels Disjunktion verknüpft. Die Materialisierung der so erlaubten Datenwerte erfolgt mittels des in Codeausschnitt 4.7 eingeführten Prädikats `restricted`.

*Komposite Einschränkungen*

12 Datenwerte sind in Alloy immer ganzzahlig.

13 $n = 2$ ist die Standardeinstellung des Simulators.

```
task A
task B
task C

variable T

process ABC{
    start(of B) implies variablewrite(of T value = 20)
    start(of C) implies variablewrite(of T value > 5 and value < 9)
}
```

Codeausschnitt 4.8: DPIL-Beispiel mit verteilten Wertebereichsbeschränkungen für Daten

```
/*Prozessregeln, Aktivitäten und Variable T*/
fact{
    restricted[T,integer/min,integer/max,PEvent] or
    restricted[T,integer/min,integer/max,PEvent] or
    restricted[T,3,7,PEvent] or
    restricted[T,7,11,PEvent]
}
```

Codeausschnitt 4.9: Beispiel für heuristische Materialisierung von Datenwerten

Zur Illustration der Wirkungsweise der Heuristik sei von dem in Codeausschnitt 4.8 dargestellten Beispiel ausgegangen. Die Zerlegung der Beschränkungen des Wertebereichs für $T$ und ihre darauf folgende resultierende Boolesche Verknüpfung ergibt $T > 5 \vee T < 9 \vee T = 10$. Mit $n = 2$ Folgewerten sind die aus dem Beispiel abzuleitenden materialisierten Datenwerte nach folgender Vorschrift zu ermitteln: $T \in \{3,4,5,6,7,\infty\} \vee T \in \{-\infty,7,8,9,10,11\} \vee T = 10$. Die Transformation der Vorschrift nach Alloy ist in Codeausschnitt 4.9 dargestellt. Die zwei ersten Beschränkungen des Wertebereichs für `T` repräsentieren die beiden Zuweisungen $T = -\infty$ respektive $T = \infty$. Da Alloy Instanzen aus einem explizit durch den Scope begrenzten Lösungsraum ermittelt, ist die Menge der ganzen Zahlen in jedem Fall endlich. Aus diesem Grund werden diese beiden Schranken mit der jeweils kleinst- respektive größtmöglichen Zahl angenähert. Welche Zahl das ist, hängt von der Festlegung des `Int`-Scopes $p'$ ab. Dieser wird gemäß Gleichung 25 berechnet, wobei $D = \max(|x|) + c, x \in R$ gilt und $R$ die Menge der zerlegten Wertebereichsbeschränkungen und $x$ die darin enthaltenen Zahlenwerte bezeichnet. Damit auch für die Näherungen von $-\infty$ und $\infty$ ein Zahlenwert materialisiert werden kann, muss der Wertebereich noch um eine Konstanze $c$ mit mindestens $c > 1$ erweitert werden. Für das in Codeausschnitt 4.8 beschriebene DPIL-Beispiel wäre die Scope-Konfiguration für `Int` mit beispielsweise $c = 10$ demzufolge $p' = \max(\lceil \mathrm{ld}\ 3 \rceil, \lceil \mathrm{ld}\, 2(11+10) \rceil) = \max(2,6) = 6$. Mit dem so definierten Scope können ganzzahlige Werte aus dem Bereich $]-32,32]$ erzeugt werden. Diese komplexe Restriktion von zu simulierenden Datenwerten schließt den Konfigurationsteil bezüglich gewünschter Inhalte ab.

*Annäherung der Schranken*

Unter den verwandten Arbeiten (Abschnitt 4.1) befindet sich ein Ansatz, welcher die Simulationsdurchführung zusätzlich durch Charakteristika der simulierten organisatorischen Ressourcen modifiziert. Konkret werden hierfür ein „fleißiger“ und ein „minimalistischer“ Prozessausführender definiert. Ersterer durchläuft den

*Ressourcen mit unterschiedlichen Charakteristika*

```
pred once{
  ∀ t:Task | #(PEvent.assoEl ∩ t) = 1
}
```

Codeausschnitt 4.10: Prädikat für einmaliges Auftreten einer Aktivität

Prozess, indem er alle bisher nicht ausgeführten Aufgaben, die ausgeführt werden können, ausführt. Letzterer führt nur die für das Erreichen des Prozessziels *unumgänglich* notwendigen Aufgaben aus. Eine explizite Möglichkeit, letzteres in Alloy zu realisieren gibt es nicht, da Alloy keine instanzübergreifenden Fakten unterstützt. Ein Weg, dieses Problem zu umgehen, ist die schrittweise Erweiterung des `PEvent`-Scopes für das jeweilige Ausführungskommando. Nacheinander kann dann in Brute-Force-Manier, beginnend bei 1, der Scope schrittweise vergrößert werden, bis der Alloy Analyzer eine Lösung ermittelt. Beschränkt man sich auf DPIL-Modelle ohne die Datenorientierte Perspektive, dann ist es möglich, diesen minimalen Scope direkt zu berechnen [131]. Die Imitation eines fleißigen Akteurs kann dagegen mittels eines weiteren Prädikats umgesetzt werden, das in Codeausschnitt 4.10 dargestellt ist. Kritisch ist hier anzumerken, dass das zugrundeliegende Prozessmodell die Ausführung zweier bestimmter Aktivitäten innerhalb derselben Instanz verbieten kann. In diesem Fall existiert keine Lösung. Ebenso ist es möglich, dass das Prozessmodell die mehrfache Ausführung eine Aktivität fordert. Auch dafür würde Alloys Analyse-System keine Lösung ermitteln können.

*Mischen von validen und nicht-validen Spuren*

Bisher könnten die beschriebenen Konfigurationen lediglich global vorgenommen werden, gälten also für alle generierten Prozessausführungsspuren. Möchte man ein Ereignisprotokoll mit einer Mischung aus validen und nicht-validen Prozessausführungsspuren erzeugen, dann kann dies mittels mehrfacher Simulationsdurchführung und der gewünschten Veränderung der Konfiguration für jede Durchführung realisiert werden. Die Kombination aller so entstehenden Prozessausführungsspuren ergibt das gewünschte gemischte Ereignisprotokoll. Dabei ist es nicht mehr garantiert, dass die Menge der Prozessausführungsspuren frei von Redundanzen ist.

*Differenzierte Konfiguration künstlichen Rauschens*

Alle vorab beschriebenen Restriktionstypen können unabhängig vom Simulationsmodus eingesetzt werden. Sie können einerseits für die Einschränkung der zu generierenden Positivbeispiele für das gegebene Prozessmodell verwendet werden. Andererseits können sie die zu generierenden Gegenbeispiele, also Ausführungsspuren, die mindestens eine Prozessregel verletzten, beschreiben. Dies ermöglicht die *differenzierte Konfiguration künstlichen Rauschens*, also die Beschränkung der Generierungergebnisse auf nicht-valide Ausführungsspuren mit bestimmten Eigenschaften. Eine beispielhafte, gezielte Konfiguration künstlichen Rauschens ist in Codeausschnitt 4.11 beschrieben (Prädikat `noise` und Behauptung `negatives`). In der hier angegebenen Verwendung werden lediglich Gegenbeispiele gegen das gegebene Prozessmodell generiert, die einerseits mehrere Prozessregeln verletzten (Zeilen 19 bis 23) und andererseits zusätzliches Rauschen enthalten (Zeile 24). Letzteres setzt sich unter anderem aus Fällen zusammen, in denen das dritte Ereignis nicht Aktivität `C` betrifft, die Variable `T` nur mit dem Wert 100 belegt werden kann und/oder mindestens eine Ressource in der simulierten Instanz inaktiv bleibt,

```
...
sig NoisyTask extends Task{}

pred noisyAddTasks[min,max: Int]{
  #NoisyTask ⩽ max and #NoisyTask ⩾ min
}
pred noisySkipTasks[min,max: Int]{
  let n = #(TaskEvent.assoEl ∩ Task) | n ⩾ min and n ⩽ max
}
pred noise{
  static[C,int[integer/min + 2],Identity,none]
  !restricted[T,100,100,PEvent]
  useAll[Identity]
  !noisyAddTasks[0,n]
  !noisySkipTasks[1,1]
}

assert negatives{
  /* Prozessregel 1 */
  /* Prozessregel 2 */
  sequence[A,C] /* Prozessregel 3 */
  direct[A,I] /* Prozessregel 4 */
  role[B,R] /* Prozessregel 5 */
  noise[]
}

check negatives for (exactly) p PEvent, (exactly) i Identity, (exactly) d
    DataObject, n NoisyTask, p' Int
```

Codeausschnitt 4.11: Differenzierte Rauschkonfiguration

also keine Aktivität durchführt (erste drei Zeilen in `noise`). Hierfür werden Prädikate für die deterministische Simulationskonfiguration wiederverwendet. Die hier angegebenen Restriktionen sind insofern Rauschen, als sie die Menge der möglichen Prozessausführungsspuren stärker einschränken als vom Modell vorgegeben. Bestimmte mögliche Verläufe des Prozesses werden explizit ausgeschlossen.

*Expliziter Ausschluss bestimmter Prozessverläufe*

Das Prädikat zur Rauschkonfiguration enthält weitere Restriktionen, die mittels Prädikatschachtelung realisiert sind. Das `noisyAddTasks`-Prädikat stellt sicher, dass alle Lösungen eine der `min`- und `max`-Parameter entsprechende Anzahl von `NoisyTask`-Atomen enthält. Die zugehörige, nicht näher quantifizierte Signatur der Atome ermöglicht die flexible Generierung einer Anzahl an zusätzlichen Aktivitäten, die nicht Teil des Prozessmodells sind. Mit `noisySkipTasks` kann eine durch `min` und `max` begrenzte Anzahl an Aktivitäten von der Simulation ausgeschlossen werden. Ob im Analyseschritt eine Lösung für diese Konfiguration gefunden wird, ist dabei vom Prozessmodell abhängig. Übersteigt die Anzahl ausgeblendeter Aktivitäten die Anzahl für den Prozess optionaler Aktivitäten, dann existiert kein Gegenbeispiel.

Die in `negatives` enthaltenen logischen Aussagen beschreiben zwei beliebige und drei spezifische Prozessregeln, für welche entsprechende Gegenbeispiele ermittelt werden sollen. Mit der als `Prozessregel 3` bezeichneten Regel können explizit Verstöße gegen eine Reihenfolgevorschrift für die Aktivitäten `A` und `B` angefragt werden. Die beiden nachfolgenden Regeln bewirken explizite Verstöße hinsichtlich der korrekten Assoziation von organisatorischen Ressourcen mit den Aufgaben `A` und `B`. Anzumerken ist hierbei, dass für die Rauschkonfiguration drei

der in Abschnitt 4.2.4 beschriebenen Alloy-Repräsentationen für DPIL-Makros wiederverwendet werden. Das bedeutet, dass für derartige Rauschkonfigurationen diese Makros definiert sein müssen.

Das in der letzten Zeile zur Ausführung verwendete `check`-Kommando wird bereits in der Beschreibung von Codeausschnitt 4.6 detailliert erläutert. Da aber mit `NoisyTask` eine neue Signatur definiert wird, muss auch dafür ein Scope angegeben werden. Dieser ist jedoch identisch mit der im Prädikat `noisyAddTasks` angegebenen oberen Schranke für zusätzliche, nicht im Modell enthaltene Aktivitäten.

Zwar ist die Ausdrucksstärke von Alloy mit diesen Beispielen für die Konfiguration zur Erzeugung von Positiv- und Gegenbeispielen keineswegs erschöpft, wohl aber die von vergleichbaren Ansätzen unterstützten Einschränkungsmöglichkeiten für Simulationsdurchführungen. Auch aus diesem Grund fehlen weitere, konkrete Anforderungen für derartige Konfigurationsmöglichkeiten, sodass an dieser Stelle auf weitere Beispiele für deterministische Simulationseinschränkungen verzichtet wird.

### 4.2.6 *Nicht-Deterministische Konfiguration und Ausführung*

Nicht jede gewünschte Konfiguration der Simulation ist deterministisch. Unsicherheiten, wie beispielsweise die Priorisierung bestimmter Aktivitäten durch Gewohnheiten der Prozessbeteiligten, werden üblicherweise mittels Wahrscheinlichkeitsverteilungen approximiert und sind damit per Definition nicht deterministisch. Alloy selbst ermittelt für eine gegebene Spezifikation und eine konfigurierte Größe des Lösungsraums *redundanzfrei jede* Lösung. Damit ist ein auf Alloy basierter Simulationsansatz zunächst deterministisch. Dieser Abschnitt beschäftigt sich mit Erweiterungen der Konfigurationsmöglichkeiten, um dennoch nicht-deterministisches Verhalten unterstützen zu können. Der Kern ist dabei, die nicht-deterministischen Berechnungen außerhalb von Alloy vorzunehmen, deren Ergebnisse aber wiederum durch gezielten Einsatz der Alloy-Sprachmittel in den generierten Prozessausführungsspuren zu materialisieren.

*Nicht-deterministisches Verhalten*

Die bisher entwickelten Techniken zur Spurgenerierung unterstützen die folgenden Aspekte nicht-deterministischen Verhaltens (Tabelle 4.1):

- Konfigurierbare Entscheidungen (nicht-deterministisch),
- Nicht-deterministisches künstliches Rauschen und
- Eine konfigurierbare Prozessumgebung.

Entscheidungen zu konfigurieren bedeutet im aktuellen Kontext die Priorisierung der Zuweisungen von Aktivitäten, Ressourcen und anderer Entitäten zu Ereignissen (Abschnitt 4.2.5). Besonders durch die Analyse historischer Prozessaufzeichnungen ist es oft möglich, derartige Priorisierungen zu erkennen und mittels Wahrscheinlichkeitsverteilungen zu approximieren. Alloy selbst beinhaltet keine Mittel, um Wahrscheinlichkeitsverteilungen zu modellieren. Aus diesem Grund ist für den im Rahmen der vorliegenden Arbeit entwickelten Ansatz eine generelle Trennung der deterministischen und nicht-deterministischen Anteile vorgenommen

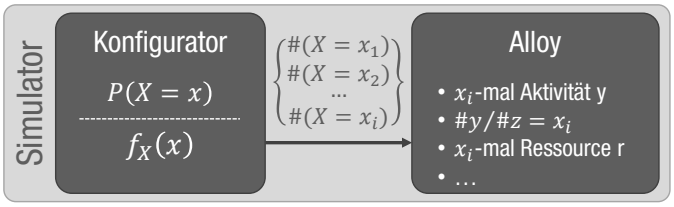



Abbildung 4.6: Abtrennung nicht-deterministischer Simulationskonfiguration

*Wahrscheinlichkeitsverteilungen via externem Konfigurator*

worden, die in Abbildung 4.6 skizziert ist. In einem externen *Konfigurator* ist es möglich, verschiedene diskrete oder kontinuierliche Wahrscheinlichkeitsverteilungen zu parametrieren, um ein in der Realität nicht deterministisches Verhalten zu modellieren (Abschnitt 3.2.3). Dazu wird eine Zufallsvariable $X$ und ihre möglichen Ausprägungen $x_1, x_2, ..., x_i$ bestimmt. Eine Zufallsvariable könnte beispielsweise Aktivitäten repräsentieren und die möglichen Ausprägungen sind die im Prozessmodell angegebenen Aktivitäten. Mittels der gewählten Wahrscheinlichkeitsverteilung wird ermittelt, wie häufig sich die jeweilige Ausprägung in einer Prozessausführungsspur manifestieren soll. Diese Häufigkeit wird mittels zusätzlicher Scope-Einschränkungen in Alloy realisiert und ist in Codeausschnitt 4.12 durch den ganzzahligen Parameter `x` bezeichnet.

```
fact{
  max[(A),x]
  min[(A + I),x]
  minRatio[(A),(B),x]
  maxRatio[(A + I),(B),x]
  minRatio[(A),(Task),x]
}

pred min[ae: set AssociatedElement, num: Int]{
  #match[ae] ⩾ num
}
pred max[ae: set AssociatedElement, num: Int]{
  #match[ae] ⩽ integer/max[num + 0]
}
pred minRatio[a,b: set AssociatedElement, r: Int]{
  let m = #(match[a])|let n = #(match[b])|(m > 0 and n = 0) or integer/div[m,n] ⩾
       integer/max[ratio,-1]
}
pred maxRatio[a,b: set AssociatedElement, ratio: Int]{
  let m = #(match[a])|let n = #(match[b])|n > 0 or integer/div[m,n] ⩽ integer/max[
       ratio,0]
}

fun match(ae: set AssociatedElement): set TaskEvent{
  { e: TaskEvent | (e.assoEl ∩ ae) = ae }
}
```

Codeausschnitt 4.12: Manifestation der Zufallsentscheidungen in Alloy

*Absolute vs. relative Häufigkeit*

Im hier beschriebenen Ansatz ist es möglich, bei der Konfiguration von Entscheidungen zwischen einer *absoluten* und einer *relativen Häufigkeit* zu unterscheiden. Mittels der Prädikate `min` und `max` kann jeweils die absolute Häufigkeit und mittels `minRatio` respektive `maxRatio` kann jeweils die relative Häufigkeit[14] des Auftretens bestimmter Assoziationen zwischen Entitäten und Ereignissen angege-

P8 P9

14 Es wird hierbei angenommen, dass für den übergebenen Parameter `x` >= 0 gilt.

ben werden. In beiden Fällen kann entweder das Auftreten *einzelner* Entitäten oder das Auftreten einer *Kombination* von Entitäten beschrieben werden. Beispiele dafür sind einerseits eine einzelne Aktivität und andererseits eine Aktivität und die ausführende Ressource. Diese Flexibilität wird durch die Funktion `match` geschaffen, die als Eingabe eine beliebige Menge an `AssociatedElement`en akzeptiert und nur die Ereignisse ermittelt, mit denen *alle* Elemente dieser Menge assoziiert sind.

Das Prädikat `min` ist gültig, wenn die Häufigkeit der Assoziation der gewählten Elemente mindestens so groß wie die gegebene Schranke $\mathfrak{num}$ ist oder $\mathfrak{num} < 0$ ist. Dagegen gilt `max`, wenn die Assoziationshäufigkeit höchstens $\mathfrak{num}$ ist. Ist $\mathfrak{num} < 0$, dann existiert keine Lösung, weswegen mit `integer/max[num + 0]` sichergestellt wird, dass der Vergleichswert für die Assoziationshäufigkeit mindestens 0 beträgt.

*Relative Häufigkeit*

Die Prädikate `minRatio` und `maxRatio` geben paarweise das Verhältnis der absoluten Assoziationshäufigkeiten einer Menge von Entitäten an – was letztlich einer relativen Häufigkeit entspricht. Die für den im Rahmen der vorliegenden Arbeit entwickelten Simulationsansatz verwendete Berechnungsvorschrift ist in Gleichung 26 dargestellt. Diese ist jedoch als beispielhaft anzusehen und kann nach Belieben ausgetauscht werden.

$$\mathrm{minRatio}[a,b,r] = \begin{cases} \mathrm{true} & |\mathrm{match}[a]| > 0 \wedge |\mathrm{match}[b]| = 0 \\ \mathrm{true} & |\mathrm{match}[a]| \,/\, |\mathrm{match}[b]| \geqslant \max(r,0) \\ \mathrm{false} & \mathrm{else} \end{cases} \tag{26}$$

$$\mathrm{maxRatio}[a,b,r] = \begin{cases} \mathrm{true} & |\mathrm{match}[b]| > 0 \wedge |\mathrm{match}[a]| \,/\, |\mathrm{match}[b]| \leqslant \max(r,0) \\ \mathrm{false} & \mathrm{else} \end{cases}$$

Alternativ könnte außerhalb von Alloy, im Konfigurator, für die gewünschte relative Häufigkeit zwei ihr entsprechende, ganzzahlige absolute Häufigkeiten für die gewünschten Mengen der Entitäten ermittelt werden. Das Häufigkeitsverhältnis könnte dadurch mittels der bereits beschriebenen Prädikate `min` und `max` realisiert werden. Dies würde die Prädikate `minRatio` und `maxRatio` überflüssig machen. Der wesentliche Nachteil dieser Lösung – und der Grund für die stattdessen vorgeschlagene – ist, dass die Simulation so auf genau eine Ausprägung des Häufigkeitsverhältnisses beschränkt ist. Die stattdessen vorgeschlagene erlaubt jede Ausprägung, welche dem konfigurierten Verhältnis entspricht.

*Erzeugung einer Zufallszahl*

In Codeausschnitt 4.12 und den nachfolgenden Erklärungen wird auf die *Verwendung* der Zufallszahl $\mathfrak{x}$ nicht aber auf die *Erzeugung* derselben eingegangen. Die gewählte und parametrierte Wahrscheinlichkeitsverteilung kann dafür auf zwei Arten zur Konfiguration der Entscheidungen verwendet werden: In der ersten Verwendungsform wird eine Zufallszahl für die Prädikate `min` und `max` mittels eines auf der gewählten Wahrscheinlichkeitsverteilung basierenden Pseudo-Zufallszahlengenerators (Abschnitt 3.2.3) ermittelt. In der zweiten Verwendungsform werden für `minRatio` respektive `maxRatio` jeweils zwei Zufallszahlen ermittelt. Anschließend wird das Verhältnis dieser beiden Zahlen gebildet und auf die nächste ganze Zahl abgerundet. Das Ergebnis ist der gesuchte Parameter für die Verwendung der Verhältnis-Prädikate.

Eine wichtige Einschränkung der hier beispielhaft angegebenen Prädikate für Häufigkeitsverhältnisse ist, dass letztere nur als ganzzahliges Vielfaches angegeben werden können. Dementsprechend kann beispielsweise nicht zwischen den Häufigkeitsverhältnissen $6/5$ und $8/5$ bezüglich zweier Aktivitäten unterschieden werden. Ursache hierfür ist, dass die im Alloy-Modul `integer` definierte Divisionsfunktion jedes Ergebnis auf die nächstmögliche ganze Zahl abrundet. Dies könnte jedoch durch die Definition des Wertebereichs der rationalen Zahlen (Abschnitt 4.2.9) und der Verwendung einer darauf ausgerichteten Divisionsfunktion modifiziert werden.

Bisher werden lediglich Zuweisung von nicht veränderlichen Objekten, wie Aufgaben oder organisatorischen Ressourcen betrachtet. Der Simulator muss jedoch Entscheidungen für das Zuweisen von Datenwerten zu Variablen treffen. Datenbelegungen können auch über das Prozessmodell hinaus zusätzlich statisch eingeschränkt werden (Abschnitt 4.2.5). Mittels der oben beschriebenen Zufallszahlen und darauf aufbauender zusätzlicher `restricted`-Prädikate können auch Datenwerte nicht-deterministisch festgelegt werden. Der einzige Unterschied stellt hier der durch Regeln festgelegte, erlaubte Datenbereich dar. Durch die Zerlegung aller komplexen Wertebereichseinschränkungen in atomare können jedoch trivial alle erlaubten Wertebereiche ermittelt werden, sodass hier auf diese Thematik nicht weiter eingegangen wird.

*Entscheidungen für Variablenbelegungen*

Ein weiterer Aspekt nicht-deterministischen Simulationsverhaltens ist das *zufällige* Auftreten von nicht-modellkonformem Verhalten in Prozessausführungsspuren. Im vorherigen Abschnitt wird bereits erläutert, wie eine Konfiguration für *gezieltes* Rauschen vorgenommen werden kann. Im verbliebenen Teil dieses Abschnitts wird beschrieben, wie *zufälliges* Rauschen verursacht werden kann.

*Zufälliges Rauschen*

Ein Aspekt des unterstützten nicht-deterministischen Rauschens ist die Missachtung einer zufälligen Anzahl von Prozessregeln. Dazu kann auf eine beliebige diskrete Wahrscheinlichkeitsverteilung zurückgegriffen werden, deren Wertebereich nach oben durch die Anzahl der im Prozessmodell enthaltenen Prozessregeln beschränkt ist. Mittels eines auf der Wahrscheinlichkeitsverteilung basierenden Zufallsgenerators wird wiederum eine Zufallszahl erzeugt. Während der Generierung des Simulationsmodells werden zufällig Prozessregeln, deren Anzahl exakt der Zufallszahl entspricht, in `assert`- anstatt `fact`-Blöcken in ihrer Alloy-Form generiert (Codeausschnitt 4.11). Dies ermöglicht vollkommen zufälliges Rauschen.

Mit dem im vorangegangenen Abschnitt beschriebenen deterministischen und dem eben knapp angesprochenen vollständig zufälligen Rauschen beschränken sich die Konfigurationsmöglichkeiten lediglich auf Extreme. In [34] werden zu diesem Zweck Rauschprofile vordefiniert, welche das völlig zufällige Rauschen auf einen kontrollierten Rahmen beschränken. Auch die hier vorgestellte Alloy-basierte Simulationstechnik unterstützt die Definition von derartigen Profilen. Dazu werden die bereits in Abschnitt 4.2.5 eingeführten Prädikate für die unterschiedlichen Formen des Rauschens in verschiedenen kompositen Prädikaten gekapselt. Die in der Literatur aufgetretenen Formen des Rauschens sind in Tabelle 4.1 aufgelistet. Die für diese Formen als Beispiele entwickelten Rauschprofile sind die in Codeausschnitt 4.13 als `behavioralNoise`, `functionalAndBehavioralNoise` und `slightMultiperspectiveNoise` bezeichneten Prädikate.

*Definition von Rauschprofilen*

P13

```
pred behavioralNoise[min,max: Int]{
  noisySkipTasks[relative[Task,min],relative[Task,max]]
  !sequence[A,B]
 .../* min bis max sequence-Regeln   */
}
pred functionalAndBehavioralNoise[min,max: Int]{
  behavioralNoise[min,max]
  noisyAddTasks[relative[Task,min],relative[Task,max]]
}
pred slightMultiperspectiveNoise[min,max: Int]{
  functionalAndBehavioralNoise[min,max]
  !direct[A,I]
 .../* min bis max direct-Regeln */
  !role[B,J]
 .../* min bis max role-Regeln   */
}

fun relative(ae: AssociatedElement, prop: Int): Int{
  integer/max[integer/div[integer/mul[#ae,prop],100]+1]
}

pred noise{
  !behavioralNoise[5,10]              // Rauschprofil 1
  !functionalAndBehavioralNoise[5,10] // Rauschprofil 2
  !slightMultiperspectiveNoise[1,5]   // Rauschprofil 3
}

assert negatives{
  noise[]
}

check negatives for...
```

Codeausschnitt 4.13: Realisierung von Rauschprofilen mittels Alloy

*Wirkungsweise der Rauschprofile*

Das Rauschprofil `behavioralNoise` sorgt – pro Prozessausführungsspur – standardmäßig für das Ignorieren von 5-10% der im Prozessmodell enthaltenen Aktivitäten sowie für das Verstoßen gegen den gleichen relativen Anteil an Reihenfolgevorschriften für Aktivitäten. Letzteres kann nur auf Basis von explizit im DPIL-Modell enthaltenen *sequence*-Regeln realisiert werden. Dabei ist anzumerken, dass diese Konfiguration zwar Reihenfolgeverstöße auf gewünschte Aktivitäten begrenzen, nicht aber garantieren. Grund ist, dass Reihenfolgeabhängigkeiten beispielsweise auch durch Konsument-Produzent-Abhängigkeiten bezüglich Daten entstehen können. Die so als Anforderung für die Generierung von Gegenbeispielen verwendeten Sequenzregeln können daher einen Widerspruch zu anderen Regeln darstellen. Dadurch kann letztlich kein Gegenbeispiel identifiziert werden, da das Modell weiterhin die Einhaltung der Reihenfolge fordert. Gleiches gilt auch für das zweite Rauschprofil mit dem Namen `functionalAndBehavioralNoise`, welches auf dem vorherigen aufbaut. Als Neuerung sorgt es auch für die Generierung von 5-10% zusätzlicher, unbekannter Aktivitäten. Das letzte Rauschprofil, `slightMultiperspectiveNoise` kombiniert wiederum das vorherige mit einer variablen Anzahl an Regeln für direkte (`direct`) und rollenbasierte ((`role`)) Zuweisungen von Ressourcen. Diese stammen wiederum aus dem Prozessmodell und werden für die kontrollierte Generierung von Verstößen gegen Regeln der Organisatorischen Perspektive verwendet. Die Anzahl der Verstöße gegen Regeln aller in diesem Profil enthaltenen Perspektiven beträgt 1 - 5%. Die Wahl dieser relativen

Häufigkeiten ist vollkommen willkürlich und soll lediglich als Nachweis dienen, dass derartige Rauschprofile mittels Alloy definiert werden können.

Die eben beschriebenen Prädikate erlauben eine zwar komplexere aber nach wie vor deterministische Konfiguration. Die zufallsbasierte Konfiguration künstlichen Rauschens – auch *ohne* Rauschprofile – kann analog der in Abbildung 4.6 beschriebenen zwei Schritte vorgenommen werden. Hierzu werden die fraglichen Parameter gemäß der gewünschten Wahrscheinlichkeitsverteilung zufällig belegt. Für Rauschprofile wird gefordert, dass die so generierten Parametrierungen jeweils im Wertebereich von $[0,100]$ liegen, da sie Prozentsätze repräsentieren.

Abschließend ist für jede der bisher beschriebenen Konfigurationsmöglichkeiten ein Nachteil zu identifizieren. Durch die nur implizit im Prozessmodell kodierten Abläufe ist es im Allgemeinen nur schwer möglich, zu erkennen, ob für eine gewählte Konfiguration Lösungen existieren. Auch ist es nicht möglich, zu ermitteln, welche der Konfigurationen oder Kombinationen derselben gegebenenfalls die Absenz einer Lösung verursachen. Aus diesem Grund wird eine iterative Entwicklung der Konfiguration empfohlen. Beispielsweise könnte zunächst ein großzügiger Scope, keinerlei Beschränkungen von Teilen der Prozessausführungsspuren und kein Rauschen eingestellt werden. In einem nächsten Schritt würde man, falls gewünscht, gezielte Beschränkungen der Spuren vornehmen. Das Prüfen, ob für eine gewählte Konfiguration Lösungen existieren, wird jedoch in der vorliegenden Arbeit nicht näher betrachtet. Grund ist, dass im Falle dessen, dass mit dieser restriktiven Variante kein Gegenbeispiel erzeugt werden kann, ein alternativer Weg existiert, welcher im nachfolgenden Abschnitt beschrieben wird.

### 4.2.7 *Nachverarbeitung*

Im letzten Schritt der Simulationsdurchführung – der *Nachverarbeitung* – werden die ermittelten Prozessausführungsspuren um zusätzliche Informationen ergänzt und Teile werden nach Vorgaben modifiziert, die sich entweder nicht oder nur unkomfortabel direkt in Alloy erfüllen lassen. Zusätzliche Informationen sind *Zeitstempel* für das Auftreten der protokollierten Ereignisse. Modifikationen beinhalten hier das künstliche Einfügen von *Rauschen*. Schließlich werden die Simulationsergebnisse, die in Form von Lösungen für das in Alloy formulierte Erfüllbarkeitsproblem vorliegen, in die Form eines *Ereignisprotokolls* überführt.

*Zeitstempel*

*Zusätzliches Rauschen*

*Finalisierung als Ereignisprotokolle*

Die mittels Alloy simulierten Beispiele für bezüglich eines Prozessmodells konformen Prozessinstanzen sind für eine gegebene maximale Länge *vollständig*, *eindeutig* und *korrekt*. Jedes dieser Attribute kann durch künstliches Rauschen oder nachträgliches Einfügen von Redundanzen auf Wunsch eliminiert werden. Künstliches Rauschen, welches im Rahmen der Nachverarbeitung erzeugt wird, betreffen vollkommen *wahlfreie* Modifikationen der Lösungen. Im Unterschied zu den in Abschnitt 4.2.5 und Abschnitt 4.2.6 beschriebenen Formen künstlichen Rauschens erfolgt das nachfolgend beschriebene Rauschen *nicht* auf Basis des Wissens über die Inhalte der Prozessausführungsspuren. Die Formen künstlichen Rauschens, welche durch den Nachverarbeitungsschritt realisiert werden, sind *(i)* das Überspringen von Aktivitäten, *(ii)* das Einfügen zusätzlicher Ereignisse und schließlich

*Erzeugen von wahlfreiem Rauschen*

*(iii)* das Vertauschen von Aktivitäten. Diese drei Formen sollten, wie nachfolgend argumentiert wird, in genau dieser Reihenfolge realisiert werden.

*Gezielt unvollständige Ereignisprotokolle*

Aktivitäten können zufällig übersprungen werden, indem zufallsbasiert Ereignisse aus der Ausführungsspur entfernt werden. Das bewirkt keine Vermeidung einer Aktivität (Abschnitt 4.2.5), sondern lediglich die Entfernung der zugehörigen *Simulationsaufzeichnung*. Dadurch können gezielt unvollständige Ereignisprotokolle produziert werde, die bei Tests von Werkzeugen des Process-Minings oder der Konformitätsprüfung zwischen Modellen und Protokollen Anwendung finden.

*Irreguläre Wiederholung von Aktivitäten*

In Abschnitt 4.2.5 und Abschnitt 4.2.6 wird gezeigt, wie künstliche, zusätzliche Aktivitäten, welche nicht Teil des ursprünglichen Prozessmodells waren, in die Simulationsdurchführung einbezogen werden können. Einige der in Abschnitt 4.1 vorgestellten Ansätze unterstützen jedoch auch die künstliche Wiederholung von im Modell *enthaltenen* Aktivitäten. Der im Rahmen dieser Arbeit vorgestellte Ansatz zur Generierung künstlicher Prozessausführungsspuren ermöglicht dies ebenfalls. Dazu wird lediglich eine konfigurierbare Anzahl an Ereignissen, welche in einer Prozessausführungsspur enthalten sind, zufällig *geklont* und in die Ereigniskette aufgenommen. Die Positionsangabe des redundanten Ereignisses wird auf Basis des Originals ermittelt. Alle nachfolgenden Ereignisse werden entsprechend verschoben. Dieser Schritt muss nach dem künstlichen Überspringen von Aktivitäten durchgeführt werden, da ansonsten durch das zufällige Löschen von Ereignissen die künstliche Wiederholung der Aktivitäten wieder eliminiert werden könnten.

*Vertauschen von Aktivitäten*

Das Vertauschen von Aktivitäten erfolgt ähnlich und ebenfalls ereignisbasiert. Der Unterschied liegt hier in der Selektion der zu vertauschenden Ereignisse. Eine Möglichkeit ist, nur unmittelbar aufeinanderfolgende Ereignisse zu vertauschen. Eine zweite Möglichkeit ist das Vertauschen zweier Ereignisse an zufällig festgelegten Positionen in der Ereigniskette. Hierzu sind entsprechend zwei Zufallszahlen notwendig. Zudem muss garantiert werden, dass die Menge an ungeordneten Paaren dieser Zufallszahlen keine Duplikate enthält. Ohne diese Garantie könnten dieselben Ereignisse mehrfach miteinander die Plätze in der Prozessausführungsspur tauschen, anstatt dass die gewünschte Anzahl an derartigen Unregelmäßigkeiten in der Prozessinstanz verwirklicht wird. Weiterhin muss dieser Schritt nach dem zufälligen Entfernen von Ereignissen stattfinden, um sicherzustellen, dass die vertauschten Ereignisse auch in der Prozessausführungsspur bestehen bleiben. Zudem wird so auch die Möglichkeit geschaffen, die vorab künstlich erzeugten Aktivitätswiederholungen zufallsbasiert in der Prozessausführungsspur zu positionieren. Andernfalls würden künstliche Wiederholungen von Aktivitäten nur unmittelbar aufeinanderfolgend auftreten. Die Anzahl aller Modifikationen zur Verursachung künstlichen Rauschens kann vom Nutzer beliebig angegeben werden.

*Berechnung von Zeitstempeln aus Referenzzeitpunkt und dynamischem Versatz*

Der Zeitstempel für das Auftreten von Ereignissen richtet sich nach zwei Parametern, namentlich *(i)* der Ankunftsrate neuer Prozessinstanzen und *(ii)* dem zeitlichen Versatz zwischen Aktivitäten. Ersteres bildet dabei den Referenzzeitpunkt $T_{start}$ für jedes Ereignis einer Prozessinstanz. Der Zeitstempel $T_1$ für das erste Ereignis der Prozessausführungsspur wird nach $T_1 = T_{start} + \epsilon_1$ berechnet. Für jedes weitere Ereignis gilt $T_{i+1} = T_i + \epsilon_i$ mit $1 < i < 1$ und $\epsilon_i > 0$. Mit $\epsilon_i$ wird ein dynamischer Versatz zwischen dem vorangegangenen und dem aktuellen

Ereignis bezeichnet[15]. Dieser kann entweder statisch oder zufällig gewählt werden. In letzterem Fall kann ein Zufallsgenerator mit einer beliebigen Wahrscheinlichkeitsverteilung konfiguriert werden, um $\epsilon$ zu ermitteln. Die zeitliche *Reihenfolge* der Ereignisse ist dabei bereits durch die *Positionsangaben* in der generierten Prozessausführungsspur festgelegt (Codeausschnitt 4.2). Auch $T_{start}$ kann für jede Instanz statisch oder zufallsbasiert festgelegt werden. Einzelne simulierte Instanzen können sich dabei zeitlich überschneiden. Falls dies nicht gewünscht ist, kann zwischen dem letzten Ereignis einer Instanz und dem ersten der nachfolgenden wiederum ein statischer oder zufällig gewählter zeitlicher Versatz nach Vorbild von $\epsilon$ ermittelt werden. Dieser Prozess der iterativen Berechnung des zeitlichen Auftretens künstlicher Ereignisse basiert auf dem in Abschnitt 3.2.2 allgemeinen Prinzip der Simulationsdurchführung im Rahmen einer ereignisorientierten Simulation.

Nachdem alle Modifikationen der Simulationsergebnisse und das Berechnen der Zeitstempel abgeschlossen sind, verbleibt lediglich die Überführung der Lösungen des SAT-Solvers in ein Ereignisprotokoll. Diese sind Atom-Tupel (Abschnitt 4.2.1), deren Semantik aus der ursprünglichen Alloy-Spezifikation hervorgeht. Alloy bietet zur Navigation in und zwischen den Tupeln die Möglichkeit, Anfragen im Alloy-Sprachgebrauch zu formulieren. So kann beispielsweise mit dem Ausdruck `searchAtPos[int[integer/min]].assoEl ∩ Task`[16] das Atom ermittelt werden, welches die im ersten Ereignis protokollierte Aktivität beschreibt. Auf dieselbe Art können alle mit einem Ereignis assoziierten Informationen ermittelt werden. Durch die zur Anfrage verwendeten Signaturnamen ist bekannt, welche Information als Extraktionsergebnis zu erwarten ist. Dieses wird unter dem Signaturnamen dann als Schlüssel-Wert-Paar im Protokoll abgelegt (Protokolldefinition in Abschnitt 3.3.1).

*Protokollbefüllung durch Alloy-Anfragen*

Zu den Schritten der Nachverarbeitung der Simulationsergebnisse gehört unter anderem das Erzeugen künstlichen Rauschens und auch die zeitliche Einordnung aller Ereignisse jeder Prozessausführungsspur. Nach dieser Modifikationen der Simulationsergebnisse, werden selbige schließlich in die Form eines Ereignisprotokolls übertragen.

### 4.2.8 *Simulation prototypischer Prozessinstanzen*

Die bisherige Beschreibung der Technik vernachlässigt die Semantik von im Prozessmodell kodierten *Empfehlungen*. Empfehlungen können in ihrer Kombination zur Beschreibung der „typischen Abläufe" eines Prozesses eingesetzt werden. In Abschnitt 4.2.4 wird lediglich die Abbildung *verpflichtender* Regeln auf eine zu Alloy konforme Repräsentation beschrieben. Damit ist zunächst nur sichergestellt, dass jede generierte Prozessausführungsspur bezüglich des zur Generierung verwendeten Prozessmodells valide ist. Mit den bisher beschriebenen Mitteln ist es folglich nicht möglich, die Protokollierung auf typische Prozessverläufe also *prototypische Prozessinstanzen* zu beschränken. Der Grund ist, dass Alloy auf Prädikatenlogik erster Stufe basiert und somit keine Unterscheidung zwischen

*Prototypische Prozessinstanzen*

15 Hier kann nicht von der Dauer der Durchführung einer Aktivität gesprochen werden, weil unbekannt ist, ob jede Aufgabe unmittelbar nach der Beendigung der vorangegangenen Aktivität gestartet wurde oder ob *Leerlaufzeiten* enthalten sind. Bei der Koordination von menschlichen Beteiligten durch beispielsweise ein Prozessausführungssystem fallen in der Regel Leerlaufzeiten an.

16 Die Funktion `searchAtPos` ist in Codeausschnitt 4.2 definiert.

```
process Dienstreise{
  task GenehmigungEinholen  //A
  task UnterkunftBuchen     //B
  task FlugBuchen           //C
  task TransferBuchen       //D
  task DokumenteArchivieren //E

  ensure once(GenehmigungEinholen)
  ensure once(FlugBuchen)
  ensure once(UnterkunftBuchen)
  ensure once(TransferBuchen)
  ensure sequence(GenehmigungEinholen,UnterkunftBuchen)
  ensure sequence(GenehmigungEinholen,FlugBuchen)
  ensure sequence(FlugBuchen,TransferBuchen)
  ensure sequence(FlugBuchen,DokumenteArchivieren)
  ensure sequence(UnterkunftBuchen,DokumenteArchivieren)

  advise response(FlugBuchen,TransferBuchen)

  milestone complete(of DokumenteArchivieren)
}
```

Codeausschnitt 4.14: DPIL-Prozessmodell mit Empfehlungen

verschiedenen Regelmodalitäten – wie beispielsweise in der Modallogik – zulässt. Nachfolgend wird erläutert, wie diese Limitierung kompensiert wird.

*Beispiel: Dienstreiseprozess*

Ein Beispiel für ein DPIL-Modell, welches neben verpflichtenden Regeln auch Empfehlungen beinhaltet, ist in Codeausschnitt 4.14 dargestellt. Dieses beschreibt eine vereinfachte Form eines Organisationsprozesses für Dienstreisen. Das Modell enthält fünf Aktivitäten und mehrere verpflichtende Regeln. Durch die `once`-Regeln wird reguliert, dass jede Aktivität maximal einmal durchgeführt werden darf. Die Kombination der verpflichtenden Sequenzregeln legt den ersten Teil der Prozessausführungsspuren fest. Zuerst muss `GenehmigungEinholen` ausgeführt werden. Anschließend können die Aktivitäten `UnterkunftBuchen` und `FlugBuchen` in beliebiger Reihenfolge ausgeführt werden. Die Meilensteindefinition legt fest, dass `DokumenteArchivieren` verpflichtend und als letztes auszuführen ist. Dafür muss allerdings vorher `UnterkunftBuchen` und `FlugBuchen` abgeschlossen worden sein. Insgesamt betrachtet bedeutet das, dass `TransferBuchen` die einzige optionale Aktivität ist. Jedoch gibt die im Modell enthaltene Empfehlung an, dass nach der Flugbuchung auch ein Transfer gebucht werden sollte.

Mit den bisher beschriebenen Mitteln würde eine Simulation die folgenden Prozessausführungsspuren erzeugen[17]:

- ABCE
- ACBE
- ABCDE (empfohlen)
- ACBDE (empfohlen)

17 Zur Verbesserung der Übersichtlichkeit ist jede Aktivität mit jeweils einem Buchstaben kodiert, wie es die Kommentare im Modell andeuten.

Nur zwei der vier Ausführungsspuren sind als „empfohlen" gekennzeichnet. In den ersten beiden wird die Empfehlung `advise response( FlugBuchen,TransferBuchen)` missachtet. Diese Empfehlung spiegelt einen *typischen* Prozessverlauf wider, welcher bei der Buchung eines Fluges auch die anschließende Buchung eines Transfers beinhaltet. Die bisherigen Ausführungen beschränken sich lediglich auf eine Abbildung der verpflichtenden Regeln (`ensure`). Transformiert man jedoch auch die Empfehlungen eines DPIL-Modells in eine Alloy-konforme Repräsentation, dann sind die generierten Prozessausführungsspuren sowohl valide als auch empfohlen. Die verpflichtenden Regeln eines DPIL-Modells unterscheiden sich von Empfehlungen syntaktisch lediglich hinsichtlich des Schlüsselworts `ensure` oder `advise` – wie auch Codeausschnitt 4.14 zeigt. Folglich ist es möglich, auch Empfehlungen mittels der in Abschnitt 4.2.4 beschriebenen Transformation auf Alloy-Sprachmittel abzubilden. Aus diesem Grund bietet der Spurgenerator einen weiteren Simulationsmodus an, der *jede* der im Eingabemodell enthaltenen DPIL-Regeln in eine zu Alloy konforme Repräsentation abbildet. Die Transformation der Empfehlungen selbst unterscheidet sich dabei nicht von der der verpflichtenden Regeln (Abschnitt 4.2.4). Der Unterschied liegt hier in der Simulationsdurchführung die in zwei Phasen erfolgt:

*Typische Prozessabläufe*

1. Zunächst werden sowohl verpflichtende Regeln als auch Empfehlungen in eine zu Alloy konforme Repräsentation überführt. Anschließend wird eine konfigurierbare Anzahl an Prozessausführungsspuren erzeugt. Diese Anzahl ist auf den Bereich von **1** bis einschließlich der konfigurierten Gesamtzahl an Prozessausführungsspuren beschränkt (Abschnitt 4.2.5). Das Ergebnis ist der erste Teil des zu erzeugenden Ereignisprotokolls, wovon jede Prozessausführungsspur sowohl alle verpflichtenden Regeln als auch alle Empfehlungen beachtet.

*Schritt 1: Generierung unter Beachtung von Regeln und Empfehlungen*

2. In der zweiten Phase werden ausschließlich verpflichtende DPIL-Regeln in das auf Alloy basierende Simulationsmodell überführt. Anschließend wird für dieses Simulationsmodell eine bestimmte Anzahl an Prozessausführungsspuren erzeugt. Diese Anzahl ergibt sich automatisch aus der Differenz der konfigurierten Gesamtzahl aller und der Anzahl der in der ersten Phase erzeugten Spuren. Das Ergebnis ist der zweite Teil des zu erzeugenden Ereignisprotokolls, wovon jede Prozessausführungsspur lediglich verpflichtende Regeln beachtet.

*Schritt 1: Generierung unter Beachtung von ausschließlich Regeln*

Die Kombination aus beiden Teilprotokollen ergibt das gewünschte Ereignisprotokoll. Umfasst die Menge der in der ersten Phase erzeugten Ausführungsspuren mehr als die Hälfte aller insgesamt generierten, dann ist mit unterschiedlicher Deutlichkeit eine Tendenz für empfohlene Prozessverläufe im Ereignisprotokoll zu erkennen. Von dieser Interpretation gehen auch gängige Process-Mining-Werkzeuge (z.B. [166]) aus. Betrachtet man das obenstehende Beispiel, dann würde der Spurgenerator die beiden Ausführungsspuren ABCDE und ACBDE häufiger erzeugen, also die übrigen zwei. Dies ermöglicht sogar eine vollständige Beschränkung der Generierung beispielhafter Prozessausführungsspuren auf empfohlene Verläufe, die als prototypisch angesehen werden können.

*Prototypische Spuren nach Häufigkeit*

Bisher wird ohne weitere Begründung davon ausgegangen, dass Empfehlungen prototypische Prozessverläufe modellieren. Ohne die Herkunft des Prozessmodells und vor allem die Motivation für die darin enthaltenen Empfehlungen zu kennen, ist diese Annahme äußerst vage. Empfehlungen können auch dazu verwendet werden, besonders kurze oder kostengünstige Prozessverläufe hervorzuheben. Die in [161] beschriebene Technik, um DPIL-Modelle mittels Process Mining zu erzeugen interpretiert Empfehlungen allerdings als Leitlinien für eben jene prototypischen Prozessinstanzen. Die Extraktion von Regeln aus den verfügbaren Ereignisprotokollen erfolgt auf Basis mehrerer Schwellwerte. Verpflichtende Regeln werden üblicherweise durch das Überschreiten sehr hoher Schwellwerte identifiziert. Regeln, welche diese Schwelle nicht überschreiten gelten entweder nicht für den beschriebenen Prozess oder werden nicht häufig genug eingehalten, um als obligatorisch zu gelten. Überschreitet die Häufigkeit der Einhaltung dieser Regel einen zweiten, niedrigeren Schwellwert, dann wird angenommen, dass die Regel dennoch eine gewisse Relevanz hinsichtlich der Beschreibung „üblicher" Prozessverläufe hat. Diese sind die gesuchten Empfehlungen und können aufgrund ihrer Herkunft als Richtlinien für prototypisches Verhalten interpretiert werden. Voraussetzung ist, dass in den für das Process Mining verwendeten Ereignisprotokollen *reale* Instanzen erfasst sind. Auf diese Weise können Erfahrungsdaten verwendet werden, um prototypische *Regeln* zu identifizieren. Umgekehrt ist jedoch nicht garantiert, dass im zugrundeliegenden Ereignisprotokoll eine oder mehrere Ausführungsspuren enthalten sind, die *alle* Empfehlungen berücksichtigen (*support* und *confidence* in Abschnitt 3.4.3). In diesem Fall stellt sich die Frage, welche der erfassten Verläufe des Prozesses als prototypisch aufgefasst werden können. Zudem muss berücksichtigt werden, dass Ereignisprotokolle lediglich einen Ausschnitt der Realität zeigen können und keine Garantie für Annahmen über zukünftige Prozessverläufe ermöglichen. In der vorliegenden Arbeit wird daher davon ausgegangen, dass eine prototypische Prozessinstanz *alle* identifizierten Empfehlungen befolgt – unabhängig davon, ob diese mittels Process Mining oder manuell modelliert wurden. Aus diesem Grund ist die Ermittlung prototypischer Prozessausführungsspuren mittels Spurgenerierung sinnvoll und kann realisiert werden, indem Empfehlungen wie obligatorische Regeln behandelt werden.

*Analoge Interpretation durch Process-Mining-Werkzeug*

*Lediglich Ausschnitt der Realität*

### 4.2.9 *Erweiterungsmöglichkeiten*

Die in den vorherigen Abschnitte beschriebene Technik zur Generierung künstlicher multi-perspektivischer, deklarativer Prozessmodelle bietet mehrere definierte Erweiterungspunkte. Da bisher existierende Spurgeneratoren für deklarative Prozessmodelle keine kombinierte Unterstützung der Funktionalen, Verhaltensorientierten, Organisatorischen und Datenorientierten Perspektive bieten, liegt der Fokus der vorliegenden Arbeit auf einer Lösung, welche diese Lücke schließt. Die Operationale Perspektive wird dabei ausgeblendet. Gleiches gilt für einige Details des Simulationsmodells. Im aktuellen Abschnitt werden daher definierte Erweiterungspunkte vorgestellt.

Aktuell kann lediglich die Bearbeitung einer Aktivität durch menschliche Prozessbeteiligte simuliert werden (`HumanTaskEvent` in Codeausschnitt 4.2), ohne auf

eventuelle Softwareunterstützung eingehen zu können. Mit `PEvent`, `TaskEvent` und `HumanTaskEvent` werden drei Abstraktionsstufen definiert. Es ist möglich, das Metamodell für Prozessausführungsspuren auf jeder dieser Abstraktionsstufen zu erweitern. Für die Unterstützung der Operationalen Perspektive kann die `AssociatedElement`-Signatur nach dem in Codeausschnitt 4.15 dargestellten Beispiel erweitert werden.

*Erweiterung des Alloy-Metamodells für Prozessausführungsspuren*

```
abstract sig Tool extends AssociatedElement{
 ...
}

sig AutoTaskEvent extends TaskEvent{}{
  #(Identity ∩ assoEl) = 0
  #(Tool ∩ assoEl) = 1
}
```

Codeausschnitt 4.15: Erweiterung für die Unterstützung der Operationalen Perspektive

Die neu eingeführte Signatur `Tool` erlaubt die Deklaration von Softwarewerkzeugen, die zur Bewältigung von Aufgaben genutzt werden können. Durch die Erweiterung der `AssociatedElement`-Signatur können Aufgaben, organisatorische Ressourcen, Daten und Softwarewerkzeuge beliebig miteinander assoziiert werden. Grundlage dafür ist die generische Relation `assoEl` der Signatur `TaskEvent`. Ein Teil der Operationalen Perspektive kann somit bereits abgedeckt werden. Zu dieser Perspektive gehören aber auch Aufgaben, die softwaregestützt automatisiert werden. Dafür wird zusätzlich die Signatur `AutoTaskEvent` eingeführt. Die Automatisierung der Durchführung der Aktivität wird durch das Verbot einer Assoziation mit einem menschlichen Ausführenden (`Identity`) ausgedrückt.

In der in Codeausschnitt 4.15 dargestellten Form können Softwarewerkzeuge lediglich *benannt* werden. Ist es gewünscht, zusätzlich eine bestimmte Spezifikation oder Konfiguration anzugeben, dann können der `Tool`-Signatur weitere Felder hinzugefügt werden. Dies ist beispielsweise bei einem Webservice sinnvoll, für den die zu nutzenden Verbindungsdaten anzugeben sind.

Eine weitere Limitierung ist die Beschränkung der Datentypen seitens Alloy. Beispielsweise sind numerische Datentypen auf den Bereich der ganzen Zahlen beschränkt. Das gewünschte Zahlenformat hängt jedoch vom Modellierungsgegenstand ab, weswegen auch die Verwendung von beispielsweise rationalen Zahlen gewünscht sein kann. Es gibt Methoden, um Bruchzahlen mittels Paare ganzer Zahlen darzustellen. Eine dieser Methoden ist die Repräsentation derselben als *Festkommazahlen*. Bei Festkommazahlen sind je eine festgelegte Anzahl von Ziffern als Vorkomma- und als Nachkommaziffern festgelegt. So lässt sich beispielsweise die Zahl $10,001$ als $10001$ darstellen, wobei die ersten zwei Ziffern als Vorkomma- und die verbliebenen als Nachkommaziffern festgelegt sind. Die Festlegung, wie viele Ziffern vor und nach dem Komma liegen, wird mittels eines sogenannten *Skalierungsfaktors* beschrieben. Das Produkt aus der ganzzahligen Darstellung des fraglichen Wertes und des Skalierungsfaktors ergibt jeweils den originalen Wert. Der Skalierungsfaktor für das genannte Beispiel ist demnach $1/1000$. Vereinfacht man das Konzept so, dass der Skalierungsfaktor lediglich zur Skalierung rationaler Zahlen auf ganzzahlige Repräsentationen verwendet wird, kann auch der Skalierungsfaktor als Ganzzahl dargestellt werden – im Beispiel also als $1000$.

*Darstellung rationaler Zahlen mittels ganzer Zahlen*

Die Repräsentation von Bruchzahlen erfolgt wiederum durch eine eigene Signatur, beispielsweise in der in Codeausschnitt 4.16 dargestellten Form.

```
sig Real{
  value: one Int,
  scale: one Int
}
```

Codeausschnitt 4.16: Erweiterung für die Automatisierung von Prozessen

Die im Codebeispiel eingeführte Signatur beschreibt die Kombination aus dem eigentlichen ganzzahlig dargestellten Zahlenwert (`value`) und dem Skalierungsfaktor (`scale`). Der ursprüngliche Wert $v_{original}$ kann folglich anhand der Formel $v_{original} = value/scale$ außerhalb von Alloy rekonstruiert werden.

*Keine arithmetischen Operationen*

Zu beachten ist, dass für diese Darstellung keinerlei arithmetische Operationen definiert sind und die Darstellung als Festkommazahl die Auflösung darzustellender Zahlenwerte beeinflusst. Ein fester Skalierungsfaktor $s$ erlaubt lediglich $\lfloor \log s \rfloor$ Nachkommastellen. Auch wird durch den Skalierungsfaktor der benötigte Wertebereich für ganze Zahlen beeinflusst. Selbiger muss, wie in Abschnitt 4.2.5 erläutert, in der Scope-Definition einer Alloy-Spezifikation angegeben werden.

Die hier beschriebenen konzeptionellen Erweiterungsmöglichkeiten sind keineswegs erschöpfend, sondern vielmehr exemplarisch. Beispielsweise unterstützt DPIL auch die Modellierung von Subprozessen und globalen Regeln, welche für alle Prozesse eingehalten werden müssen. Subprozesse können jedoch in Ereignisprotokollen nicht vermerkt werden und globale Regeln lassen sich auf Prozessregeln abbilden, die ohne ihre explizite Modellierung in *jedem* Simulationsmodell materialisiert werden. Im Allgemeinen besteht die Möglichkeit der Realisierung zusätzlicher Erweiterungen. Die Umsetzbarkeit spezieller Erweiterungen kann jedoch nur individuell geprüft werden.

### 4.2.10 *Ansatzspezifische Annahmen*

Die Funktionalität des in den vorangegangenen Abschnitten vorgestellten Ansatzes zur Generierung von Prozessausführungsspuren basiert auf einigen Annahmen, die nachfolgend zusammengefasst sind. Diese gehen aus den Erläuterungen der vorangegangenen Abschnitte hervor und werden daher im aktuellen Abschnitt nur knapp erläutert.

Die fraglichen Annahmen formulieren lediglich Grenzen für die aktuelle konzeptionelle Entwicklung des vorgestellten Spurgenerators oder Bedingungen für dessen Einsatz. Keine gefährdet jedoch die prinzipielle Funktionalität eines auf Alloy basierenden Spurgenerators für multi-perspektivische, deklarative Prozesmodelle. In Abschnitt 4.2.9 wird bereits auf Eliminierungsstrategien für einen Teil der konzeptionellen Einschränkungen eingegangen. Die Annahmen und Einsatzbedingungen lauten im Einzelnen:

1. Regeln und Meilensteine sind im DPIL-Modell lediglich mittels der in Tabelle 4.3 und Tabelle 4.4 dargestellten Schablonen formuliert.
2. Die Beziehungen zwischen einzelnen Ressourcen sowie zwischen diesen und komplexen Organisationseinheiten sind extern definiert.

3. Die möglichen Werte für in DPIL-Modellen enthaltenen Variablen sind entweder Zeichenketten oder ganzzahlig.

4. Für die Simulation möglicher Belegungen von in DPIL-Modellen enthaltenen Variablen genügt eine konfigurierbare Anzahl an Beispielen, welche sich an den in Prozessregeln enthaltenen Abhängigkeiten von Vergleichen mit diesen Datenwerten orientieren. Weiterhin existiert zu jeder Variable eine Regel, welche einen derartigen Vergleich enthält.

5. Die in den DPIL-Modellen enthaltenen Empfehlungen stellen Richtlinien dar, die in ihrer Kombination prototypische Prozessinstanzen beschreiben.

6. Historische Ereignisprotokolle sind entweder nicht verfügbar oder ihre Qualität ist fraglich.

7. Eine adäquate Konfiguration von Länge und Anzahl der zu generierenden Prozessausführungsspuren ist bekannt oder kann schrittweise angenähert werden.

8. Die Spurgenerierung sollte der Beantwortung einer konkreten Fragestellung dienen.

*Schablonenbasierte Modellierung*

Die in DPIL enthaltenen Schachtelungsmöglichkeiten mittels komplexer Ausdrücke gestatten eine potentiell unendlich große Anzahl an Regelschablonen. Folglich ist eine explizite Angabe aller Transformationsregeln zwangsläufig unvollständig. Daraus resultiert die erste Annahme der obenstehenden Liste. Diese kann eliminiert werden, indem die bereitgestellten Schablonen für die Abbildung von DPIL- auf Alloy-Konstrukte nicht auf *Regelebene*, sondern auf Ebene der einzelnen Regel*bestandteile* angegeben werden. Die einzelnen Bestandteile lassen sich in DPIL zu beliebig komplexen Regeln kombinieren, was sich durch die Aufteilung in kleinere Regelfragmente auch in Alloy realisieren ließe. Somit stellt diese Annahme lediglich eine momentane Einschränkung des Konzepts der Übersetzung von DPIL-Modellen in eine Alloy-Spezifikation dar.

*Organisationsstruktur extern definiert*

Die zweite Annahme wird getroffen, da DPIL selbst keine organisatorischen Strukturen kodiert. Diese werden extern festgelegt und sind demnach für den hier vorgestellten Simulationsansatz nicht zugänglich. Aus diesem Grund ist derzeit eine manuelle Konfiguration dieser Beziehungen mit den in Codeausschnitt 4.4 dargestellten Mitteln notwendig. Diese Annahme kann eliminiert werden, indem eine automatisierte Abbildung der extern modellierten Organisationsstrukturen auf eben dieses Alloy-basierte Metamodell für Organisationsstrukturen bereitgestellt wird. Da das verwendete organisatorische Metamodell jedoch selbst in Alloy definiert ist (Codeausschnitt 4.4), ist es jederzeit möglich, eine Instanzen desselben und damit organisatorische Strukturen direkt in Alloy zu modellieren.

*Variablenwerttypen eingeschränkt*

Alloy unterstützt lediglich Zeichenketten und ganzzahlige numerische Datentypen. Damit wird eine Darstellung von beispielsweise rationalen Zahlen nicht direkt unterstützt, was zu Annahme 3 führt. In Abschnitt 4.2.9 ist jedoch am Beispiel der rationalen Zahlen eine Möglichkeit zur Einführung komplexerer Datentypen beschrieben.

Die vierte Annahme behandelt die Problematik, dass Prozessausführungsspuren die Realität lediglich ausschnittsweise repräsentieren können. Das ist besonders dann problematisch, wenn beispielsweise Variablen einen kontinuierlichen Wertebereich aufweisen. Da Alloy standardmäßig nur ganzzahlige numerische Werte unterstützt, ist die Anzahl möglicher Datenwerte zwar zählbar aber dennoch potentiell unendlich groß. Aus diesem Grund wird für den vorab präsentierten Simulationsansatz eine Heuristik verwendet, welche die Anzahl der zu simulierenden Datenwerte begrenzt. Als Grundlage werden die im Prozessmodell enthaltenen Regeln verwendet, welche auf Basis des Vergleichs von Datenwerten die Menge valider Ausführungsspuren begrenzen. Voraussetzung ist demnach, dass das Modell derartige Regeln beinhaltet. Ist dies nicht der Fall, dann sind die Datenwerte eines Prozesses von diesem vollkommen unabhängig und durch Simulation erzeugte Daten haben somit auch keinerlei Informationsgehalt. Also stellt diese Annahme tatsächlich *keine* Einschränkung der Funktionalität des Spurgenerator dar.

*Anzahl gewünschter Beispiele möglicher Variablenbelegungen begrenzt*

Die fünfte Annahme ist für die Verwendung des Simulators zur Generierung von Ereignisprotokollen für prototypische Prozessinstanzen relevant. DPIL-Modelle können Empfehlungen beinhalten, deren Existenz auf verschiedenartige Gründe zurückgeführt werden kann. Die obenstehende Annahme legt jedoch fest, dass Empfehlungen ausschließlich zur Modellierung von Richtlinien für typische Prozessverläufe verwendet werden. Diese Annahme kann zwar nicht eliminiert dafür aber zumindest validiert werden. Falls für den zu simulierenden Prozess reale Ereignisprotokolle vorliegen, kann mittels Process Mining überprüft werden, ob die Einhaltung von Empfehlungen vergleichsweise häufig und folglich *typisch* ist. Ist das der Fall, so kann man die Annahme als korrekt ansehen. Eine Garantie kann jedoch in keinem Fall gegeben werden, da auch reale Ereignisprotokolle zwangsläufig unvollständig sind und keine absolute Vorhersage hinsichtlich *zukünftig typischer* Prozessverläufe zulässt. Diese Einschränkung betrifft jedoch lediglich die Interpretation von Empfehlungen als Beschreibung prototypischer Prozessverläufe. Der Spurgenerator ist dennoch in der Lage, empfohlene Prozessausführungsspuren zu bevorzugen, falls dies gewünscht ist.

*Empfehlungen markieren prototypisches Verhalten*

Der Einsatz eines Spurgenerators ist besonders dann sinnvoll, wenn keine historischen Ereignisprotokolle zur Verfügung stehen. Dies ist vor allem während der erstmaligen Modellierung eines Prozesses der Fall oder, wenn kürzlich Änderungen an einem bestehenden Modell vorgenommen worden. Existieren bereits Aufzeichnungen realer, historischer Prozessverläufe, dann ist der Einsatz eines Spurgenerators zu empfehlen, wenn die Qualität derselben entweder unbekannt oder ungenügend ist. Dies ist Inhalt der sechsten Annahme.

*Historische Ereignisprotokolle nicht verwendbar*

Die siebte Annahme betrachtet die Fähigkeit des Nutzers, eine zweckmäßige Konfiguration des Spurgenerators zu ermitteln. Verpflichtend müssen dabei Anzahl und Länge der Prozessausführungsspuren parametriert werden. Dies wird insofern abgeschwächt, als in Abschnitt 4.2.5 eine Faustregeln für die Bestimmung von unteren Schranken beider Parameter angegeben ist. Zudem kann das Verfahren auch dann verwendet werden, wenn eine schrittweise Annäherung an eine geeignete Konfiguration in den Arbeitsprozess des Nutzers zu integrieren ist.

*Konfiguration des Protokollumfangs bekannt*

Die achte und letzte Annahme ist lediglich eine Empfehlung. Da Ereignisprotokolle schnell umfangreich und unübersichtlich werden können, empfiehlt es sich,

*Konkrete Fragestellung*

den Spurgenerator zur Beantwortung einer konkreten Fragestellung zu verwenden. Eine solche Fragestellung kann beispielsweise die Ermittlung möglicher Prozessverläufe unter Vorgabe einer partiellen Prozessausführungsspur sein.

Mit diesem Kapitel wird ein Konzept vorgeschlagen, welches zur Erfüllung einer der zentralen Anforderungen der vorliegenden Arbeit notwendig ist (Anforderung A2 in Abschnitt 1.2). Das Konzept beschreibt einen Spurgenerator für deklarative Prozessmodelle, welcher als MuDePS bezeichnet wird (Abschnitt 2.1). Dieser setzt eine Anzahl bereits bekannter Funktionen verwandter Werkzeuge um (Tabelle 4.1). Als große Neuerung ist hierbei die Bereitstellung dieser Funktionalität für die Simulation *multi-perspektivischer*, deklarativer Prozessmodelle zu verstehen. Zudem können wahlweise Positiv- oder Gegenbeispiele für oder gegen ein gegebenes Prozessmodell erzeugt werden. Mit der logischen Fundierung auf Alloy ist es zudem möglich, die Spurgenerierung sehr differenziert zu beeinflussen. Eine detailliertere Diskussion der bereitgestellten Funktionalität wird im Evaluationsteil der Arbeit (Abschnitt 8.1.1) durchgeführt.

# 5 INDUKTIVE TRANSLATION VON PROZESSMODELLEN

Durch die große Menge verfügbarer und zum Teil sehr verschiedenartiger konzeptueller Prozessmodellierungssprachen besteht häufig die Notwendigkeit Prozessmodelle von einer Sprache in eine andere zu übersetzen. Ein entsprechendes Translationssystem kann die Möglichkeit eröffnen, Prozessmodelle auf verschiedenen Ausführungssystemen auszuführen und ein besseres Verständnis der Modellsemantik seitens der Domänenexperten zu entwickeln. Folglich sind die Anwendungsmöglichkeiten einer derartigen Technik besonders auf die Designphase sowie die Implementierungs- und Konfigurationsphase des Prozessmanagement-Lebenszyklus (Abschnitt 1.1.1) fokussiert. Ein Konzept eines solchen Systems, welches SiMiTra genannt wird, ist in diesem Abschnitt beschrieben (Abbildung 2.1).

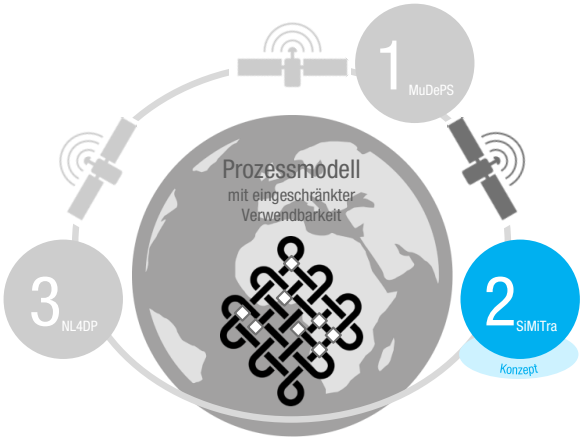



Abbildung 5.1: SiMiTra: Induktives Prinzip für die Translation von Prozessmodellen

Die Translation eines Prozessmodells ist eine Spezialisierung des Prinzips allgemeinen Modell-zu-Modell-Transformation. Letztere basieren nahezu ausschließlich auf der Definition von Transformationsregeln auf der Metamodellebene der Sprachen. Diese Regeln werden durch eine Transformations-Engine zur Instanziierung von Zielmodellelementen für eine selektierte Menge an Quellmodellelementen verwendet. In Abschnitt 5.1 werden zunächst existierende Vertreter dieser Klasse von Translationstechniken für deklarative und imperative Prozessmodelle analysiert. Anschließend werden Limitierungen des üblichen Prinzips der Modell-zu-Modell-Transformation diskutiert (Abschnitt 5.2.1). Diese spiegeln sich vordergründig in einem hohen Initialaufwand bei der Regeldefinition und der damit verbundenen Fehleranfälligkeit wider. Beides kann drastische Ausmaße annehmen, da bei einer Translation zwischen imperativen und deklarativen Sprachen Translationsregeln nicht mehr kontextfrei definiert werden können. Auf diesen Erkenntnissen aufbauend werden als Kern dieses Kapitels Potential und Konzept eines *induktiven* Translationsansatzes erläutert (Abschnitte 5.2.2 bis 5.2.4), was der zweite wesentliche Forschungsbeitrag der vorliegenden Arbeit ist (Abschnitt 1.2).

*Induktives Translationsverfahren*

## 5.1 VERWANDTE ARBEITEN

Als Ausgangspunkt und Begründung für die Entwicklung und Analyse der in Abschnitt 5.2 beschriebenen Technik zur Translation von Prozessmodellen wird in diesem Abschnitt eine Übersicht über bestehende Übersetzungstechniken gegeben. Als Grundlage dient eine Auswahl der gängigsten imperativen und deklarativen Prozessmodellierungssprachen, die auch in Abschnitt 1.2.2 zur Analyse der Interoperabilität konzeptueller Prozessmodellierungssprachen verwendet werden. Eine wesentliche Erkenntnis des aktuellen Abschnittes ist, dass vergleichsweise viele Ansätze zur Translation imperativer Prozessmodelle in alternative, aber imperative Prozessmodellierungssprachen existieren. Dagegen liegt, nach aktuellem Kenntnisstand, keine einzige Übersetzungstechniken mit einer deklarativen Prozessmodellierungssprache als Zielsprache vor. Für die Übersetzung deklarativer Prozessmodelle in eine imperative Sprache existieren ebenfalls nahezu keine Ansätze.

| ↦ | Ⓐ UML 2 AD | Ⓑ BPMN | Ⓒ EPC | Ⓓ PN | Ⓔ CPN | Ⓕ PT | Ⓖ YAWL | Ⓗ CMMN | Ⓘ CLIMB | Ⓙ Declare | Ⓚ MP-Declare | Ⓛ DPIL | Ⓜ TLA+ | Ⓝ DCR | Ⓞ EM-BrA²CE |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
| UML 2 AD | — | [134] | [134] | [63, 171] | [171] | | $\xrightarrow{B}$ | | | | | | | | |
| BPMN | [40, 114] | — | [134, 183] | [56, 63, 149] | [150] | | [53, 197] | | | | | | | | |
| EPC | [139, 174] | [178, 183] | — | [63] | $\xrightarrow{B}$ | | [127] | | | | | | | | |
| PN | $\xrightarrow{B}$ | [95] | $\xrightarrow{B}$ | — | $\xrightarrow{B}$ | | $\xrightarrow{B}$ | | | | | | | | |
| CPN | $\xrightarrow{D,B}$ | $\xrightarrow{D}$ | $\xrightarrow{D,B}$ | [89] | — | | [91] | | | | | | | | |
| PT | $\xrightarrow{D,B}$ | $\xrightarrow{D}$ | $\xrightarrow{D,B}$ | [32, 120] | $\xrightarrow{D,B}$ | — | $\xrightarrow{D,B}$ | | | | | | | | |
| YAWL | $\xrightarrow{D,B}$ | $\xrightarrow{D}$ | $\xrightarrow{D,B}$ | [196] | [158, 159] | | — | | | | | | | | |
| CMMN | | | | | | | | — | | | | | | | |
| CLIMB | | | | | | | | | — | | | | | | |
| Declare | $\xrightarrow{B}$ | [50] | $\xrightarrow{B}$ | [147] | $\xrightarrow{B}$ | | $\xrightarrow{B}$ | | | — | | | | | |
| MP-Declare | | | | | | | | | | | — | | | | |
| DPIL | | | | | | | | | | | | — | | | |
| TLA+ | | | | | | | | | | | | | — | | |
| DCR | | | | | | | | | | | | | | — | |
| EM-BrA²CE | | | | | | | | | | | | | | | — |

Tabelle 5.1: Translationstechniken für konzeptuelle Prozessmodelle

*Übersicht über verwandte Arbeiten*

Tabelle 5.1[1] gibt eine Übersicht über einen Großteil der existierenden Translationstechniken für Prozessmodelle. Übersetzungstechniken sind nicht zwangsläufig bidirektional. Der Zeilenname bezeichnet daher jeweils die Quell- und der Spaltenname die Zielsprache. Die matrixartige Tabelle unterteilt sich in vier Quadranten, welche jeweils eine andere Paradigmenkombination von Quell- und Zielsprache symbolisieren. Dabei ist die Hauptdiagonale der Matrix leer, was bedeutet, dass Ansätze, deren Quell- und Zielsprache identisch sind, hier nicht betrachtet werden. Hintergrund ist, dass es sich bei diesen Ansätzen nicht um Translationstechniken im eigentlichen Sinne handelt. Vielmehr dienen diese Techniken der Vereinfachung von Prozessmodellen, ohne jedoch einen Sprachwechsel zu vollziehen. Aus diesem Grund werden diese Verfahren für die vorliegende Arbeit ausgeblendet. Techniken für die direkte Übersetzung eines Prozessmodells von einer Quell- in die jeweilige Zielsprache sind in der Tabelle mit der Quellenangabe zur entsprechenden

1 Trotz großer konzeptioneller Unterschiede in den Sprachversionen der Business Process Model and Notation werden die beiden Versionen in der Tabelle zur Vereinfachung zusammengefasst. Dies verändert jedoch nicht die Grundaussage der Tabelle.

Veröffentlichung markiert. Andere Zellen enthalten Markierungen der Form $\xrightarrow{x}$. Diese Markierungen zeigen die Möglichkeit einer *transitiven* Übersetzung, wobei das Quellmodell in einem oder mehreren Zwischenschritten in eine der Nicht-Zielsprachen übersetzt wird. Ein Beispiel für eine solche transitive Transformation ist die Übersetzung eines YAWL-Modells in die BPMN mit der Übersetzung in eine Petri-Netz-Repräsentation als Zwischenschritt. Nicht in jedem Fall ist eine solche transitive Übersetzung verlustfrei möglich, was auch für das eben genannte Beispiel gilt. YAWL- und BPMN-Modelle unterstützen auch die Operationale und die Datenorientierte Perspektive, während das für Petri-Netze und ganz besonders für die im Prozessmanagementkontext häufig anzutreffenden Workflow-Netze nicht gilt. Ohne genaue Kenntnis aller an einer Translation beteiligten Prozessmodellierungssprachen ist der zu erwartende Erfolg einer Übersetzung vom jeweiligen Nutzer kaum einzuschätzen. Folglich ist eine *direkte* Übersetzung meist zu bevorzugen.

Nachfolgend werden einige der existierenden direkten Translationsansätze der Quadranten I und II von Tabelle 5.1 erläutert. Da aktuell für die Übersetzung imperativer Prozessmodelle in deklarative (Quadrant III) sowie die Übersetzung deklarativer Modelle in andere deklarative Sprachen (Quadrant IV) keine Ansätze existieren, entfällt eine Diskussion derselben vollständig. Die Tabelle zeigt, dass für lediglich 10% der möglichen Sprachpaarungen eine adäquate, direkte Übersetzungstechnik existiert. Berücksichtigt man auch die transitiven Abschlüsse, dann steigert dies die Abdeckung auf etwa 20%.

### 5.1.1 *Translation imperativ – imperativ*

Im ersten Quadranten (I) von Tabelle 5.1 sind Ansätze aufgeführt, welche imperative Prozessmodelle in andere imperative Prozessmodellierungssprachen übersetzen. Diese Aufstellung ist aller Wahrscheinlichkeit nach nicht vollständig[2], zeigt aber mit 45% Abdeckung ohne und 86% Abdeckung mit transitiven Abschlüssen im Vergleich zu den übrigen Quadranten deutlich den Unterschied in der Unterstützung der zwei Modellierungsparadigmen[3]. Die meisten dieser Translationstechniken übersetzen entweder BPMN- oder EPC-Modelle in andere imperative Sprachen. Nahezu jeder der aufgeführten Ansätze basiert auf dem in Abschnitt 3.5 beschriebenen Grundprinzip der Modell-zu-Modelltransformation, also der Definition von Abbildungsregeln auf Metamodellebene und deren Anwendung auf das jeweilige Quellmodell durch eine Transformations-Engine. Der in Abschnitt 5.2 beschriebene Translationsansatz beruht auf einem anderen Prinzip. Aus diesem Grund soll die *Anwendung* des M2MT-Prinzips hier lediglich am Beispiel der Transformation eines EPC-Modells in die Sprache BPMN beschrieben werden, um den grundlegenden Unterschied der zwei Prinzipien später verdeutlichen zu können.

*45% Abdeckung (transitiv: 86%)*

2 Da kein Register für derartige Translationstechniken existiert, mussten letztere einzeln recherchiert werden. Zudem sorgt die große Verbreitung der einzelnen Sprachen für eine Vielfalt an Werkzeugen – jedoch nicht zwangsläufig für deren wissenschaftlichen Beleg.

3 Diese Aufstellung ist im Hinblick auf die bei weitem längere Historie imperativer Prozessmodellierungssprachen plausibel.

In [178] wird eine regelbasierte Abbildung von EPC- auf BPMN-Modellelemente beschrieben. So wird beispielsweise eine EPC-Funktion in einen BPMN-Task transformiert. Ereignisse des EPC-Quellmodells können dagegen nicht so einfach auf Ereignisse in BPMN-Notation abgebildet werden, da ihre Semantik in vielen Fällen zu unterschiedlich ist. Beispielsweise können Ereignisse in EPC-Modellen das Auftreten von Situationen im Prozess beschreiben. Dagegen fungieren Ereignisse in BPMN eher als Auslöser oder „Verbraucher" verschiedenartiger Inhalte. Zu den wenigen Ausnahmen zählen beispielsweise explizite Eingangs- und Ausgangszustände in EPC-Modellen, die auf Start- respektive Endereignisse der BPMN-Notation abgebildet werden. Weiterhin existieren Regeln für die Abbildung der unterschiedlichen Formen von Verzweigungen und Zusammenführungen. Die EPC-Notation umfasst hierbei sogenannte Konnektoren, die entweder Exklusiv-Entscheidungen (XOR), Inklusiv-Entscheidungen (OR) oder Parallelisierungen (AND) beschreiben. Hierfür existieren entsprechende Gateways in der BPMN-Notation. Einige weitere Regeln beschreiben die Abbildung organisatorischer Einheiten (Organisatorische Perspektive), Daten und Informationen (Datenorientierte Perspektive) sowie EPC-Systeme (Operationale Perspektive). Hinzu kommen eine Reihe komplexerer Regeln, die eine Abbildung einzelner EPC-Modellelemente nur gestatten, wenn bestimmte Kontextbedingungen erfüllt sind. Dazu zählen beispielsweise Sendefunktionen für Nachrichten, die auf je ein Intermediate Message Event abgebildet werden können. Bildet die EPC-Sendefunktion jedoch die letzte Funktion der Kette, dann wird stattdessen ein End Message Event im BPMN-Zielmodell erzeugt. Besonders diese komplexeren kontextsensitiven Abbildungsregeln zeigen, dass der hier beschriebene Ansatz *sprachspezifisch* ist. Das gilt auch für einen Großteil der übrigen Ansätze der Tabelle. Allgemein ist ein Translationsansatz wünschenswert, der im Idealfall Prozessmodelle aus beliebigen Sprachen in jede beliebige andere Sprache übersetzen kann.

*Sprachspezifische Translationsregeln*

Ein besonderer Vertreter der M2MT-basierten Translationsansätze ist in [174] beschrieben. Darin weisen die Autoren auf die Fehleranfälligkeit sowie den potentiell hohen Zeitaufwand der manuellen Erzeugung von Modelltransformationsregeln hin. In Konsequenz zu dieser Erkenntnis wird in der vorliegenden Arbeit ein Translationsansatz vorgestellt, welcher zwar wiederum Regeln auf Ebene der Metamodelle verwendet, diese aber weitestgehend eigenständig erzeugt. Letzteres geschieht semi-automatisch auf Basis einiger vom Nutzer zur Verfügung gestellter Beispiel-Korrespondenzen zwischen Teilen eines Quell- und Teilen eines zugehörigen Zielmodells. Auf diese Weise *erlernt* das Translationssystem die Abbildungsregeln anhand von Beispielen, sodass eine manuelle Spezifikation derselben unnötig wird [174]. Allerdings wird die Notwendigkeit dieser Spezifikation nicht vollständig eliminiert, sondern lediglich auf die Modellebene verlagert. Der Nutzer beschreibt die Transformation nunmehr an einem konkreten Beispiel. Dadurch können sich jedoch Mehrdeutigkeiten ergeben, die in einem Nachbereitungsschritt bereinigt werden müssen. Die grundlegende Idee der Verwendung von Beispielen für die Translation von Prozessmodellen wird von dem in Abschnitt 5.2 beschriebenen Ansatz übernommen. Es wird jedoch gezeigt, dass das vollständige Konzept aus [174] sich nicht für die *paradigmenübergreifende* Übersetzung eignet, da sich

*Semi-automatische Ableitung von Transformationsregeln*

deklarative und imperative Prozessmodelle hinsichtlich der Prä- und Absenz von Modellelementen zur Beschreibung eines Prozesses grundlegend unterscheiden.

In [135] wird ein Ansatz beschrieben, der ebenfalls Transformationsregeln auf Ebene der Metamodelle verwendet. Allerdings handelt es sich hierbei um komplexere Regeln, die häufig wiederkehrende Muster in imperativen Prozessmodellen *abstrakt* beschreiben. Ein Beispiel dafür ist das sogenannte Verzweigungsmuster, wozu beispielsweise die AND, OR und XOR Gateways der BPMN-Notation aber auch die Konnektoren der EPC-Sprache gehören. Die Beschreibung dieser Verzweigungen erfolgt zunächst mittels einer Zwischensprache und wird anschließend in die gewünschte Zielsprache überführt. Damit ist dieser Ansatz einer der wenigen, die als sprach*unabhängig* bezeichnet werden können. Allerdings wird in [135] lediglich die Abbildung von EPC-Modellen auf die im Geschäftsprozess-Managementwerkzeug ADONIS[4] verwendete interne Repräsentation von Prozessmodellen sowie die Transformation in umgekehrter Richtung belegt. Zudem sind nachweislich viele der identifizierten Muster – besonders die in [160] beschriebenen Kontrollflussmuster – nicht auf deklarative Prozessmodelle anwendbar, da sie Teile des Prozesses explizit beschreiben [199].

*Sprachunabhängige Translationsregeln*

Der in [134] weicht vom Standardprinzip der M2MT ab. Die Transformationsregeln basieren hierbei nicht auf einer direkten Abbildungen von Elementen des Metamodells, sondern auf der Abbildung derselben auf Elemente eines generischen Metamodells und wiederum deren Transformation in Elemente der Zielsprache. Damit stellt dieser Ansatz eine Erweiterung der vorab beschriebenen Technik aus [135] dar. Die in Prozessmodellen häufig vorkommenden Muster werden verwendet, um das generische Metamodell aufzubauen. Da nicht jede Sprache alle Muster unterstützt, ist eine Translation nicht in jedem Falle möglich. Zudem basiert das identifizierte Metamodell lediglich auf der Musteranalyse für einen stark reduzierten Elementsatz der Sprachen BPMN, EPC, UML Activity Diagram sowie der ADONIS-internen Repräsentation. Eine Übertragung dieser, auf imperativen Mustern basierenden Technik auf Transformationsprobleme mit Beteiligung deklarativer Prozessmodellierungssprachen ist nicht möglich [199].

*Translation mit Zwischenrepräsentation*

Die übrigen existierenden Verfahren zur Translation imperativer Prozessmodelle werden hier nicht weiter betrachtet. Der Grund dafür ist einerseits, dass das in Abschnitt 5.2 beschriebene Verfahren im Gegensatz zu diesen Techniken nicht dem klassischen M2MT-Prinzip folgt. Andererseits ist die vergleichsweise hohe Abdeckung imperativ-imperativer Verfahren die Begründung dafür, dass sich die vorliegende Arbeit der weit größeren Forschungslücke der Übersetzung mit Beteiligung deklarativer Prozessmodellierungssprachen zuwendet.

### 5.1.2 *Translation deklarativ - imperativ*

Für die vergleichsweise junge Familie der deklarativen Prozessmodellierungssprachen existieren zum aktuellen Zeitpunkt im Wesentlichen zwei Translationsansätze mit einerseits den Petri-Netzen und andererseits der BPMN als Zielsprachen (Tabelle 5.1, Quadrant II). Diese beiden Ansätze werden nachfolgend erläutert.

4 https://de.boc-group.com/adonis/, letzter Zugriff: 06.02.2017

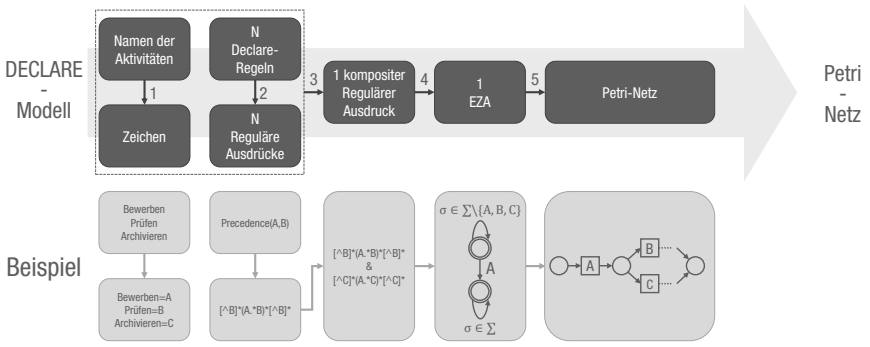



Abbildung 5.2: Prinzip der Translation von Declare-Modellen in die Petri-Netz-Notation

*Abbildung auf reguläre Ausdrücke*

*EZA*

In [147] wird ein Ansatz zur Übersetzung von Declare-Modellen in ein semantisch äquivalentes Petri-Netz beschrieben. Die in Abbildung 5.2 veranschaulichte Transformation beinhaltet fünf Schritte, wobei die ersten vier stark an den in Abbildung 4.2 skizzierten Declare-Simulationsansatzes erinnern. Der erste Schritt ist die eindeutige Abbildung aller Declare-Tasks auf jeweils ein Zeichen eines Alphabets. Anschließend werden alle Declare-Regeln in einen regulären Ausdruck für dieses Alphabet übersetzt. Für jeden regulären Ausdruck existiert ein äquivalenter, deterministischer, endlicher Zustandsautomat (EZA). Ein EZA entspricht dann folglich *einer* Declare-Regel. Die Regeln eines Declare-Modells sind jedoch mittels Konjunktion verknüpft, was direkt auf eine Und-Verknüpfung der einzelnen regulären Ausdrücke abgebildet wird. Der so entstehende einzelne, endliche Zustandsautomat umfasst somit alle Declare-Regeln. Schließlich wird der EZA auf Basis der sogenannten *theory of regions* in ein semantisch äquivalentes Petri-Netz übersetzt. Diese Theorie beschreibt die Gruppierung von Zuständen mit denselben Eingangs- und Ausgangstransitionen in gemeinsame *Regionen*. Diese Regionen werden in Petri-Netz-Plätze übersetzt.

*Modellierung nur mittels bekannter Regelschablonen*

Ein zentraler Kritikpunkt für diesen Ansatz ist die Tatsache, dass die Abbildung einer Declare-Regel auf einen regulären Ausdruck statisch definiert ist. Folglich können nur die Declare-Regelschablonen im Ziel-Petri-Netz berücksichtigt werden, für die ein regulärer Ausdruck hinterlegt ist. Zudem ist die Übertragbarkeit auf die Übersetzung multi-perspektivischer, deklarativer Prozessmodelle aufgrund der Eindimensionalität regulärer Ausdrücke problematisch. Die Kodierung der Ausführung einer Aktivität von einer bestimmten Ressource mittels eines bestimmten Software-Tools sorgt einerseits für eine drastische Vergrößerung des Alphabets und kann andererseits zu einer Zustandsexplosion bezüglich der resultierenden EZAs führen (Abschnitt 4.1.2). Standard-EZAs haben den Nachteil, dass jede Kante mit einer einzelnen Aktivität belegt ist und es dadurch möglich wird, dieselben Zustände auf unterschiedlichen Wegen zu erreichen. Es ist anschließend nicht nachvollziehbar, auf welcher Ausführungsspur man zum jeweiligen Zustand gelangt ist. Dieses Problem wird von dem in [147] beschriebenen Ansatz nicht diskutiert. Ein weiterer Nachteil dieser EZA-basierten Lösung ist das mögliche Auftreten von Redundanzen bezüglich der Aktivitätsnamen.

Die Quellsprache des in [50] beschriebenen Translationsansatzes ist ebenfalls Declare, wohingegen BPMN die Zielsprache ist. Es handelt sich dabei um einen

Ansatz zur *hybriden* Modellierung von Prozessen, wobei Teile desselben Prozesses mit imperativen und andere Teile mit deklarativen Konstrukten beschrieben werden können. Die Integration der beiden grundverschiedenen Sprachen erfolgt durch eine Abbildung der Declare-Regeln auf Kontrollflussmuster in BPMN-Notation. Als Wirtssprache dient dabei die BPMN. In Konsequenz dazu werden nur Declare-Regeln unterstützt, welche sich eindeutig auf eine BPMN-konforme Konstruktion abbilden lassen. Als Grundlage dient dabei eine rückwärtskompatbile Erweiterung der BPMN-Notation um deklarative Muster. Die Autoren geben an, dass jede deklarative Sprache, deren Semantik auf LTL basiert, auf diese Erweiterungskonstrukte abgebildet werden kann. Durch die Rückwärtskompatibilität ist es somit möglich, auch jedes Erweiterungskonstrukt in Standard-BPMN umzuformen.

*Abbildung auf Basis von Kontrollflussmustern*

Der Transformationsprozess erfolgt im Wesentlichen in drei Schritten. Im ersten Schritt werden die LTL-Ausdrücke für jede Declare-Regel, ähnlich wie in [147], in Zustandsautomaten übersetzt, die zusätzlich komprimiert werden. Anstatt der in [147] verwendeten deterministischen, endlichen Zustandsautomaten in Standard-Form verwenden die Autoren von [50] sogenannte *Constraint-Automaten* (CA). Diese tragen als Kantenbeschriftung Constraints anstelle von Symbolen, die Aktivitätsnamen repräsentieren. Ein Zustandsübergang kann bei dieser Automatenart durch das Auftreten *jedes* Zeichens erfolgen, welches das jeweilige Constraint erfüllt. Folglich akzeptiert ein solcher Automat jede Ausführungsspur, die in einem Anfangszustand beginnt, in einem Endzustand endet und dazwischen nur Zustände besucht, welche durch Constraint-konforme Transitionen erreicht werden können.

*Constraint-Automaten*

Im zweiten Schritt werden die Constraint-Automaten in die angesprochene BPMN-Erweiterung transformiert. Im Wesentlichen werden dabei alle Transitionen in BPMN-Aktivitäten und alle Zustände in Kontrollflusskonnektoren übersetzt. Die Abbildung erfolgt regelbasiert und ist auf BPMN als Zielsprache beschränkt. Im dritten und letzten Schritt werden die Erweiterungskonstrukte in Standard-BPMN-Konstruktionen transformiert. Dies geschieht auf Basis statischer Assoziationen derselben.

Die Verwendung von LTL-Formeln anstelle regulärer Ausdrücke als Zwischenrepräsentation der Declare-Regeln ermöglicht eine generische Übersetzung derselben in die gewünschten Constraint-Automaten. Folglich ist der aktuell besprochene Ansatz im Gegensatz zu dem in [147] beschriebenen nicht an vordefinierte Regelschablonen gebunden. Mit BPMN als Zielsprache ist zwar eine multiperspektivische Prozessmodellierungssprache involviert, jedoch ist die Quellsprache Declare auf die Funktionale und die Verhaltensorientierte Perspektive beschränkt. Deshalb kann der Ansatz keine Aussage hinsichtlich der Übertragbarkeit auf die Übersetzung *multi-perspektivischer* deklarativer Prozessmodelle in eine imperative Sprache treffen. Zudem ist im letzten Schritt eine statische Abbildung zwischen den eingeführten deklarativen Erweiterungskonstrukten entwickelt worden, deren Übertragbarkeit auf alternative imperative Prozessmodellierungssprachen nicht garantiert werden kann.

*LTL-Formeln anstelle regulärer Ausdrücke*

Vordergründiges Ziel von Abschnitt 5.1 ist es, die Abdeckung der gebräuchlichsten imperativen und dem Großteil der verfügbaren deklarativen Prozessmodellierungssprachen durch Translationstechniken zu analysieren. Eine wesentliche Erkenntnis ist, dass diese Techniken fast ausschließlich für imperative Sprachen

zur Verfügung stehen. Aber auch hier wird nur etwa die Hälfte der Paarungen der gebräuchlichsten Sprachen direkt unterstützt. Im Bereich der deklarativen Sprachen stehen nahezu gar keine derartigen Techniken zur Verfügung. Aufgrund der geringen Abdeckung durch Translationstechniken, des hohen Initialaufwandes bei deren Entwicklung sowie einer damit verbundenen hohen Fehleranfälligkeit (Abschnitt 5.2.1) ist ein generisches Prinzip (Abschnitt 5.2) erstrebenswert.

## 5.2 PROZESSMODELL-TRANSLATION MITTELS PROZESSSIMULATION UND PROCESS MINING

In den vorherigen Abschnitten werden Ansätze beschrieben, welche auf Basis *explizit* modellierter Abbildungen von Strukturen der Quell- auf Strukturen der Zielsprache Translationen von Prozessmodellen ermöglichen. Im aktuellen Abschnitt wird ein alternatives Prinzip beschrieben, welches in Abbildung 5.3 skizziert ist und den zweiten der drei Kernbeiträge der vorliegenden Arbeit darstellt. Das Prinzip wird im Verlauf der Arbeit SiMiTra genannt (Abschnitt 1.2).

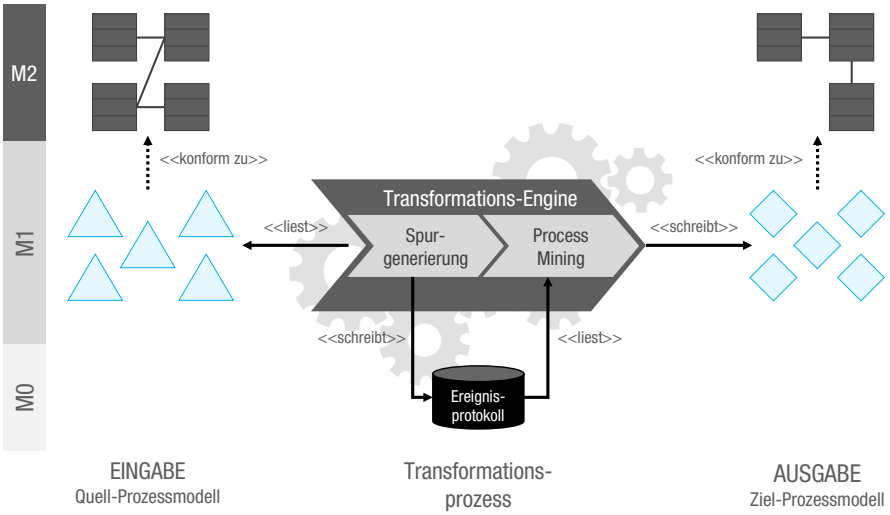



Abbildung 5.3: Prinzip der induktiven Translation von Prozessmodellen

*Allgemeines Prinzip*

Die Translation erfolgt im Wesentlichen in zwei Schritten. Im ersten Schritt wird für das Quell-Prozessmodell ein Spurgenerator (Kapitel 4) verwendet, um modellkonforme aber künstliche Prozessausführungsspuren zu generieren. Das so entstandene Ereignisprotokoll wird im zweiten Schritt von einer geeigneten Process-Mining-Technik analysiert und zur Erzeugung eines Prozessmodells in der gewünschten Zielsprache verwertet. Im Gegensatz zu klassischen Modell-zu-Modell-Transformationstechniken erfolgt somit die Translation nicht durch die Definition von Transformationsregeln auf Ebene der Metamodelle, sondern durch die Analyse von künstlichen Instanzen. Aus diesem Grund wird der hier beschriebene Ansatz als *induktiv* bezeichnet, wobei die Ereignisprotokolle als *Transfermedium* dienen. Letztere haben somit eine ähnliche Position wie die in [134] eingeführte Zwischensprache. Der grundlegende Unterschied ist, dass die Zwischensprache ebenso ein Modell ist (M1 in Abbildung 5.3), während das Ereignisprotokoll mögliche Instanzen des Quell-Prozessmodells beschreibt.

*Induktion statt Transformation*

*Überarbeitung während der Translation*

Das bietet die Möglichkeit, simultan zur Übersetzung eine *Überarbeitung* des Modells durch eine Neuausrichtung an Erfahrungsdaten vorzunehmen. Die Verwendung rein historischer Daten kann jedoch aus verschiedenen, bereits in Abschnitt 3.3.2 genannten Gründen auch problematisch sein. Eine Kombination mit künstlichen Prozessausführungsspuren hat das Potenzial, beispielsweise die Überanpassung auf Standardabläufe zu verhindern, indem seltene oder fehlende Prozessverläufe durch Simulation künstlich protokolliert werden. Die Translation auf Basis rein historischer Ereignisprotokolle wird nachfolgend nicht betrachtet, da dies die Translation auf ein klassisches Process-Mining-Szenario reduzieren würde und dies nicht Betrachtungsgegenstand der vorliegenden Arbeit ist.

*Wiederverwendung statt Neuentwicklung*

Durch die Kombination aus Spurgenerierung und Process Mining wird die Definition von Transformationsregeln umgangen. Allerdings ist die Definition der Spurgeneratoren und passender Process-Mining-Verfahren ein Mehraufwand, der die Praxistauglichkeit der induktiven Methode infrage stellt. Spurgeneratoren werden aber, wie Abschnitt 4.1 zeigt, auch für andere Zwecke benötigt, zum Beispiel zur Evaluation von Process-Mining-Verfahren. Letztere wiederum haben vielfältige Anwendungsgebiete (Abschnitt 3.4). Damit soll an dieser Stelle hervorgehoben werden, dass das hier vorgestellte Translationsprinzip zwar von der Verfügbarkeit von Spurgenerator und Process-Mining-Technik für das jeweilige Sprachpaar abhängig ist, diese aber in anderen Anwendungsgebieten ebenfalls benötigt werden. Aus diesem Grund wird die Entwicklung dieser Techniken nicht als Teil des induktiven Translationsansatzes angesehen; selbiger greift stattdessen ausschließlich auf *existierende* Ansätze zurück. Damit ist das Prinzip selbst generisch.

Die Abweichung vom klassisch-regelbasierten Translationsprinzip ist die Konsequenz einiger zentraler Schwächen desselben, welche nachfolgend erläutert werden (Abschnitt 5.2.1). Darauf aufbauend wird im Anschluss argumentiert, warum das eben beschriebene Prinzip einen Teil dieser Schwächen umgeht. Dazu werden die relevanten Eigenschaften von Ereignisprotokollen in ihrer Rolle als Transfermedium analysiert. Die Möglichkeit der Anwendung dieses Prinzips auf eine Auswahl von Prozessmodellierungssprachen, notwendige Parametrierungen sowie eine Übersicht über die verwendeten vereinfachenden Annahmen bilden den Abschluss dieses Kapitels.

### 5.2.1 *Limitierung regelbasierter Translationsansätze*

*Schwächen regelbasierter Translationsansätze*

Um den im Folgeabschnitt beschriebenen *induktiven* Translationsansatz rechtfertigen und einordnen zu können, werden im aktuellen Abschnitt einige ausgewählte Schwächen *regelbasierter* Ansätze diskutiert. Diese sind:

- Ein hoher *technischer Aufwand* bezüglich der Definition der Transformationsregeln und eventueller Zwischenrepräsentationen,
- eine erhöhte *Fehleranfälligkeit* besonders bei der Definition von paradigmenübergreifenden Transformationsregeln und
- die fehlende Möglichkeit der Berücksichtigung von *Erfahrungsdaten*, beispielsweise in Form historischer Ereignisprotokolle.

Translationsansätze unterscheiden sich unter anderem darin, ob die Abbildung von Quell- auf Zielmodellelemente *direkt* oder auf einer *Zwischensprache* basierend definiert wird (Abschnitt 3.5). Die bisher betrachteten Techniken (Abschnitt 5.1) gehören größtenteils zur Klasse der direkten und einige wenige gehören zu den auf einer Zwischensprache basierenden Translationsansätzen. Keine Technik dieser beiden Klassen ist bisher in der Lage, Prozessmodelle einer *beliebigen* in eine andere, ebenso beliebige Sprache zu übersetzen. Allerdings unterscheidet sich der *technische Aufwand* zur Definition von Transformationsregeln für direkte und Interlingua-basierte Translationssysteme drastisch.

*Hoher technischer Aufwand*

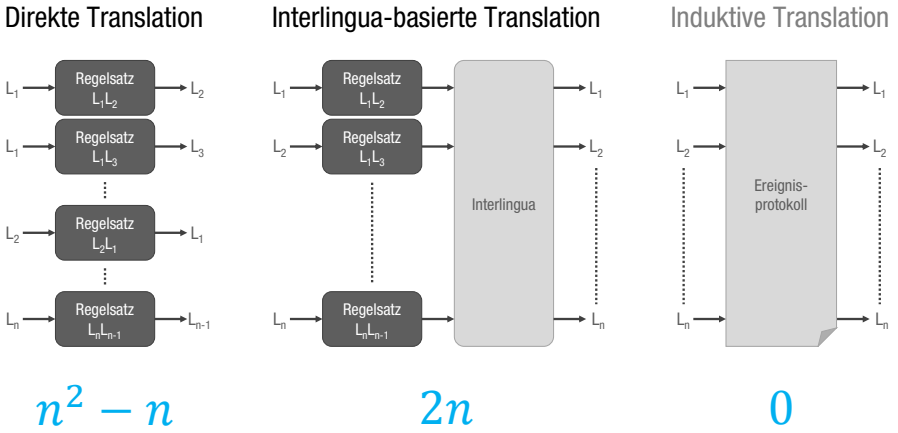



Abbildung 5.4: Technischer Aufwand der Erzeugung von Translationsregeln

*Konservativ: Direkte oder Interlingua-basierte Translation*

Wie Abbildung 5.4 zeigt, werden für die direkte, paarweise Übersetzung zwischen $n$ Sprachen $n^2 - n$ unidirektionale Translationsregelsätze benötigt. Durch die Einführung einer Zwischensprache kann der Aufwand auf eine lineare Größenordnung reduziert werden. Der Grund ist, dass für die unidirektionale Translation von einer beliebigen Prozessmodellierungssprache in die Zwischensprache *ein* spezifischer Regelsatz benötigt wird. Für die Gegenrichtung ist lediglich ein weiterer erforderlich. Der technische Aufwand wird hier allerdings nur dann reduziert, wenn eine für das jeweilige Sprachpaar *passende* Zwischensprache existiert. Einige der in Abschnitt 5.1 beschriebenen Techniken definieren zwar derartige Zwischensprachen, allerdings sind diese allesamt auf die Abbildung der Semantik *imperativer* Prozessmodellierungssprachen ausgerichtet (z.B. [134]). Somit beinhaltet die Zwischensprache Elemente wie Konnektoren für Sequenzflüsse und Verzweigungen, wie sie in imperativen, auf gerichteten Graphen basierenden Prozessmodellierungssprachen vorkommen. Eine explizit für die Abbildung der Semantik imperativer und deklarativer Prozessmodellierungssprachen entworfene Zwischensprache existiert zum aktuellen Zeitpunkt nicht. Aus diesem Grund muss entweder ein erhöhter Aufwand der Abbildung deklarativer Prozessmodelle auf eine der existierenden Zwischensprachen in Kauf genommen werden oder es muss zunächst eine passende Zwischensprache definiert werden. Andernfalls ist es nicht möglich, die Menge der notwendigen Translationsregelsätze zu reduzieren. Der im nachfolgenden Abschnitt beschriebene Ansatz verwendet keine Translationsregeln im ursprünglichen Sinne, sodass kein Mehraufwand für die Definition derselben entsteht. Folglich bietet sich selbiger besonders für Übersetzungen mit Paradigmenwechsel an.

Die Definition von Transformationsregeln für die Translation von Prozessmodellen ist nicht nur aufwendig, sondern auch fehleranfällig [174] und unkomfortabel. Eine besondere Herausforderung stellen die Translationen mit Paradigmenwechsel, also Übersetzungen deklarativer Modelle in imperative oder umgekehrt, dar. Das Beispiel Abbildung 5.5 veranschaulicht diese Problematik.

*Fehleranfälligkeit*

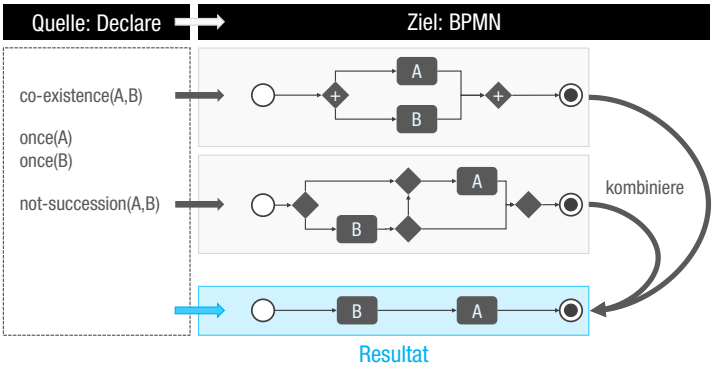



Abbildung 5.5: Notwendigkeit der Kontextsensitivität der Translationsregeln

Die Abbildung zeigt ein Declare-Modell (links). Für die Regeln $\mathtt{co\text{-}existence}(A, B)$ und $\mathtt{not\text{-}succession}(A, B)$ werden jeweils sich äquivalent verhaltende BPMN-Konstruktionen angegeben, wobei zur Vereinfachung auch die beiden $\mathtt{once}$-Regeln berücksichtigt werden. Das Beispiel beschreibt also eine direkte Abbildung zwischen deklarativen Regeln auf ein imperatives Teilmodell. Das Resultat zeigt jedoch, dass diese Abbildungsregeln nicht kontextfrei definiert werden können. Betrachtet man beispielsweise die Abbildung der $\mathtt{not\text{-}succession}(A, B)$-Regel, dann erlaubt diese unter anderem eine exklusive Ausführung der modellierten Aktivitäten. Die $\mathtt{co\text{-}existence}(A, B)$-Regel verbietet dies jedoch. Dadurch entfällt jede Verzweigung und außerdem jede Kante, die nicht vom Startereignis über $B$, weiter über $A$ zum Endereignis führt.

Aus dem Beispiel lässt sich schlussfolgern, dass eine wesentliche Ursache für die Fehleranfälligkeit darin liegt, dass die *Existenz* von Modellelementen in der Quellsprache häufig eine *Absenz* von Elementen in der Zielsprache bedeutet. Zudem können die Regeln deklarativer Sprachen für die Abbildung auf Konstrukte der imperativen Zielsprache nicht unabhängig voneinander betrachtet werden.

*Gegenläufige Existenzbedingungen*

Nicht immer ist die Translation des Ist-Standes eines Prozessmodells gewünscht. Denkbare Gründe sind beispielsweise Veränderungen der vom Modell beschriebenen Abläufe oder ein nach und nach erworbenes, fundiertes Wissen über vorher eher diffuse Prozessverläufe. Dadurch wird ein existierendes Modell oft nicht länger als adäquate Prozessbeschreibung angesehen. Die manuelle Anpassung des Modells *vor* der Translation ist eine Möglichkeit, die Konsistenz wiederherzustellen. Eine andere ist die Anpassung *während* der Translation durch Ausnutzung vorhandener Erfahrungsdaten, z.B. in Form von Ereignisprotokollen. Wie Abschnitt 3.3.2 entnommen werden kann, ist davon auszugehen, dass Prozesswissen durch verschiedenartige Systeme in strukturierter Form abgelegt wird; *Erfahrungsdaten* sind also in den meisten Fällen vorhanden. Die Berücksichtigung derselben ist in allen vorab beschriebenen Ansätzen (Abschnitt 5.1) und im allgemeinen Prinzip der Modell-zu-Modell-Transformation weder vorgesehen noch explizit unterstützt.

*Erfahrungsdaten*

### 5.2.2 *Eigenschaften des Transfermediums*

Bei einigen Translationstechniken wird ein Transfermedium verwendet, um die Zusammenhänge des Quell-Prozessmodells zu kodieren und die Anzahl notwendiger Regelsätze für Übersetzungen in andere Sprachen zu reduzieren. Dafür wird eine formale Zwischensprache eingeführt, welche in der Lage sein muss, die Semantik aller zu berücksichtigenden Sprachen abzubilden. Künstlich erzeugte Ereignisprotokolle nehmen in dem in der vorliegenden Arbeit beschriebenen Translationsprinzip eine ähnliche Funktion ein, unterscheiden sich von jenen Zwischensprachen aber grundlegend. Ziel dieses Abschnitts ist es, die Eigenschaften und Limitierungen der Ereignisprotokolle im Kontext der Translation zu analysieren. Die Beantwortung der Frage, welche Informationen ein Ereignisprotokoll beinhalten *kann*, ist notwendig, um die dadurch vorgegebenen Potentialgrenzen des vorgestellten Translationsansatzes zu identifizieren. Da auch jeder Process-Mining-Ansatz diesen Beschränkungen unterliegt, wird als ergänzende Literatur auf die Quelle [6] verwiesen.

*Abdeckung der Prozessperspektiven*

Eine mögliche Dimension der Eigenschaftsanalyse von Ereignisprotokollen ist die Abdeckung der einzelnen *Prozessperspektiven* (Abschnitt 1.1.2). Die allgemeine Definition der Ereignisprotokolle aus Abschnitt 3.3 beschreibt die Informationen, die mit einem Ereignis assoziiert sind generisch als Schlüssel-Wert-Paare. Damit ist es möglich, unter einem eindeutigen Schlüssel sowohl den Namen des Prozessschritts, des Ausführenden als auch Namen involvierter Datenobjekte und Softwarewerkzeuge anzugeben. Die Verhaltensorientierte Perspektive kann dagegen implizit durch die Totalordnung der Ereignisse einer protokollierten Instanz im Ereignisprotokoll kodiert werden. Jede Ausführungsspur beschreibt genau einen möglichen Pfad durch ein Prozessmodell. Folglich könnte man annehmen, dass die Ereignisprotokolle alle Perspektiven adäquat repräsentieren können. Das trifft jedoch nicht in jedem Fall zu, was aus der nachfolgenden differenzierteren Betrachtung ersichtlich wird.

Zwar gestatten Ereignisprotokolle die Berücksichtigung jeder *Perspektive*, nicht jedoch aller *Informationen* derselben. Ereignisprotokolle geben keinen Aufschluss bezüglich folgender Fragestellungen:

1. Ist ein Teilprozess, zum Beispiel ein Prozessschritt, beliebig oft wiederholbar?
2. Auf welcher Entscheidungsgrundlage kann sich der Prozess verzweigen und welche Informationsgrundlage führt zu welcher Entscheidung?
3. Welche Werte kann ein Datenobjekt annehmen?
4. Ist die Assoziation derselben Entität zu zwei verschiedenen Prozessschritten auf eine dynamische oder statische Beziehung zurückzuführen?

*Schleifen vs. Abgeschlossene Ereignisprotokolle*

Eine wesentliche Restriktion ist die Annahme der *Vollständigkeit*. Viele Process-Mining-Techniken basieren auf der Annahme, dass alle möglichen Verläufe des Prozesses im Ereignisprotokoll festgehalten sind [6, S. 199]. Enthält das Prozessmodell implizite oder explizite, unbeschränkte Schleifen (Fragestellung 1), dann kann nur eine Teilmenge aller möglichen Verläufe vollständig protokolliert werden,

da die Anzahl derselben unendlich ist. Einige Algorithmen basieren daher auf einer abgeschwächten Vollständigkeitsannahme, die lediglich vorschreibt, dass jedes mögliche Paar *direkt aufeinanderfolgender* Prozessschritte zumindest einmal protokolliert worden sein muss. Die Anzahl dieser Paare ist endlich und somit kann ein Ereignisprotokoll diese Information enthalten.

Verzweigungen in imperativen Prozessmodellen basieren meist auf exklusiven (XOR) oder inklusiven (OR) Entscheidungen oder der Parallelisierung von Aktivitäten (AND). Wie Abbildung 5.6 zeigt, können inklusive Entscheidungen jedoch mit einer Kombination aus exklusiven Entscheidungen und einer Parallelisierung nachgebildet werden. Gültige Ausführungsspuren für beide Beispiele der Abbildung sind $\{A\}, \{B\}, \{A, B\}, \{B, A\}$. Die Entscheidungen, die jeweils zu einer dieser Spuren führen, können *nicht* im Ereignisprotokoll kodiert werden (Fragestellung 2), da selbiges lediglich die Protokollierung der Ausführung von Aktivitäten gestattet. Somit ist es für das gegebene Beispiel nicht möglich, zu ermitteln, ob Aktivität $A$ aufgrund der Lage des Reiseziels im Inland und der Entscheidung, Gepäck mitzunehmen ausgeführt wurde (a) oder ob dies auf die Entscheidung für den PKW als Reisemittel zurückzuführen ist (b).

*Keine Protokollierung von Entscheidungen*

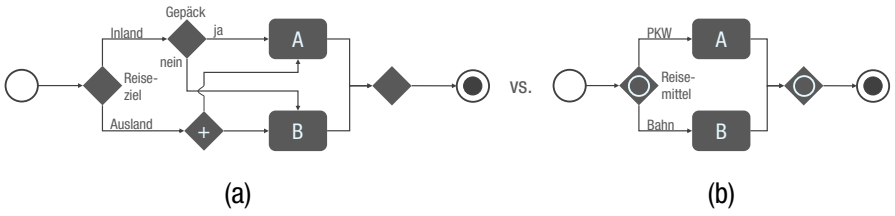



Abbildung 5.6: Mehrdeutigkeit der Protokollierung von Entscheidungen

*Ableitung des Wertebereichs für Datenobjekte nicht möglich*

Die Ableitung des gültigen Wertebereichs für in den Prozess involvierte Datenobjekte aus Ereignisprotokollen ist im Allgemeinen nicht möglich (Fragestellung 3). Der Grund ist, wie auch im Falle der ersten Fragestellung, dass Ereignisprotokolle endlich sind, während die Anzahl möglicher Wertebelegungen für ein Datenobjekt potentiell unendlich sein kann.

*Statische vs. dynamische Beziehungen in Ereignisprotokollen*

Die letzte Fragestellung bezieht sich auf die Unterscheidung dynamischer und statischer Assoziationen von Prozessentitäten. Die Assoziation einer Ressource oder eines Softwarewerkzeugs mit einer Aktivität kann durch ein entsprechendes Attribut im Protokolleintrag für das jeweilige Ereignis vermerkt werden. Werden zwei unterschiedlichen Schritten beispielsweise immer dieselbe Ressource zugewiesen und unterscheidet sich diese nur von Instanz zu Instanz, dann suggeriert dies intuitiv eine organisatorische Abhängigkeit der beiden Schritte. Vergleicht man beispielsweise die DPIL-Regel `direct(A,i) and direct(B,i) or direct(A,j) and direct(B,j)` mit der Regel `binding(A,B)`, dann ist eine Unterscheidung erst dann möglich, wenn mindestens drei Identitäten in den Prozess involviert sind. Andernfalls wären die möglichen Assoziationen der Aktivitäten `A` und `B` mit den jeweiligen Ausführenden identisch zu denen für die `binding`-Regel. Selbst wenn man von der Vollständigkeit des Ereignisprotokolls ausgeht, ist es dennoch nicht möglich, eindeutig zu entscheiden, welche DPIL-Regel im Quellmodell enthalten ist. Dennoch kann die Ausführungssemantik potentiell erhalten bleiben und betrifft daher eher die Translation auf konzeptueller Ebene.

Bislang wird ausschließlich von der Frage ausgegangen, welche Informationen ein Ereignisprotokoll kodieren kann und welche nicht. Diese Schwarz-Weiß-Kategorisierung deckt jedoch nicht alle denkbaren Fragestellungen ab. Zwei weitere, für die spezielle Protokollinhalte gefordert werden müssen, sind:

- Ist das Quell-Prozessmodell hierarchisch aufgebaut und enthält es demnach Subprozesse?
- Beschreibt das Modell Interaktionen zwischen verschiedenen Prozessinstanzen?

*Subprozesse vs. flache Ereignisprotokolle*

Zur Steigerung der Lesbarkeit von Prozessmodellen können Teile als Subprozess modelliert werden. Im Falle eines Modells zur Beschreibung eines Dienstreiseprozesses kann die Ermittlung der Reisegesamtkosten komplex sein. Eine Kapselung aller notwendigen Schritte in einem Subprozess blendet diese Komplexität bei Bedarf aus. Ereignisprotokolle bestehen nach der Definition aus Abschnitt 3.3 aus Ausführungsspuren und diese wiederum aus Ereignissen, welche die Ausführung von Aktivitäten beschreiben. Letztere können entweder atomar oder komplex sein – und in letzterem Fall einen Subprozess beschreiben. Da die Definition keine explizite Schachtelung von Ereignissen zulässt, kann die Unterscheidung zwischen komplexer Aktivität und „Sub-Aktivitäten" nur durch die Aufteilung der Ereignisse auf zwei verschiedene Prozessausführungsspuren ermöglicht werden. Die Beziehung zwischen diesen beiden kann mittels eines Spezialattributs kodiert werden, dass von Process-Mining-Werkzeugen als Referenz verwertet werden kann.

*Interaktion zwischen Instanzen vs. entkoppelte Instanzen*

Für die Protokollierung von Interaktion zwischen verschiedenen Instanzen desselben Prozesses oder unterschiedlicher Prozesse kann ein ähnliches Prinzip angewendet werden. Voraussetzung ist, dass die Interaktion explizit im Modell, d.h. durch eine Aktivität modelliert ist, da sich Ereignisse auf die Ausführung von Aktivitäten beziehen. So können beispielsweise empfangene Daten wie gewohnt als Schlüssel-Wert-Paare erfasst werden. Ein Zusatzattribut mit einem Verweis auf die andere Instanz kann zur Kodierung der Datenquelle eingeführt werden.

Eine grundlegende Erkenntnis dieses Abschnitts und ein weiterer Beitrag der vorliegenden Arbeit ist, dass verschiedene Konstrukte und Elemente des Prozessmodells, deren protokollierte Ausführungen gleich sind, nicht voneinander unterschieden werden können. In den zugehörigen Beispielen hat das jedoch nur Auswirkungen für das konzeptuelle Modell. Für eine Translation zum Zwecke der Portierung auf ein alternatives Ausführungssystem sind derartige Probleme irrelevant. Eine weitere zentrale Erkenntnis ist, dass die für die Translation verwendeten Spurgeneratoren bestenfalls die Vollständigkeit des Ereignisprotokolls garantieren können müssen. Ist dies nicht möglich, dann sollten zumindest die Präsenz der erlaubten Sequenzpaare aller Aktivitäten sowie aller Assoziationen organisatorischer und operationaler Entitäten mit Aktivitäten garantiert werden können. Das Ableiten des Wertebereichs für Datenobjekte ist dagegen nicht möglich, was jedoch kein für den hier beschriebenen Translationsansatz spezifisches Problem ist, sondern eine allgemeine Limitierung induktiver Verfahren darstellt.

Da das Auftreten der oben genannten Probleme stark von der Wahl von Quell- und Zielsprache für die Translation abhängig ist, ist die Liste der aufgezählten Limitierungen bewusst unvollständig. Eine vollständige Analyse würde den Rahmen

der vorliegenden Arbeit sprengen. Stattdessen wird ein Überblick über verschiedenartige Informationen gegeben, die entweder nativ oder durch Erweiterungen im Ereignisprotokoll aufgenommen werden können. Zusätzlich werden auch Informationen diskutiert, die nicht aus einem Ereignisprotokoll extrahiert werden können. Für weitere Informationen über Eigenschaften und Limitierungen von Ereignisprotokollen sei an dieser Stelle auf die zugehörige Literatur im Bereich des Process Mining (u.a. [6] und konkrete referenzierte Ansätze aus Abschnitt 5.2.3) verwiesen.

### 5.2.3 *Konfiguration: Wahl der Komponenten*

Der in der vorliegenden Arbeit vorgestellte Ansatz zur Translation von Prozessmodellen basiert auf einer Kombination zweier Komponenten – eines Spurgenerators und einer Process-Mining-Technik. Die konkrete Konzeption des Ansatzes für ein bestimmtes Sprachpaar beinhaltet somit insbesondere die Auswahl dieser beiden Techniken. Ziel dieses Abschnitts ist es daher, einen Überblick über die existierenden Komponenten zu geben. Da sowohl Simulatoren als auch Process-Mining-Ansätze nicht immer alle Informationen über den modellierten Prozess protokollieren respektive verarbeiten können, wird die Verfügbarkeit von geeigneten Komponenten nach den fünf bereits beschriebenen Prozessperspektiven (Abschnitt 1.1.2) differenziert. Zusätzlich erlauben einige Sprachen die Unterscheidung verschiedener Ereignistypen. In DPIL sind dies beispielsweise der Start und die Beendigung einer Aufgabe. In Declare und einigen anderen Modellierungssprachen ist diese Unterscheidung dagegen nicht möglich. Deswegen wird dieses Kriterium bei der Komponentenauswahl ebenfalls berücksichtigt. Das Ergebnis ist folglich eine differenzierte Übersicht über paarweise Kombinationen verfügbarer Simulations- und Process-Mining-Techniken, die jeweils ein Translationssystem bilden.

*Wiederzuverwertende Spurgeneratoren*

In Abschnitt 4.1 wird bereits ein Überblick über verfügbare Spurgeneratoren gegeben. Zudem werden diese auch hinsichtlich ihrer Fähigkeiten, perspektivenrelevante Informationen zu protokollieren, untersucht. Tabelle 5.2 zeigt eine reduzierte Version der entsprechenden Übersichtstabelle. Es werden dabei nur die Spurgeneratoren ausgewählt, welche die generierten Ereignisprotokolle in einem standardkonformen Format serialisieren. Aus diesem Grund ist beispielsweise der Spurgenerator für DCR-Graphen nicht in der Tabelle enthalten. Auch der Satz der untersuchten Eigenschaften wird reduziert, da beispielsweise die Fähigkeit eines Spurgenerators, künstliches Rauschen zu produzieren für seine Rolle in einem Translationssystem irrelevant ist. Die Möglichkeit der Konfiguration von Entscheidungen wird ebenfalls ausgeblendet. Der Grund ist, dass diese Einstellungsmöglichkeiten potentiell nützlich sind, eine konkrete Parametrierung aber häufig problematisch ist. Dies wird in Abschnitt 5.2.4 ausführlicher begründet.

In der Tabelle sind die Spurgeneratoren für in Literatur und Praxis häufig anzutreffende Prozessmodellierungssprachen aufgeführt. In jeder Spalte ist vermerkt, ob der Ansatz relevante Informationen für die jeweilige Perspektive oder die Unterscheidung der Ereignistypen protokollieren kann (✓) oder nicht (-). Einige Notationen erlauben es jedoch nicht, Elemente und Zusammenhänge *jeder* Perspektive zu modellieren. Beispielsweise beschränken sich Standard-Petri-Netze rein auf die Anordnung der Prozessschritte und erlauben keine Beziehungen zu

| Sprache | Perspektive | | | | | Ereignistypen: Start/Ende |
|---|---|---|---|---|---|---|
| | Funktionale | Verhaltensorientierte | Datenorientierte | Organisatorische | Operationale | |
| Petri-Netze | | | | | | |
| – [36] | ✓ | ✓ | 0 | 0 | 0 | – |
| – [97] | ✓ | ✓ | 0 | 0 | 0 | – |
| – [173] | ✓ | ✓ | 0 | 0 | 0 | – |
| – [168] | ✓ | ✓ | 0 | 0 | 0 | ✓ |
| CPN [124] | ✓ | ✓ | ✓ | ✓ | – | ✓ |
| Prozessbäume [92] | ✓ | ✓ | 0 | 0 | 0 | ✓ |
| BPMN [34] | ✓ | ✓ | ✓ | – | – | ✓ |
| YAWL [156] | ✓ | ✓ | ✓ | ✓ | 0 | ✓ |
| Declare [41] | ✓ | ✓ | 0 | 0 | 0 | 0 |
| DPIL [Abschnitt 4.2] | ✓ | ✓ | ✓ | ✓ | – | – |

Tabelle 5.2: Verfügbare Translationskomponenten: Spurgeneratoren

Ressourcen, Daten oder Software-Werkzeugen. In diesem Fall ist es im Kontext der Translation eines solchen Prozessmodells die Protokollierung dieser Informationen weder möglich noch notwendig (0).

Wie die Tabelle zeigt, ist lediglich die Protokollierung künstlicher Prozessinstanzen im Falle von BPMN problematisch, da diese weder organisatorische noch operationale Informationen berücksichtigt. Ein möglicher Grund ist, dass in BPMN die involvierten Akteure oder Rollen als Pools und Swimlanes modelliert werden, die jedoch nach dem aktuellen BPMN-Standard keine Semantik haben. Der operationale Aspekt hat dagegen in der Process-Mining-Gemeinschaft bisher wenig Beachtung gefunden. Da existierende Spurgeneratoren fast ausschließlich zur Evaluation von Process-Mining-Techniken eingesetzt werden, ist es nicht verwunderlich, dass der in der Tabelle aufgelistete BPMN-Spurgenerator diese Informationen nicht protokolliert.

*Wiederverwendung des Spurgenerators für DPIL-Modelle*

In der letzten Zeile von Tabelle 5.2 ist der in Abschnitt 4.2 vorgestellte Spurgenerator für DPIL-Modelle eingeordnet. Dies zeigt einerseits die Wiederverwendbarkeit desselben für einen anderen Zweck und andererseits die Ausdrucksmächtigkeit der Sprache DPIL hinsichtlich der Abdeckung der einzelnen Perspektiven.

*Wiederverwendung von Process-Mining-Techniken*

Die Translation erfolgt, wie bereits beschrieben, in zwei Schritten. Im ersten wird für die *Quellsprache* einer der in Tabelle 5.2 aufgelisteten Spurgeneratoren verwendet, um ein Ereignisprotokoll zu erzeugen. Dieses wird im zweiten Schritt durch eine Process-Mining-Technik analysiert, welche daraufhin ein Prozessmodell in der gewünschten *Zielsprache* erzeugen kann. Eine Auswahl der Techniken, die als Komponente zur Umsetzung für diesen zweiten Schritts dienen können, sind in Tabelle 5.3 dargestellt. Diese Auflistung ist nicht vollständig, da lediglich gezeigt werden soll, dass für die meisten der üblichen Prozessmodellierungssprachen ein Process-Mining-Ansatz existiert und somit die beschriebene Translationskette aus Simulator und Mining-Verfahren geschlossen werden kann.

Die Tabelle zeigt, dass viele der Process-Mining-Techniken in der Lage sind, die Informationen aller relevanten Perspektiven zu analysieren, um ein Prozessmodell in der gewünschten Zielsprache zu erzeugen. Die Bewertung für die diesbezügliche Abdeckung der Perspektiven und der Unterscheidung der Ereignistypen erfolgt nach demselben Schema wie in Tabelle 5.2 (✓, – und 0). In diesem Fall bedeutet eine neutrale Bewertung (0) jedoch, dass die *Zielsprache* keine Modellierung der jeweiligen Perspektive respektive keine Unterscheidung der Ereignistypen zulässt. Somit ist es beispielsweise weder notwendig noch möglich, datenorientierte

| Sprache | Perspektive | | | | | Ereignistypen: Start/Ende |
|---|---|---|---|---|---|---|
| | Funktionale | Verhaltensorientierte | Datenorientierte | Organisatorische | Operationale | |
| Petri-Netze | | | | | | |
| – Alpha Miner [8] | ✓ | ✓ | 0 | 0 | 0 | – |
| – Inductive Miner [106] | ✓ | ✓ | 0 | 0 | 0 | ✓ |
| – ILP Miner [192] | ✓ | ✓ | 0 | 0 | 0 | ✓ |
| – Flexible Heuristics Miner [191] | ✓ | ✓ | 0 | 0 | 0 | ✓ |
| CPN [157] | ✓ | ✓ | ✓ | ✓ | – | ✓ |
| Prozessbäume [31] | ✓ | ✓ | 0 | 0 | 0 | ✓ |
| BPMN | | | | | | |
| – [43] | ✓ | ✓ | – | – | – | 0 |
| – [52] | ✓ | ✓ | – | ✓ | – | 0 |
| YAWL [31] | ✓ | ✓ | – | – | 0 | ✓ |
| EPC [31] | ✓ | ✓ | – | – | 0 | ✓ |
| (MP-)Declare | | | | | | |
| – [115, 116] | ✓ | ✓ | 0 | 0 | 0 | 0 |
| – [54] | ✓ | ✓ | 0 | 0 | 0 | 0 |
| – [117] | ✓ | ✓ | (✓) | 0 | 0 | 0 |
| – [165] | ✓ | ✓ | ✓ | ✓ | ✓ | ✓ |
| DPIL [166] | ✓ | ✓ | ✓ | ✓ | – | – |

Tabelle 5.3: Verfügbare Translationskomponenten: Process Mining

Zusammenhänge eines Prozesses mit dem *Flexible Heuristics Miner* aus einem Ereignisprotokoll zu extrahieren.

*Besonders geringe Abdeckung im multi-perspektivischen Bereich*

Wie ebenfalls der Tabelle 5.3 entnommen werden kann, weisen besonders die ausdrucksstärkeren imperativen Prozessmodellierungssprachen, also BPMN, YAWL und EPC eine mangelhafte Unterstützung der Prozessperspektiven auf. Im Gegensatz zu Petri-Netzen, Prozessbäumen und den deklarativen Prozessmodellierungssprachen ist die Popularität dieser Sprachen in der Process-Mining-Gemeinschaft eher gering. Zudem wird für BPMN die Erschließung der Funktionalen und der Verhaltensorientierten Perspektive mittels eines der aufgelisteten Algorithmen zur Extraktion von Petri-Netzen durchgeführt. Als einzige Neuerung wird zusätzlich eine Rollenpartitionierung durchgeführt, um Informationen über die Organisatorische Perspektive zu erhalten. Im Falle der Process-Mining-Technik für YAWL- und EPC-Modelle wird für das Kontrollflussgerüst auf den *Evolutionary Tree Miner* zurückgegriffen, welcher zunächst einen Prozessbaum erzeugt. Dieser wird dann über einen nachträglichen Konvertierungsschritt in die gewünschten Zielsprache überführt. Daher sind die Möglichkeiten der Informationsextraktion aus dem Ereignisprotokoll an die Ausdrucksmächtigkeit der Prozessbäume gebunden. Das erklärt, warum die Datenorientierte und die Organisatorische Perspektive nicht von den Process-Mining-Techniken für EPC- und YAWL-Modelle berücksichtigt werden.

*Gute Kontrollflussunterstützung*

Kombiniert man die Informationen beider Tabellen, dann stellt sich heraus, dass die Funktionale und die Verhaltensorientierte Perspektive sowohl von Seiten der Spurgeneratoren als auch von allen Process-Mining-Techniken unterstützt werden. Diese Kombination wird in der wissenschaftlichen Gemeinschaft überraschenderweise als Grundgerüst jedes Prozesses aufgefasst, obwohl aus Projekterfahrungen im Wirtschafts- und Industrieumfeld hervorgeht, dass beispielsweise die Reihenfolge der Abarbeitung der einzelnen Prozessschritte wesentlich öfter durch Informationen aus der Datenorientierten Perspektive beeinflusst wird [3, 82, 85].

*Übersicht über Abdeckung durch Kombination beider Komponenten*

Die Kombination der Informationen aus Tabelle 5.2 und Tabelle 5.3 ist in Tabelle 5.4 visualisiert. In selbiger beschreibt eine Zelle jeweils die Translation aus der Sprache im Zeilenkopf in die Sprache im Zeilenkopf. Die grünen, grauen und roten Symbole beschreiben die Berücksichtigung der einzelnen Prozessperspektiven

| ↬ | PN | CPN | PT | BPMN | YAWL | EPC | Declare | MP-Declare | DPIL |
|---|---|---|---|---|---|---|---|---|---|
| PN | | F V D<br>O T E | F V D<br>O T E | F V D<br>O T E | F V D<br>O T E | F V D<br>O T E | F V D<br>O T E | F V D<br>O T E | F V D<br>O T E |
| CPN | F V D<br>O T E | | F V D<br>O T E | F V D<br>O T E | F V D<br>O T E | F V D<br>O T E | F V D<br>O T E | F V D<br>O T E | F V D<br>O T E |
| PT | F V D<br>O T E | F V D<br>O T E | | F V D<br>O T E | F V D<br>O T E | F V D<br>O T E | F V D<br>O T E | F V D<br>O T E | F V D<br>O T E |
| BPMN | F V D<br>O T E | F V D<br>O T E | F V D<br>O T E | | F V D<br>O T E | F V D<br>O T E | F V D<br>O T E | F V D<br>O T E | F V D<br>O T E |
| YAWL | F V D<br>O T E | F V D<br>O T E | F V D<br>O T E | F V D<br>O T E | | F V D<br>O T E | F V D<br>O T E | F V D<br>O T E | F V D<br>O T E |
| EPC | – | – | – | – | – | | – | – | – |
| Declare | F V D<br>O T E | F V D<br>O T E | F V D<br>O T E | F V D<br>O T E | F V D<br>O T E | F V D<br>O T E | | F V D<br>O T E | F V D<br>O T E |
| MP-Declare | F V D<br>O T E | F V D<br>O T E | F V D<br>O T E | F V D<br>O T E | F V D<br>O T E | F V D<br>O T E | F V D<br>O T E | | F V D<br>O T E |
| DPIL | F V D<br>O T E | F V D<br>O T E | F V D<br>O T E | F V D<br>O T E | F V D<br>O T E | F V D<br>O T E | F V D<br>O T E | F V D<br>O T E | |

Tabelle 5.4: Verfügbare Translationssysteme differenziert nach Funktionaler (F), Verhaltensorientierter (V), Organisatorischer (O), Datenorientierter (D) und Operationaler (T) Perspektive sowie Unterscheidung zwischen Ereignistypen (E)

und damit letztlich den zu erwartenden Informationsverlust durch die Translation. Ein Informationsverlust kann hierbei auf zwei Arten entstehen: *(i)* Entweder beinhaltet die Zielsprache keine Mittel zur Modellierung der Zusammenhänge der jeweiligen Perspektive/Ereignisunterscheidung (graues Symbol) oder *(ii)* der Translationsansatz ist nicht in der Lage, diese Informationen adäquat zu übersetzen (rotes Symbol). Zusätzlich wird ein graues Symbol auch dann vergeben, wenn die *Quellsprache* die jeweilige Perspektive oder die Unterscheidung der Ereignistypen nicht unterstützt. Beispielsweise beschreibt die Zelle in der Zeile *MP-Declare* und der Spalte *CPN* die Übersetzung eines MP-Declare-Modells in die Sprache CPN. Dabei wird folglich ein Spurgenerator für MP-Declare und eine Process-Mining-Technik für CPN-Modelle verwendet. Die Tabellenzelle beschreibt, inwiefern mit dieser Technik-Kombination alle relevanten Inforationen in der Übersetzung berücksichtigt werden können. Die Funktionale und die Verhaltensorientierte Perspektive werden sowohl von Spurgenerator als auch Process-Mining-Technik unterstützt. Folglich ist das Translationssystem insgesamt in der Lage, diese Perspektiven zu berücksichtigen. Die Datenorientierte, Organisatorische und Operationale Perspektive werden jedoch vom Spurgenerator nicht unterstützt. Genauer gesagt existiert für MP-Declare nur der rein kontrollflussbasierte Declare-Simulator, was die Ursache für den Informationsverlust darstellt. Die Unterscheidung der Ereignistypen ist nicht möglich, was ebenfalls auf Limitierungen des Spurgenerators zurückzuführen ist. Damit sind im Allgemeinen auch alle übrigen Translationen mit MP-Declare als Quellsprache verlustbehaftet.

*Unterscheidung der Limitierung durch Translationsansatz oder Sprachpaar*

Die Unterscheidung zwischen fehlenden Möglichkeiten der Abbildung irrelevanter und relevanter Informationen erlaubt eine differenziertere Betrachtung der Güte des hier vorgestellten Translationsansatzes. Gestatten Ziel- oder Quellsprache die Modellierung von Beziehungen einer bestimmten Prozessperspektive nicht, dann ist das keine Limitierung des Translationsansatzes, sondern ein Unterschied in der Ausdrucksmächtigkeit des Sprachpaares. Falls dagegen beide Sprachen die Modellierung der entsprechenden Perspektive erlauben, diese jedoch von Spurgenerator und/oder Process-Mining-Verfahren nicht unterstützt wird, dann bedeutet dies eine Einschränkung, die durch die hier vorgestellte Translationstechnik verursacht

wird. Damit stellt der im aktuellen Abschnitt diskutierte Translationsansatz für Prozessmodelle ohne jegliche Notwendigkeit der expliziten Definition von Translationsregeln für rund 50% der Sprachpaare eine adäquate Übersetzungsmöglichkeit dar. Für den Großteil der verbleibenden 50% möglicher Sprachpaarungen ist es zumindest möglich, die Informationen der Funktionalen und der Verhaltensorientierten Perspektive zu übersetzen. Die einzige Ausnahme ist hierbei jede Paarung mit EPC als Quellsprache, wobei die Ursache darin liegt, dass für EPC-Modelle aktuell keine Spurgeneratoren existieren.

*Auswirkungen der Nichtunterstützung von Perspektiven auf unterstützte*

Eine Frage, die unweigerlich aufkommt, wenn bei der Translation für ein konkretes Sprachpaar *nicht* alle Perspektiven unterstützt werden ist, welche Auswirkungen dies auf die Korrektheit der Abbildung der anderen Perspektiven hat. Die Antwort darauf ist: Keine! Hintergrund ist, dass die verfügbaren Spurgeneratoren vielleicht nicht alle im Modell enthaltenen Informationen protokollieren, diese aber implizit berücksichtigen. Ein Beispiel dafür ist der Spurgenerator für MP-Declare-Modelle, der identisch ist mit dem Spurgenerator für rein kontrollflussbasierte Declare-Modelle. In MP-Declare können perspektivenübergreifende Regeln formuliert werden, wie z.B.: „Wenn Aktivität A ausgeführt wurde und der Wert der Variable v gleich 1 ist, dann muss vor dem Prozessende irgendwann die Aktivität B ausgeführt werden." Eliminiert man die Datenorientierte Perspektive und damit die eben genannte Regel, dann produziert der Spurgenerator sowohl Ausführungsspuren, in denen B irgendwann auf A folgt, als auch solche, wo dies nicht der Fall ist. Das verletzt die beschriebene MP-Declare-Regel in keiner Form. Lediglich die Gründe sind nicht mehr erkennbar, die das Auftreten von B nach A bedingen.

*Unterschiedlicher Grad der Unterstützung einzelner Perspektiven*

Eine weitere Fragestellung ist, wie die Bewertung für Perspektiven erfolgt, die sowohl von den beiden fraglichen Prozessmodellierungssprachen als auch vom zugehörigen Translationssystem unterstützt werden, wobei eine Sprache jedoch nicht alle Konstrukte der anderen Sprache abbilden kann. Hierbei können zwei Fälle unterschieden werden: *(i)* Die Quellsprache unterstützt weniger Konstrukte als die Zielsprache oder *(ii)* genau umgekehrt. In erstem Falle wird die Translation als möglich (grün) eingestuft, falls die Quellsprache *prinzipiell* Konstrukte zur Modellierung der betroffenen Perspektive beinhaltet. Gleiches gilt für den zweiten Fall, in dem die Zielsprache wenige ausdrucksstark ist. Die Begründung dafür ist, dass mit Tabelle 5.4 nicht die Ausdrucksmächtigkeit der jeweiligen Sprachpaare analysiert werden sollte, sondern das Erfolgspotential einer auf Simulation- und Process-Mining basierenden Translationstechnik.

Ziel des aktuellen Abschnitts ist es, durch Kombinationen aus jeweils einem Spurgenerator und einem Process-Mining-Werkzeug Translationssysteme für verschiedene Paarungen an Prozessmodellierungssprachen bereitzustellen. Eine solches Vorgehen kann den Verlust von Informationen nach sich ziehen. Allerdings kann dieser Verlust bereits durch die Wahl der Zielsprache aber auch durch das Translationssystem selbst verursacht werden. Für 50% der gebräuchlichsten Sprachpaarungen stellt das vorgestellte Translationsprinzip eine adäquate Möglichkeit zur Übersetzung dar, was anhand der Abdeckung der einzelnen Prozessperspektiven sowie der Unterscheidungsmöglichkeit ersichtlich wird. Etwaige Unterschiede in der Ausdrucksstärke des konkreten Sprachpaares werden dabei nicht als Limitierung des Translationssystems angesehen.

### 5.2.4 *Konfiguration: Parametrierung der Komponenten*

Im vorherigen Abschnitt werden die Wahlmöglichkeiten für die zwei Komponenten des Translationssystems beschrieben. Für alle diese Komponenten wird eine Parametrierung benötigt, was im aktuellen Abschnitt diskutiert wird. Dabei wird herausgestellt, dass für die Parametrierung kein einheitliches Schema abgeleitet werden kann. Der Grund ist, dass sich besonders die Charakteristika der einzelnen Process-Mining-Techniken stark unterscheiden. Daher wird lediglich eine allgemeine Vorgehensweise zur Identifikation der geeigneten Parameterwerte sowie eine Überprüfungsmöglichkeit für die getroffene Wahl vorgestellt.

*Harmonie der Komponenten*

Die beiden Komponenten des Translationssystems müssen in allen Fällen einzeln konfiguriert werden, sind aber nicht voneinander unabhängig. Eine geeignete Konfiguration des Spurgenerators sorgt dafür, dass nach Möglichkeit die Informationen über mögliche Prozessausführungsspuren *vollständig* im künstlich erzeugten Ereignisprotokoll festgehalten sind. Die Konfiguration der Process-Mining-Technik sollte so gewählt werden, dass die im Protokoll explizit enthaltenen Informationen korrekt interpretiert werden sowie die implizit enthaltenen Informationen abgeleitet werden können. Da die Einstellung des Spurgenerators stark von den Charakteristika des verwendeten Process-Mining-Werkzeugs abhängig ist, werden zunächst die relevanten Eigenschaften letzterer erläutert.

#### 5.2.4.1 *Parametrierung der Process-Mining-Komponente*

*Relevanz der Konfigurationsmöglichkeiten*

Die zentralen problematischen Eigenschaften von Process-Mining-Werkzeugen [6, S. 195ff] und daraus resultierende Konfigurationsmöglichkeiten sind nachfolgend aufgelistet und bezüglich ihrer Relevanz für das Translationsproblem bewertet:

- *Rauschempfindlichkeit:* Prozessverläufe, die in der Realität nur sehr selten auftreten, gelten in der Literatur nicht als *typisches* Prozessverhalten. In vielen Fällen kann die sogenannte Rauschempfindlichkeit der Werkzeuge so konfiguriert werden, dass sie diese Verläufe auszublenden. (irrelevant)
- *Vollständigkeitsannahme:* Viele Process-Mining-Techniken basieren auf der Annahme, dass die gegebene Datengrundlage *vollständig* ist. (relevant)
- *Ausschluss von (unbeschränkten) Schleifen:* Schleifen führen beispielsweise zu Problemen, wenn der Algorithmus von einer Blockstruktur des Zielmodells ausgeht. (relevant)
- *Ausschluss von OR-Verzweigungen:* Inklusiv-Oder-Entscheidungen werden von vielen Process-Mining-Werkzeugen nicht unterstützt. (relevant)
- *Ausschluss von nicht-freien Entscheidungen:* Verknüpft man in einem imperativen Modell mehrere Verzweigungen, dann kann es vorkommen, dass diese sich gegenseitig beeinflussen. Einige Algorithmen sind unfähig, alle Formen dieser Wechselwirkungen im Modell abzubilden. (relevant)
- *Ausschluss von Hierarchien:* Die meisten Process-Mining-Verfahren sind unfähig, hierarchische Prozessmodelle durch die Ableitung von Subprozessen zu extrahieren. (relevant)

*Rauschempfindlichkeit*

Die *Rauschempfindlichkeit* ist in der Literatur [6] eine zentrale Problematik bei der Anwendung von Process-Mining-Techniken auf historische Ereignisprotokolle (Abschnitt 3.3.2). Es wird davon ausgegangen, dass es in einigen Fällen gewünscht sei, ausschließlich die „typischen" Prozessverläufe mittels Process-Mining in einem Modell abzubilden. Tritt im Ereignisprotokoll jedoch auch das „nicht-typische" Verhalten auf, dann steht das Mining-Verfahren vor der Herausforderung, unterscheiden zu müssen, welche Verläufe typisch und welche nicht typisch sind. Der Alpha-Algorithmus ist beispielsweise nicht in der Lage dazu, diese Entscheidung zu treffen. Ein Gegenbeispiel ist der Flexible Heuristics Miner. Diese Eigenschaft ist für den Einsatz der Process-Mining-Technik als Komponente eines Translationssystems nicht relevant. Der Grund ist, dass alle Ereignisprotokolle durch Spurgeneratoren erzeugt werden. Diese wiederum interpretieren dazu ein existierendes Prozessmodell. Ist darin nicht-typisches Verhalten modelliert, dann sollte dieses bei einer Übersetzung in eine andere Sprache, im Sinne der Erhaltung der Semantik, auch im Zielmodell repräsentiert werden. Der Grund ist, dass davon ausgegangen werden muss, dass das Quellmodell einen Prozess adäquat abbildet. Ist dies nicht der Fall, so handelt es sich um einen Fehler bei der Erstellung des Quellmodells und folglich sollte der Fehler auch in diesem korrigiert werden. Dementsprechend sollten die verwendeten Process-Mining-Technologien diesbezüglich keine Filterung vornehmen, weswegen die Rauschempfindlichkeit der Process-Mining-Techniken als irrelevant eingestuft wird.

*Vollständigkeitsannahme*

Ein weiteres Problem stellt die *Vollständigkeitsannahme* (engl. *completeness assumption*) vieler Process-Mining-Techniken dar (Abschnitt 5.2.2). Beispielsweise geht der Alpha-Algorithmus davon aus, dass, falls eine Aktivität B unmittelbar auf eine Aktivität A folgen darf, diese Sequenz auch mindestens einmal im Ereignisprotokoll auftritt. Andere Algorithmen nehmen sogar an, dass jede mögliche *Ausführungsspur* im Protokoll enthalten sein muss. Letzteres wird als *starke Vollständigkeitsannahme* (engl. *strong completeness assumption*) bezeichnet. Algorithmen mit einer sehr liberalen Vollständigkeitsannahme tendieren zur Erzeugung von Modellen, die zu schlecht auf die tatsächlichen Ausführungsspuren eingestellt sind. Algorithmen mit einer zu starken Vollständigkeitsannahme produzieren dagegen tendenziell eine Überanpassung an das konkrete Ereignisprotokoll, sodass bis dahin ungesehene Prozessverläufe zu oft ausgeschlossen werden. Diese Eigenschaft ist auch im Kontext der Translation wichtig. Enthält ein imperatives Prozessmodell Verzweigungen oder erlaubt ein deklaratives diese implizit, dann entscheiden Spurgeneratoren meistens auf Basis von Wahrscheinlichkeitsverteilungen, auf welchem Pfad die Simulation fortgesetzt wird. Dadurch kann in den meisten Fällen nicht garantiert werden, dass alle erlaubten Pfade im Ereignisprotokoll festgehalten sind. Das Problem der Vollständigkeit ist somit die größte Herausforderung des hier vorgestellten Translationsansatzes.

*Ausschluss von Schleifen*

Komplexere Konstruktionen wie Schleifen und Inklusiv-Oder-Entscheidungen sind auch im Kontext der Translation problematisch. *Schleifen* führen besonders dann zu Problemen, wenn der Algorithmus von einer Blockstruktur des Zielmodells ausgeht. Beispielsweise ist der Alpha-Algorithmus auch hier ein Problemfall, da er Schleifen über einen oder zwei Schritte nur in einem Nachbearbeitungsschritt im Prozessmodell repräsentieren kann. XOR-Verzweigungen und

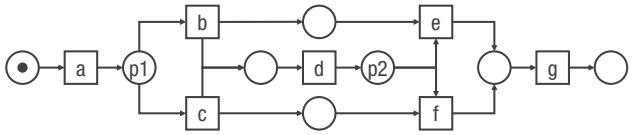



Abbildung 5.7: Petri-Netz mit nicht-freiem Entscheidungspunkt in p2

*Ausschluss von OR-Verzweigungen*

AND-Verzweigungen werden von den meisten Process-Mining-Techniken unterstützt. Das trifft auf *OR-Verzweigungen* und zugehörige -Zusammenführungen jedoch nicht zu. Der Grund ist, dass die meisten Mining-Verfahren für imperative Prozessmodelle zumindest intern auf der Petri-Netz-Semantik basieren, die jedoch kein Inklusiv-Oder beinhaltet. In diesen Fällen ist es dennoch möglich, dass die konstruierten Prozessmodelle bezüglich des Ausführungsverhaltens semantisch korrekt sind, da sich OR-Verzweigungen durch eine Kombination aus XOR- und AND-Verzweigungen nachbilden lassen. Allerdings erhöht dies die Komplexität der resultierenden Prozessmodelle.

*Abhängigkeiten zwischen Entscheidungspunkten*

Nicht an jedem Entscheidungspunkt ist die Wahl des Pfades zur Fortsetzung des Prozesses unabhängig von den vorherigen Entscheidungen. Ein Beispiel für eine nicht-freie Entscheidung ist in dem in Abbildung 5.7 dargestellten Petri-Netz erkennbar. Die Entscheidung im Platz $\mathbf{p2}$ kann nicht frei getroffen werden. Wurde in $\mathbf{p1}$ der obere Pfad gewählt, so muss nach $\mathbf{p2}$ die Transition $\boldsymbol{e}$ aktiviert werden. Wurde dagegen nach $\mathbf{p1}$ die Transition $\mathbf{c}$ aktiviert, so ist die spätere Aktivierung von Transition $\mathbf{f}$ obligatorisch. Nicht jeder Process-Mining-Algorithmus ist in der Lage, derartige Abhängigkeiten zu erkennen. Da diese aber im Quellprozessmodell enthalten sein können (Abbildung 5.7), ist dieses Problem ebenfalls im Kontext der Translation relevant. Einige Algorithmen, wie beispielsweise der Alpha-Algorithmus sind nicht in der Lage, diese Abhängigkeiten zu rekonstruieren, falls die fraglichen Verzweigungen nicht unmittelbar aufeinanderfolgen. Andere Algorithmen, wie beispielsweise der Flexible Heuristics Miner behandeln diese Problematik explizit.

*Keine Translation von Subprozessen*

Eine letzte Problematik liegt in der Identifikation von Subprozessen. Ein Ereignisprotokoll enthält gewöhnlich keinerlei Informationen, zu welchem Subprozess eine Aktivität gegebenenfalls zugehörig ist. Wie die Definition eines Ereignisprotokolls in Abschnitt 3.3 zeigt, werden lediglich die drei Ebenen Protokoll, Ausführungsspur und Einzelereignis dokumentiert. Folglich ist es nicht möglich, Subprozesse des Quellmodells im Ereignisprotokoll zu kodieren (Abschnitt 5.2.2).

Alle der oben genannten Probleme sind bereits aus dem Bereich des Process-Mining bekannt und resultieren dementsprechend nicht aus der Kombination mit einem Spurgenerator zum Zweck der Translation von Prozessmodellen. Aus diesem Grund wird an dieser Stelle auch nicht auf konkrete Lösungsvorschläge für die genannten Probleme eingegangen. Stattdessen sei hier auf die referenzierte Literatur für die einzelnen Process-Mining-Techniken verwiesen, in welcher mögliche Parametrierungen zur Bewältigung der Problemfälle beschrieben werden.

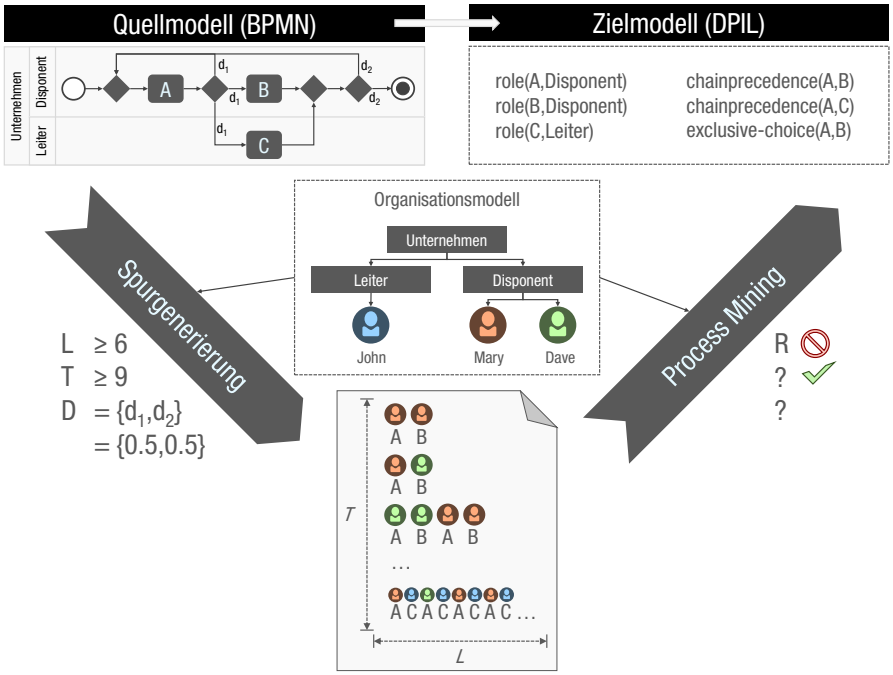



Abbildung 5.8: Beispielkonfiguration für die Translation BPMN-Modell → DPIL-Modell

### 5.2.4.2 *Parametrierung der Spurgenerator-Komponente*

*Parametrierung des Spurgenerators*

Ein Teil der im vorherigen Abschnitt beschriebenen Probleme kann durch die gezielte Konfiguration des jeweiligen Spurgenerators praktisch abgemildert werden. Es ist jedoch für alle Probleme im Allgemeinen nicht möglich, eine Garantie abzugeben, dass sich selbige nicht im resultierenden Prozessmodell materialisieren. Die Spurgeneratoren haben im Wesentlichen zwei Möglichkeiten der Einflussnahme:

1. Ist die Anzahl möglicher unterschiedlicher Prozessausführungsspuren endlich, dann kann jede erlaubte Prozessausführungsspur im Ereignisprotokoll enthalten sein.

2. Falls das beispielsweise aufgrund unbegrenzter Schleifen nicht möglich ist, kann die Länge der zu generierenden Prozessausführungsspuren künstlich nach oben begrenzt werden. Dadurch kann zumindest innerhalb dieser Begrenzung Vollständigkeit erreicht werden.

*Wichtigste Parameter: Länge und Anzahl der Spuren*

In jedem Fall kann nur eine endliche Anzahl an Spuren im Ereignisprotokoll festgehalten werden. Es werden folglich zwei grundlegende Parameter gesucht: Die *maximale Spurlänge* $\mathbf{L}$ und die *Anzahl der Spuren* $\mathbf{T}$.

Der Definition eines Ereignisprotokolls (Abschnitt 3.3) ist zu entnehmen, dass die Ausführung einer Prozessaktivität durch eine bestimmte Ressource mittels eines bestimmten Software-Werkzeugs und unter Verwendung oder Produktion bestimmter Daten in einem einzelnen Ereignis protokolliert werden. Die unterschiedlichen Assoziationen zwischen allen genannten Entitäten kann folglich in einem Ereignisprotokoll bereits dann abgebildet werden, indem pro Ausführungsspur jeweils nur eine konkrete Konstellation derselben protokolliert wird. Damit wäre der Parameter $\mathbf{L}$ immer 1. Da das Prozessmodell jedoch nicht nur diese Assoziationen,

sondern auch die *Reihenfolge des Auftretens* dieser Assoziationen – im einfachsten Fall die Reihenfolge der Bearbeitung einzelner Aktivitäten – beschreibt, ist die maximale Länge $L$ einer vom Prozessmodell erlaubten Prozessausführungsspur ausschließlich von der Anzahl möglicher Prozessschritte pro Instanz abhängig. Die Wahrscheinlichkeit, dass sich innerhalb des kompletten Protokolls alle Assoziation materialisieren, ist dagegen von der Länge des Protokolls, also der Anzahl der enthaltenen Prozessausführungsspuren $T$, abhängig. Im Sinne eines generischen Vorgehensmodells zur Bestimmung dieser Parameter verwenden die nachfolgenden Prinzipien so wenig Wissen über und aus dem gegebenen Prozessmodell.

*Faustregel für Untergrenze der Spurlänge*

Für die Bestimmung von $L$ lässt sich eine Faustregel ableiten, die lediglich auf der Anzahl $|A|$ der Aktivitäten im Prozessmodell beruht (Gleichung 27).

$$L \geqslant 2|A| \tag{27}$$

*Spurlänge entspricht der doppelten Anzahl der modellierten Aktivitäten*

Diese Faustregel nutzt keinerlei Informationen über die vom Prozessmodell gestatten Sequenzen von Aktivitäten. Sie basiert stattdessen auf der Grundlage, dass die längste Folge von unterschiedlichen Aktivitäten identisch mit der im Modell enthaltenen *Anzahl* derselben ist ($|A|$). Verdoppelt man diese Zahl, so sind auch Ketten inbegriffen, in denen eine, zwei bis hin zu $|A|$ Aktivitäten wiederholt werden können. Damit können auch Prozessverläufe berücksichtigt werden, in denen die längstmögliche Kette an unterschiedlichen Aktivitäten wiederholt wird. Das bedient eine eventuelle Vollständigkeitsannahme der auf den resultierenden Ereignisprotokollen aufbauenden Process-Mining-Technik. In Abbildung 5.8 ist ein BPMN-Modell dargestellt, welches in die deklarative Prozessmodellierungssprache DPIL übersetzt werden soll. Nach der oben gegebenen Berechnungsvorschrift ergibt sich mit $|A| = 3$ die Parametrierung $T \geqslant 6$. Für die einfache Vollständigkeitsannahme des Alpha-Algorithmus ist es beispielsweise ausreichend, wenn jede Paarung direkt aufeinanderfolgender Aktivitäten, die vom Modell gestattet ist, auch im Ereignisprotokoll festgehalten wird. Diese sind im hier betrachteten Beispiel $(A, A)$, $(A, B)$, $(A, C)$, $(B, A)$ und $(C, A)$. Diese Paare allein bilden jedoch nicht alle eine vollständige Prozessausführungsspur. Beispielsweise sind die Sequenzen $(A, A)$ und $(B, A)$ mit mindestens einer weiteren Aktivität zu kombinieren. Ein Extremfall ist ein BPMN-Modell, welches eine sequenzielle Ausführung der drei Aktivitäten vorschreibt und eine beliebige Wiederholung dieser Kette gestattet. Damit ist die kürzestmögliche Ausführungsspur, welches die Sequenz $(C, A)$ enthält, die Folge $(A, B, C, A, B, C)$. Für das oben gegebene Beispiel ist demnach eine maximale Länge $L = 4$ ausreichend. Diese Abstufung würde allerdings eine Berücksichtigung aller in einem imperativen Prozessmodell enthaltenen Verzweigungen erfordern. In einem deklarativen Modell sind diese Verzweigungen sogar nur implizit kodiert, was eine Ermittlung dieses Parameters erschweren würde. Folgerichtig basiert die oben beschriebene Faustregel ausschließlich auf der Anzahl der im Prozessmodell enthaltenen Aktivitäten[5].

5 Dies stellt scheinbar einen Widerspruch bezüglich der Open World Assumption deklarativer Prozessmodelle dar, welche besagt, dass beliebig viele, nicht im Modell kodierte Aktivitäten völlig wahlfrei ausgeführt werden können. Imperative Modelle gestatten jedoch naturgemäß keine Aktivitäten, welche nicht explizit im Modell modelliert sind. Folglich ist die Transformation eines imperativen in ein deklaratives Modell nur unter Verlust der Open World Assumption möglich, sodass der scheinbare Widerspruch aufgehoben ist.

Die Bestimmung der notwendigen Anzahl an Prozessausführungsspuren ist weit schwieriger zu ermitteln. Spurgeneratoren für imperative Modelle nutzen in der Regel Wahrscheinlichkeiten oder Priorisierungen, um Entscheidungen an den modellierten Entscheidungspunkten des Eingabemodells zu simulieren. Folglich ist die Generierung der Ereignisprotokolle nicht deterministisch. Auch im Falle deklarativer Prozessmodelle wird auf Wahrscheinlichkeiten zurückgegriffen, um bei mehreren möglichen Fortsetzungen eines Prozessverlaufs die jeweils nächste Aktivität zu bestimmen. Eine Ausnahme hierbei bildet die in Abschnitt 4.2 beschriebene Technik zur Spurgenerierung, die durch die Transformation des Simulations- in ein Erfüllbarkeitsproblem und für eine gegebene Größe des Lösungsraums Vollständigkeit und Redundanzfreiheit des Ereignisprotokolls garantiert. Daher ist diese Technik von der Notwendigkeit der nachfolgenden Berechnung ausgenommen.

Um eine Konfiguration des Parameters $T$ festlegen zu können, muss die Anzahl der modellierten Entitäten des Prozesses betrachtet werden. Falls das Prozessmodell auch die Datenorientierte Perspektive beschreibt, dann ist es möglich, dass im Prozessmodell Variablen mit beliebigen Wertebereichen verwendet werden. In diesem Fall ist die Anzahl der möglichen Assoziationen der Entitäten und Datenwerte potentiell unendlich groß. Folglich ist es nicht möglich, die notwendige Anzahl an Prozessausführungsspuren absolut zu bestimmen. Es ist jedoch möglich, eine *Untergrenze* festzulegen, welche die Datenwerte ausblendet und lediglich die Anzahl der möglichen Assoziationen zwischen den Entitäten berücksichtigt. Dabei wird davon ausgegangen, dass im Extremfall pro Ausführungsspur lediglich ein Ereignis protokolliert wird. Damit definiert sich die Untergrenze für die Festlegung des Parameters $T$ nach Gleichung 28.

*Faustregel für Untergrenze der Spuranzahl*

$$T \geqslant |P_1| \times |P_2| \times ... \times |P_a| \tag{28}$$

Dabei bezeichnet die Menge $P_x$ alle Entitäten einer bestimmten Perspektive, im Beispiel aus Abbildung 5.8 also die Menge aller Aktivitäten oder aller organisatorischer Ressourcen. Die Anzahl aller möglichen Kombinationen der Entitäten aller Perspektiven ist folglich identisch mit der Größe des kartesischen Produktes der Größe der Entitätsmengen aller Perspektiven. Somit ergeben sich im Beispiel, ohne Berücksichtigung der BPMN-Swimlanes, für drei Ressourcen und drei Aktivitäten die möglichen Assoziationen $(A, John)$, $(A, Mary)$, $(A, Dave)$, $(B, John)$ etc. Die Anzahl dieser Paarungen, also die Größe des kartesischen Produktes ist in diesem Fall 9. Diese Anzahl ließe sich durch die Berücksichtigung der rollenbasierten Modellvorschriften bezüglich der Assoziation von Aktivitäten und organisatorischen Ressourcen reduzieren. Dies erforderte aber einerseits eine komplexere Analyse des Modells und andererseits ist die oben stehende Definition für den Parameter $T$ eine *untere* Grenze. Diese eröffnet lediglich die *Möglichkeit*, alle relevanten Assoziationen der Prozessentitäten im Ereignisprotokoll anzutreffen. Durch die Verwendung von Wahrscheinlichkeiten zur Modellierung menschlicher Entscheidungen ist das jedoch nicht garantiert. Aus diesem Grund ist anzunehmen, dass die praktische Anwendung der hier vorgestellten Translationstechnik eine signifikant höhere Anzahl an Prozessausführungsspuren erfordert. Da dies jedoch stark vom jeweiligen Spurgenerator und den Anforderungen der gewählten Process-Mining-Technik abhängig ist, wird diese Problematik im konzeptionellen

*Mindestanzahl entspricht kartesischem Produkt der Größe aller Entitätsmengen*

Teil der vorliegenden Arbeit nicht näher betrachtet. Im Evaluationsteil werden jedoch verschiedene Beispielkonfigurationen auf eine Auswahl an Translationsproblemen angewendet und bezüglich der Qualität des Resultats bewertet.

*Konfiguration von Entscheidungen*

Ein letzter Einflussfaktor der Spurgeneratoren auf die Inhalte der künstlichen Ereignisprotokolle sind die bereits mehrfach erwähnten Wahrscheinlichkeitsverteilungen zur Simulation von Entscheidungen auf Basis von Zufallszahlen. Jede Entscheidung kann in der Realität mit unterschiedlicher Häufigkeit getroffen werden, was traditionell mittels einer geeigneten Verteilung approximiert wird (Abschnitt 3.2.3). Die mit diesen Zufallszahlen simulierten Entscheidungen können sowohl datenbasiert sein als auch eine freie Wahl der nächsten Prozessschritte beschreiben. Diese Unterscheidung ist jedoch im Kontext der hier beschriebenen Translation unerheblich, da die Entscheidungspunkte selbst nicht in den als Transfermedium verwendeten Ereignisprotokollen festgehalten werden. Stattdessen werden lediglich die Entscheidungsgrundlage – in Form einer partiellen Prozessausführungsspur – *vor* und die Auswirkungen *nach* der Entscheidung protokolliert. Ein Spurgenerator muss demnach derart konfiguriert werden, dass die Wahrscheinlichkeit der Manifestation beider Entscheidungen im Protokoll maximal ist. Auch hier wird aus Gründen der Generizität und Simplizität des Translationsverfahrens eine Vereinfachung vorgenommen.

Betrachtet man das in Abbildung 5.8 dargestellte BPMN-Modell, dann lassen sich darin zwei Verzweigungen des Kontrollflusses erkennen. Mit dem Parameter T wird eine feste Anzahl an Simulationsdurchläufen festgelegt. Die Gesamtanzahl dieser Durchläufe teilt sich an der ersten Verzweigung auf die nachfolgenden Pfade auf. Eine „50:50"-Aufteilung maximiert die Wahrscheinlichkeit, dass die beiden Aktivitäten B und C im Ereignisprotokoll auftreten. Würde jedoch auf Aktivität B ein weiterer Entscheidungspunkt folgen und auf einem der daraus austretenden Pfade ein weiterer etc., dann verteilen sich diese 50% an Simulationsdurchläufen rekursiv auf jede nachfolgende Entscheidung. Dadurch würde eine eventuelle Aktivität am tiefsten Punkt dieser Entscheidungskette mit einer vergleichsweise geringen Wahrscheinlichkeit in einer Ausführungsspur protokolliert wird. Eine Erhöhung dieser Wahrscheinlichkeit würde allerdings eine Berücksichtigung aller Verzweigungen und Schleifen des Modells erfordern, was dem Anspruch der Simplizität des Ansatzes entgegenstehen würde. Da eine triviale Beziehung zwischen der Wahrscheinlichkeit der Protokollierung einer bestimmten Aktivität und dem Parameter T besteht, ist die Erhöhung von letzterem eine pragmatische Lösung, die auch im Evaluationsteil der vorliegenden Arbeit verwendet wird.

*Problem der lokalen Konfiguration von Entscheidungen*

#### 5.2.4.3 *Automatisierung der Konfiguration*

Die Konfiguration eines Process-Mining-Werkzeugs ist eine Balanceraufgabe zwischen Überanpassung und Unteranpassung des resultierenden Prozessmodells an das gegebene Ereignisprotokoll (Abschnitt 3.4). Die beiden Metriken Fitness und Appropriateness werden üblicherweise verwendet, um den Erfolg der Bewältigung dieser Balanceraufgabe zu messen. Im Kontext des hier vorgestellten induktiven Translationsansatzes können diese Metriken dazu verwendet werden, die Bestimmung einer geeigneten Konfiguration zu automatisieren. Dazu können die Parameter, welche die Toleranz des Mining-Ansatzes bedingen, zunächst zufällig

*Automatisierung der Konfiguration*

eingestellt werden. Bei der in [166] beschriebenen Process-Mining-Technik für DPIL-Modelle sind das die beiden Schwellwerte für Support und Confidence. Nach zufallsbasierter Festlegung dieser Werte, wird die Übersetzung durchgeführt. Nachfolgend muss die Übereinstimmung des resultierenden Modells mit dem Ereignisprotokoll unter Zuhilfenahme der Fitness- und Appropriateness-Metriken gemessen werden. Ist der berechnete Fitnesswert zu niedrig, der für die Appropriateness aber sehr hoch, dann sollte die Toleranz des Process-Mining-Werkzeugs erhöht werden. Im DPIL-Mining bedeutete dies eine Absenkung der Support- und Confidence-Schwellen. Anschließend muss eine erneute Übersetzung und eine erneute Messung der Güte des Ergebnisses durchgeführt werden. Ist die Konfiguration des Process-Mining-Werkzeugs nun zu tolerant ausgefallen, dann stellt sich ein vergleichsweise hoher Fitnesswert aber ein niedriger Wert für die Appropriateness ein. Folglich sollte die Extraktion des Prozessmodells weniger tolerant erfolgen.

Die automatische Korrektur der Toleranz erfordert eine konkrete Anpassung der entsprechenden Schwellwerte. Eine Möglichkeit wäre, die Schwellwerte schrittweise durch Addition oder Subtraktion eines konstanten Werts anzupassen. Das gefährdet jedoch das Terminierungskriterium des Anpassungsalgorithmus. Addiert man die Konstante zu den Schwellwerten, kann das eine Überanpassung des Modells zur Folge haben. Subtrahiert man dann dieselbe oder eine andere Konstante, dann kann das Resultat wiederum eine Unteranpassung darstellen. Folglich sollte die Anpassung zwar schrittweise aber unter Zuhilfenahme *dynamischer* Korrekturwerte vorgenommen werden. Da sich die entsprechend einzustellenden Parameter von Process-Mining-Verfahren zu Process-Mining-Verfahren unterscheiden, ist es hier nicht das Ziel, für jedes dieser Verfahren einen *individuellen* Algorithmus zur Anpassung der Schwellen anzugeben. Stattdessen soll an dieser Stelle auf einen generischen Algorithmus verwiesen werden, welcher unter dem Namen *Simulated Annealing* [59] im Kontext von Optimierungsproblemen bekannt ist. Simulated Annealing ist ein heuristisches Optimierungsverfahren, welches den physikalischen Abkühlungsprozess von vorher erhitztem Metall zum Vorbild hat. Beim Abkühlen ordnen sich die Atome nach und nach so an, dass ein energetisch günstiger und damit stabiler Zustand erreicht wird. Dabei ist es auch zulässig, dass zwischenzeitlich energetisch ungünstige Zustände angenommen werden. Mit sinkender Temperatur des Metalls sinkt die Wahrscheinlichkeit dafür aber gegen 0. Auf das Optimierungsproblem der Toleranz des Process-Mining-Algorithmus transferiert, bedeutete dies wieder eine schrittweise Anpassung der Konfiguration. Diese kann mit einer Konstante oder, im günstigeren Fall, mit einem dynamischen Wert vorgenommen werden. Die Wahrscheinlichkeit, dass eine Anpassung stattfindet, sinkt aber mit der zunehmenden Zahl an Anpassungsiterationen – was dem Abkühlungsprozess entspricht und den Namen Simulated Annealing motiviert. Simulated Annealing garantiert eine Terminierung nach einer festgelegten Anzahl an Anpassungsschritten. Auf diesem Weg lässt sich eine für das gegebene Quellprozessmodell und das gewählte Process-Mining-Verfahren optimale Übersetzungslösung ermitteln.

*Simmulated Annealing*

Ziel dieses Abschnitts über die Konfiguration der Translationskomponenten ist vor allem die Identifikation der damit verbundenen Herausforderungen und möglicher Bewältigungsstrategien. Eine wesentliche Erkenntnis ist dabei, dass die Hauptprobleme im Bereich der Vollständigkeit der durch Spurgeneratoren erzeug-

ten Ereignisprotokolle liegen. Somit sind für die Mining-Komponente nach Möglichkeit Verfahren vorzuziehen, die lediglich auf der einfachen Vollständigkeitsahnnahme aufbauen, also lediglich eine vollständige Menge aller geordneten Paare an Aktivitäten benötigen. Über eine angegebene Berechnungsvorschrift können untere Schranken für Länge und Anzahl der zu erzeugenden Ausführungsspuren ermittelt werden. Mit Techniken, wie beispielsweise dem Simmulated Annealing, kann der Konfigurationsprozess automatisiert werden, was jedoch nicht garantiert, dass die resultierende Konfiguration dem globalen Optimum entspricht.

### 5.2.5 *Ansatzspezifische Annahmen*

Es besteht aktuell ein großer Mangel an geeigneten Translationstechniken für Prozessmodelle. Die Entwicklung von Translationstechniken für jedes einzelne Paar konzeptueller Prozessmodellierungssprachen ist aufwendig. Das in den vorangegangenen Abschnitten beschriebene Prinzip ist von einer konkreten Sprachpaarung unabhängig und ist dementsprechend generisch. Diese Generizität geht jedoch mit folgenden, anschließend detaillierter betrachteten Annahmen für die Anwendbarkeit dieses Prinzips einher:

- Für die Übersetzung eines Prozessmodells stehen sowohl ein Spurgenerator für die Quellsprache, als auch ein Process-Mining-Werkzeug für die Zielsprache zur Verfügung.
- Ein eventueller Informationsverlust wird akzeptiert oder vom Nutzer nachträglich kompensiert.
- Das Quellmodell ist syntaktisch und inhaltlich fehlerfrei oder es existieren korrekte Ereignisprotokolle für die Ausführung des modellierten Prozesses.
- Die Translation muss weder inkrementell sein, noch müssen Beziehungen zwischen Quell- und Ziel explizit dokumentiert werden.

*Existenz geeigneter Spurgeneratoren und Process-Mining-Techniken*

Eine grundlegende Motivation für die Entwicklung eines generischen Translationsverfahrens ist der technische Aufwand der Definition von Translationsregelsätzen für alle beliebigen Paarungen von Prozessmodellierungssprachen. Der hier vorgestellte Ansatz basiert auf der Kombination der Prinzipien der Spurgenerierung und des Process Minings. Die Entwicklung von Umsetzungen dieser beiden Prinzipien ist ihrerseits aufwendig. Das in der vorliegenden Arbeit vorgestellte Translationsprinzip baut folglich auf der Annahme auf, dass Spurgeneratoren und Process-Mining-Techniken auch *unabhängig* von ihrer Verwendung als Komponenten eines Translationssystems benötigt und entwickelt werden. Das Prinzip ist demnach nur dann anzuwenden, wenn sowohl ein Spurgenerator für die Quell- und eine Process-Mining-Technik für die Zielsprache existieren.

Die Nutzung von Ereignisprotokollen als Transfermedium bedeutet potentiell einen *Informationsverlust*, da beispielsweise Entscheidungspunkte in selbigen nicht explizit beschrieben werden. Da sich jedoch auch die Ausdrucksmächtigkeit von Quell- und Zielsprache der Translation unterscheiden können, ist die Möglichkeit

und Notwendigkeit zur Abbildung der betroffenen Zusammenhänge des Quellmodells im Zielmodell nicht zwangsläufig gegeben. Aus diesem Grund wird angenommen, dass ein durch das Transfermedium eventuell verursachter Informationsverlust für die gewählte Zielsprache entweder irrelevant ist oder der Nutzer des Translationssystems die fraglichen Informationen manuell nachträgt. Soll die Translation verlustfrei erfolgen, dann sollte auf ein regelbasiertes Übersetzungssystem zurückgegriffen werden.

*Irrelevanz oder Kompensation von Informationsverlust*

Die manuelle Modellierung eines Prozesses und auch die Erstellung des Prozessmodells mittels Process-Mining-Techniken ist fehleranfällig. Eine Translationstechnik für Prozessmodelle ist jedoch per Definition darauf ausgerichtet, für ein gegebenes Quellmodell ein semantisch äquivalentes Prozessmodell in der Zielsprache zu erzeugen. Enthält das Quellmodell *inhaltliche oder syntaktische Fehler*, dann werden erstere in vielen Fällen ins Zielmodell übernommen und letztere blockieren den in dieser Arbeit vorgeschlagenen Translationsansatz. Aus diesem Grund wird angenommen, dass das Quellmodell syntaktisch korrekt ist. Die in den vorherigen Abschnitten beschriebene Technik ist in der Lage, mittels historischer Ereignisprotokolle inhaltliche Veränderungen während der Übersetzung vorzunehmen. Daher wird weiter angenommen, dass das Quellmodell entweder auch inhaltlich korrekt ist oder ein Ereignisprotokoll zur Verfügung steht, welches die Informationen enthält, die eine Process-Mining-Technik benötigt, um die entsprechenden Fehler zu korrigieren.

*Syntaktisch fehlerfreies Quellmodell*

*Inhaltliche Korrektheit des Quellmodells vs. Korrekturinformationen*

Der vorgestellte Ansatz zur Translation von Prozessmodellen arbeitet nicht inkrementell, sondern überschreibt eine eventuell bereits existierende Übersetzung. Es wird daher davon ausgegangen, dass die Übersetzung in der Form ausgeführt werden soll, dass eventuelle Änderungen am Zielmodell *überschrieben* werden dürfen. Weiterhin wird davon ausgegangen, dass Änderungen im Zielmodell nicht auf die zugehörigen Änderungen im Quellmodell zurückverfolgt werden können sollen.

*Nicht-inkrementelle, zurückverfolgbare Translation nicht gewünscht*

Abschließend ist an dieser Stelle zu betonen, dass in diesem Kapitel ein Translations*prinzip* beschrieben wird, nicht aber ein konkretes Translations*system*. Das Prinzip gestattet jedoch ein solches System sukzessive aufzubauen, zu parametrieren und zu verwenden. Dabei ist jedoch zu berücksichtigen, dass es sich um einen induktiven Ansatz handelt, der naturgemäß von der Vollständigkeit und Korrektheit der exemplarischen Datengrundlage abhängig ist. Da diese durch die Verwendung von Spurgeneratoren jedoch künstlich erzeugt wird, ist eine Einflussnahme auf diese Qualitätskriterien möglich. Eine zentrale Schwierigkeit ist hier die Bestimmung von Länge und Anzahl der künstlichen Prozessausführungsspuren. Durch pragmatische Rechenvorschriften können hierfür jedoch zumindest untere Schranken ermittelt werden. Das sich anschließende Process Mining ist der eigentliche, induktive Teil des Prinzips. Die hierfür zur Verfügung stehenden Techniken sind ursprünglich für historische und nicht für künstliche Ereignisprotokolle ausgelegt und ihre Konfiguration ist daher häufig aufwendig. Durch die Verwendung künstlicher Ereignisprotokolle kann dieser Aufwand aber stark eingeschränkt werden, da beispielsweise die Häufigkeit mit der bestimmte Verhaltensmuster protokolliert werden, meist irrelevant ist. Zentral ist dagegen, *dass* diese Verhaltensmuster protokolliert werden. Trotz der diskutierten Lösungsstrategien und Optimierungen kann keinesfalls garantiert werden, dass das hier beschriebene induktive Verfah-

*Abschließende Bewertung*

ren fehlerfreie Ergebnisse liefert. Ein wesentlicher Vorteil liegt aber darin, dass gegenüber des herkömmlichen traditionellen, regelbasierten Translationsprinzips nur ein sehr geringer manueller Aufwand notwendig ist, um für beliebige Sprachpaare ein funktionsfähiges Translationssystem zu entwickeln. Dies wird allerdings dadurch bedingt, dass für das jeweilige Sprachpaar ein Spurgenerator und ein Process-Mining-Werkzeug zur Verfügung steht. Für beide Klassen von Werkzeugen stehen unabhängig vom Übersetzungskontext bereits zahlreiche Vertreter zur Verfügung. Einer Anwendung des Prinzips auf verschiedenste Sprachpaarungen steht demnach nichts im Wege. Allerdings ist dies ohne automatisierte Kontrollmechanismen (Abschnitt 5.2.4.3) nur ratsam, wenn in den Übersetzungsprozess auch eine manuelle Ergebniskontrolle und Nachbesserung durch Modellierungs- und Domänen-Experten integriert werden kann.

Ziel dieses Kapitels ist die Entwicklung eines generischen Prinzips zur Translation von Prozessmodellen. Dies soll die Abdeckung der Paarungen gängiger Prozessmodellierungssprachen bezüglich verfügbarer Übersetzungssysteme erhöhen. Regelbasierte Übersetzungstechniken sind zwar weitestgehend zuverlässig, bedeuteten aber einen hohen initialen Aufwand bei ihrer Entwicklung. Außerdem ist die dafür notwendige Definition von Transformationsregeln besonders schwierig, wenn ein Übersetzungssystem für ein Paar aus einer imperativen und einer deklarativen Prozessmodellierungssprache entwickelt werden soll (Abschnitt 5.2.1). Das im aktuellen Abschnitt vorgestellte induktive Prinzip basiert auf Kombinationen aus existierenden Spurgeneratoren und Process-Mining-Werkzeugen und ist dadurch generisch. Somit bedeutet die Entwicklung eines funktionsfähiges Übersetzungssystem für ein konkretes Sprachpaar lediglich einen Konfigurationsaufwand. Dies setzt voraus, dass für das Sprachpaar Spurgeneratoren und Process-Mining-Werkzeuge zur Verfügung stehen. Ein wesentlicher Nachteil des induktiven Charakters dieses Prinzips ist, dass die Korrektheit der Übersetzung nicht garantiert werden kann. Allerdings ist eine drastische Reduktion der Fehlerwahrscheinlichkeit möglich, indem eine geeignete Konfigurationen der Spuranzahl und -länge gewählt und eventuelle Rauschfilter verwendeter Process-Mining-Werkzeuge deaktiviert werden. Dies zeigt sich auch bei der späteren beispielhaften Evaluation dieses Prinzips (Abschnitt 8.4.3).

# 6

# GENERIERUNG NATÜRLICHSPRACHLICHER BESCHREIBUNGEN FÜR MULTI-PERSPEKTIVISCHE, DEKLARATIVE PROZESSMODELLE

Eine grundlegende Herausforderung bei der Nutzung von Prozessmodellen als Verständigungshilfe ist, dass der Zielgruppe die verwendete Modellierungssprache bekannt sein muss. Außerdem sind einige dieser Sprachen für verschiedene Typen von Prozessen unterschiedlich gut geeignet. Deklarative Sprachen werden tendenziell für die Modellierung entscheidungsintensiver Prozesse mit einem großen Variantenreichtum an Abläufen verwendet. Um das Erlernen solcher Modellierungssprachen zu erleichtern, um deklarativ modellierte Prozesse auch natürlichsprachlich dokumentieren zu können und um in Ausführungssystemen natürlichsprachliche Benachrichtigungen über relevante Modellteile anbieten zu können, werden in diesem Kapitel Möglichkeiten untersucht, aus deklarativen Prozessmodellen natürlichsprachliche, beschreibende Texte abzuleiten (Anforderung A3).

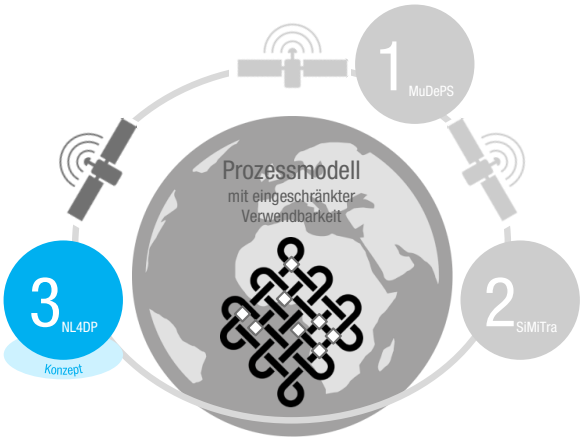



Abbildung 6.1: NL4DP: Generierung natürlichsprachlicher Prozessbeschreibungen für multi-perspektivische, deklarative Prozessmodelle

Nachfolgend (Abschnitt 6.1) werden existierende Ansätze für die Generierung natürlichsprachlicher Texte für Prozessmodelle analysiert. Anschließend (Abschnitt 6.2) wird ein Verfahren zur Generierung natürlichsprachlicher Texte für *multi-perspektivische deklarative* Prozessmodelle beschrieben. Dieses ist einerseits der dritte Kernbeitrag dieser Arbeit (Abbildung 6.1) und wird andererseits zu großen Teilen in Form eines exemplarischen Durchstichs untersucht. Dies ist damit begründet, dass die Generierung natürlichsprachlicher Texte aus zahlreichen Einzelproblemen besteht, für die vielfältige Lösungsstrategien und Konzepte existieren. Fokus der vorliegenden Arbeit ist die Untersuchung der spezifischen Probleme eines solchen Systems für multi-perspektivische, deklarative Prozessmodelle sowie ein *möglicher* Vorschlag für dessen Konzeption.

## 6.1 VERWANDTE ARBEITEN

*Keine vergleichbaren Ansätze für deklarative Prozessmodelle*

Nachfolgend werden existierende Ansätze zur Generierung natürlichsprachlicher Beschreibungen für imperative Prozessmodelle diskutiert und hinsichtlich ihrer Anwendbarkeit auf *deklarative* Prozessmodelle untersucht. Für letztere existieren zum aktuellen Zeitpunkt keine vergleichbaren Techniken. Damit fehlt auch eine adäquate Vergleichsgrundlage für den später vorgestellten Ansatz zur Generierung natürlichsprachlicher Beschreibungen für *deklarative* Prozessmodelle (Abschnitt 6.2).

*Hauptproblem: Fehlende native Anordnung der Modellelemente*

Eine wesentliche Erkenntnis der Analyse der Technik für imperative Prozessmodelle ist, dass sich ein mögliches Prinzip der Natural Language Generation für deklarative Modelle besonders durch die fehlende native Anordnung der Beschreibungsgegenstände von ersterer unterscheidet. Ein weiterer zentraler Unterschied ist die Fokussierung imperativer Prozessmodelle auf die Beschreibung der Reihenfolge aller Prozessschritte. Deklarative Prozessmodelle spezifizieren diese Abfolgen nur implizit; die zu beschreibenden Modellinhalte setzen sich aus einer ungeordneten Menge von Regeln zusammen, die verschiedenste Typen von Abhängigkeiten zwischen Entitäten aller Prozessperspektiven beschreiben können. In Tabelle 6.1 sind die zu bewältigenden Herausforderungen eines NLG-Systems für Prozessmodelle aufgelistet. Darin ist auch ersichtlich, welche davon durch die beschriebenen verwandten Arbeiten bewältigt werden und ob die dafür entwickelten Konzepte in einem NLG-Ansatz für multi-perspektivische, deklarative Prozessmodelle direkt (✓), nach Anpassung (↻) oder gar nicht wiederverwendet (×) werden können.

Derzeit existieren für die Generierung natürlichsprachlicher Prozessbeschreibungen von Prozessmodellen lediglich zwei Ansätze [107, 118] für *imperative* Modelle, welche nachfolgend (Abschnitt 6.1.1 und Abschnitt 6.1.2) vorgestellt und untersucht werden. Bei [107] handelt es sich um eine Technik zur Generierung natürlichsprachlicher Texte aus gegebenen Prozessmodellen der BPMN-Notation. Das Ziel dieser Technik ist es, den Nutzerkreis für Prozessmodelle auf Domänenexperten zu erweitern, indem für ansonsten möglicherweise unverständliche BPMN-Modelle natürlichsprachliche Beschreibungen erzeugt werden. Die Technik wird in [108] erweitert und fokussiert sich dabei auf das Anwendungsgebiet der Validierung von Prozessmodellen. Der zweite nachfolgend vorgestellte Ansatz [118] transformiert ebenfalls BPMN-Modelle in natürlichsprachlichen Text, wobei dieser sich an die Konventionen der *Semantics of Business Vocabulary and Business Rules (SBVR)* der Object Management Group hält und dementsprechend Texte nur in *eingeschränkt*- natürlicher Sprache erzeugen kann.

| Phase | Herausforderung | Konzepte verwandter Ansätze | Übertragbarkeit |
|---|---|---|---|
| Dokumentplanung | Extraktion der Linguistischen Informationen | [107, 108, 118] | ✓ |
| | Linearisierung des Modells | [107, 108], ([118]) | × |
| | Textstrukturierung | [107, 108] | ↻ |
| Mikroplanung | Lexikalisierung | [107, 108, 118] | ↻ |
| | Verfeinerung der Nachrichten | [107, 108] | × |
| Oberflächenrealisierung | Generierung des Zieltexts | [107, 108, 118] | ✓ |
| Flexibilität | Unterschiedliche Charakteristika der Eingabemodelle | [107, 108] | × |

Tabelle 6.1: Herausforderungen bei der Generierung natürlichsprachlicher Texte aus Prozessmodellen [16, 107, 108]

### 6.1.1 *NLG für BPMN-Modelle*

Die in [107] und [108] vorgestellte Technik basiert auf dem in Abschnitt 3.6.2 beschriebenen allgemeinen Pipeline-Modell für NLG-Systeme. Es transformiert demnach schrittweise ein gegebenes BPMN-Modell in natürlichsprachlichen Text in englischer Sprache. Leopold et al. beschreiben auf Basis vorangegangener Untersuchungen eine Grundmenge an Herausforderungen, welche bei der Konzeption eines NLG-Systems für Prozessmodelle bewältigt werden müssen (Tabelle 6.1). Nachfolgend werden alle aufgelisteten Herausforderungen näher erläutert und es wird diskutiert, wie der NLG-Ansatz, welcher in Abbildung 6.2 skizziert ist, ihnen begegnet.

#### 6.1.1.1 *Dokumentplanung*

*Extraktion Linguistischer Informationen*

In der Phase der Dokumentplanung ist eine der Teilaufgaben die *Extraktion* der in Prozessmodellen bereits enthaltenen *Linguistischen Informationen*. Dies beinhaltet vor allem die Lokalisierung und Aufbereitung der Aktivitätsnamen des Prozessmodells. Beispielsweise muss ein Aktivitätsbezeichner „Antrag prüfen" in das Verb „prüfen" und das Nomen „Antrag" zerlegt werden. Ohne diese Zerlegung könnten ausschließlich Sätze wie „Der Nutzer muss die Aktivität *Antrag prüfen* zweimal durchführen" generiert werden. Mit dieser Zerlegung kann das Verb der Aktivitätsbezeichnung als solches in beliebigen Sätzen verwendet werden, was die Erzeugung von Sätzen wie „Der Nutzer muss den Antrag zweimal prüfen" ermöglicht. In [109] beschreiben Leopold et al. Möglichkeiten, diese Dekomposition der Aktivitätsbezeichner durchzuführen, weisen aber gleichzeitig darauf hin, dass die Kürze dieser Beschriftungen durch die Mehrdeutigkeit vieler Wörter problematisch ist. Daher werden auf Basis der Analyse einer Menge an Prozessmodellen aus dem Industrieumfeld nach wiederkehrenden Mustern in der Benennung von Modellelementen wie Aktivitäten und Entscheidungspunkte gesucht. Zusätzlich werden Technologien wie der *Stanford POS Tagger* [98] und *WordNet* [130] zur Erkennung und Disambiguierung von Wortarten eingesetzt (Abbildung 6.2). Da sowohl imperative als auch deklarative Prozessmodelle Aktivitäten modellieren, lassen sich dieselben Dekompositionstechniken, die für die Bezeichner von Aktivitäten imperativer Modelle verwendet werden, auch auf die Bezeichner von Aktivitäten deklarativer Prozessmodelle anwenden.

*Linearisierung des Modells*

Eine weitere Herausforderung der Dokumentplanung ist die *Linearisierung des Modells*. Prozessmodelle beinhalten häufig nicht nur Sequenzen von Aktivitäten, sondern auch Verzweigungen wie Nebenläufigkeit und Entscheidungspunkte. Die Sätze natürlichsprachlicher Texte sind jedoch üblicherweise sequenziell angeordnet. Es ist folglich zu entscheiden, in welcher Reihenfolge die weiterführenden Pfade einer Verzweigung des Prozessmodells im Text beschrieben werden. Die Linearisierung des BPMN-Eingabemodells erfolgt durch eine Transformation des Modells in einen sogenannten *Verfeinerten Prozessstrukturbaum (VPSB)*. Dieser ist eine Hierarchie an Subgraphen des BPMN-Modells und basiert auf der Beobachtung, dass jeder Prozessgraph in eine Hierarchie logisch unabhängiger Subgraphen mit je einem Ein- und Ausgang zerlegt werden kann. Die Wurzel des Baumes ist das vollständige Prozessmodell. Jeder innere Knoten ist ein Subgraph

des jeweiligen Elternknotens, wobei die Blätter Fragmente mit einer einzelnen Kante sind. Grundsätzlich wird zwischen verschiedenen Fragmenten unterschieden, die beispielsweise eine einfache Sequenz von Aktivitäten oder Verzweigungen repräsentieren. Diese Fragmente werden gemäß des vom Modell vorgeschriebenen Kontrollflusses geordnet. Zusätzlich werden die einzelnen Fragmente mit den aus dem Originalmodell extrahierten linguistischen Informationen annotiert. Damit ist die Linearisierung des Originalprozessmodells abgeschlossen. Eine Linearisierung ist nicht nur für imperative, sondern auch für deklarative Prozessmodelle relevant. Allerdings sind die Voraussetzungen grundverschieden. Imperative Prozessmodelle basieren auf gerichteten Graphen, sodass explizit die möglichen Abfolgen von Aktivitäten durch den Prozess und auch die Verzweigungspunkte klar erkennbar sind. Ein natürlichsprachlicher Text kann sich an diesen Abfolgen orientieren, um die Beschreibung der Modellinhalte sequenziell anzuordnen. Deklarative Prozessmodelle verfügen per Definition nicht explizit über derartige Informationen. Folglich lassen sich auf die Linearisierung deklarativer Prozessmodelle nicht die gleichen Prinzipien anwenden wie auf imperative.

*Textstrukturierung*

Die Strukturierung des Zieltextes ergibt sich zum Teil aus der Linearisierung des Prozessmodells, welche die Reihenfolge der Sätze zur Beschreibung der Modellinhalte vorgibt. Ein Dokument kann jedoch auch hierarchische Strukturen, wie beispielsweise Kapitel, Abschnitte, Absätze und Aufzählungen aufweisen. Damit ist die allgemeine *Textstrukturierung* ebenfalls eine Herausforderung jedes NLG-Ansatzes. Die hier diskutierte NLG-Technik bietet eine Möglichkeit, Schwellwerte für die Maximalgröße von Paragraphen und für die maximale Länge einer durch Komma getrennten Aufzählung zu parametrieren. Bei Überschreitung des Paragraphen-Schwellwertes wird der laufende Paragraph an einer inhaltlichen Sollbruchstelle – beispielsweise bei einem Rollenwechsel – abgeschlossen und es wird ein neuer begonnen. Überschreitet eine Aufzählung den dafür vorgesehenen Schwellwert, dann wird die Aufzählung in Spiegelstriche umgewandelt. Geschachtelte Verzweigungen werden durch entsprechende Einrückung abgebildet. In einer Textstruktur für imperative Prozessmodelle können Verzweigungen als Aufzählungspunkte abgebildet werden. In deklarativen Prozessmodellen sind diese Entscheidungsmöglichkeiten aber meist nicht explizit formuliert. Daher können Methoden zur Textstrukturierung für imperative Prozessmodelle nur teilweise für deklarative Prozessmodelle wiederverwendet werden.

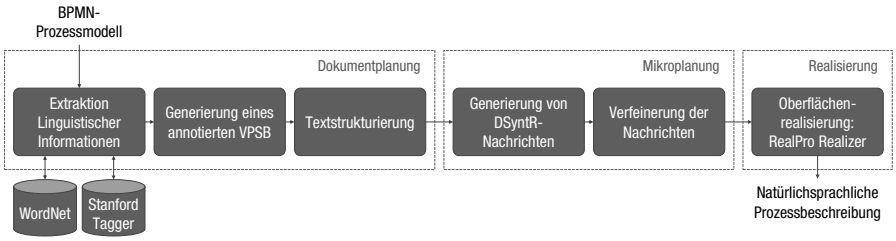



Abbildung 6.2: Architektur des NLG-Systems für BPMN-Modelle (basierend auf: [107])

#### 6.1.1.2 *Mikroplanung*

Im Bereich der Mikroplanung identifizieren die Autoren zwei weitere Herausforderungen. Die *Lexikalisierung* ist eine der beiden Herausforderungen, da sie die Abbildung von Konstrukten des Prozessmodells auf geeignete Wörter und Phrasen im Zieltext erfordert. Zusätzlich ist eine Integration der aus dem Modell extrahierten linguistischen Informationen – z.B. der Verbteil eines Aktivitätsnamens – in verständlicher Form notwendig. Der in Abbildung 6.2 dargestellte Ansatz transformiert dazu den annotierten Verfeinerten Prozessstrukturbaum in eine Liste von Zwischennachrichten. Letztere werden als DSyntR-Strukturen (Abschnitt 3.6.3) organisiert und entsprechen folglich Spezifikationen natürlichsprachlicher Sätze, welche als Baumstruktur repräsentiert wird. Für die möglichen Fragmenttypen des VPSB sind unterschiedliche DSyntR-Schablonen vordefiniert, welche mit den Inhalten aus den Annotationen des VPSB befüllt werden. So können beispielsweise das Verb eines Aktivitätsnamens als Hauptverb, die Rolle als Subjekt und ein Datenobjekt als Objekt der DSyntR-Struktur eingesetzt werden. Die zu den Lexemen der Nachrichtenstruktur gehörenden Grammeme werden auf Basis eines auf empirischer Grundlage entwickelten Regelsystems in die Struktur eingefügt. Die Einbettung dieser bereits extrahierten linguistischen Modellinformationen in vorgefertigte DSyntR-Schablonen ist vom Modellierungsparadigma unabhängig. Folglich kann dieser Teil der Lexikalisierung für ein NLG-System für deklarative Prozessmodelle wiederverwendet werden. Die Prozessmodellelemente imperativer Sprachen, die in natürlichsprachlichen Beschreibungen abgebildet werden müssen, unterscheiden sich stark von denen deklarativer Prozessmodelle. Folglich sind auch ein anderer Wortschatz sowie andere Regeln zur Wortwahl notwendig, um einen verständlichen, natürlichsprachlichen Text zu produzieren. Demnach können Techniken für die Lexikalisierung nur zum Teil auf NLG-Systeme für deklarative Prozessmodelle übertragen werden.

*Lexikalisierung*

Die zweite Herausforderung im Bereich der Mikroplanung ist die *Verfeinerung der Nachrichten*. Dabei handelt es sich einerseits um die Aggregation von Sätzen, also die Verknüpfung oder Schachtelung von Sätzen zu einem komplexeren Satz. Andererseits beinhaltet es auch die Einführung referenzierender Ausdrücke, wie beispielsweise Pronomen und der gezielte Einsatz von Diskursmarkierungen wie „danach“ oder „zuletzt“. Für die Aggregation muss zunächst das Potential der Zusammenführung erkannt werden. Die Zusammenführung selbst stellt eine eigene Herausforderung dar. Um referenzierende Ausdrücke verwenden zu können, müssen die unterschiedlichen Entitätstypen ermittelt werden. Das ist notwendig, damit beispielsweise *der* Antrag respektive *das* Sekretariat korrekt mit *er* respektive *es* referenziert werden. Diskursmarkierungen sind Orientierungspunkte in natürlichsprachlichen Texten, welche ihre Lesbarkeit verbessern. In [107] werden Sequenzen von Aktivitäten, welche von der gleichen Rolle bearbeitet werden, die gleichen Geschäftsobjekte verwenden oder gleiche Tätigkeiten beschreiben, aggregiert. Existieren nach dieser Aggregation dennoch aufeinanderfolgende Nachrichten für Aktivitäten, die von der gleichen Rolle bearbeitet werden, dann wird ab der zweiten Nachricht der Rollenname jeweils durch ein Personalpronomen ersetzt. An dieser Stelle wird erneut WordNet eingesetzt, um das korrekte Pronomen zu ermitteln. Diskursmarkierungen werden zufällig aus einer vorkonfigurierten Menge

*Verfeinerung der Nachrichten*

ausgewählt und gezielt in direkt aufeinanderfolgende Nachrichten eingefügt. Die Aufgabe der Verfeinerung der Nachrichten ist von der jeweiligen Domäne und somit auch vom verwendeten Paradigma der Prozessmodellierung weitestgehend unabhängig. Das Erkennen des Potentials einer Aggregation und ihre Umsetzung beruhen auf rein linguistischen Konzepten, sodass auf Textebene operierende Techniken unmittelbar wiederverwendet werden können. Gleiches gilt für die Einführung referenzierender Ausdrücke und Diskursmarkierungen. Leopold et al. nutzen jedoch strukturelle und inhaltliche Informationen des Prozessmodells, um ein Aggregationspotential zu erkennen. Deklarative Prozessmodelle verfügen nicht über die gleichen Strukturinformationen wie ein imperatives, sodass beispielsweise eine Sequenz von Aktivitäten, die von der gleichen Rolle ausgeführt werden sollen, ad hoc nicht erkennbar ist. Damit ist eine Übertragung dieser Techniken auf den in Abschnitt 6.2 vorgestellten Ansatz nicht möglich. Die Grundidee, sich überschneidende Strukturen des Modells als Indikator für ein Aggregationspotential zu verwenden, wird jedoch später wieder aufgegriffen.

#### 6.1.1.3 *Realisierung und unterschiedliche Charakteristika der Eingabemodelle*

Nach Bewältigung der bisher diskutierten Herausforderungen liegt als Eingabe für die Oberflächenrealisierung eine Textspezifikation vor. Damit ist eine der verbleibenden Herausforderungen die *Generierung des Zieltexts* auf Basis dieser Spezifikation. Die Autoren argumentieren hier, dass die verfügbaren Ansätze zur Oberflächenrealisierung sich zwar hinsichtlich der zugrundeliegenden Theorie und benötigter Eingaben unterscheiden, jedoch alle domänenunabhängig sind. Daher ist es nicht verwunderlich, dass es sich hierbei um das am besten erforschte Teilgebiet der Natural Language Generation handelt. Die generelle Domänenunabhängigkeit der Techniken zur Oberflächenrealisierung ermöglicht folglich auch eine Wiederverwendung derselben für die Entwicklung eines NLG-Systems für deklarative Prozessmodelle.

*Generierung des Zieltexts*

Eine letzte Herausforderung liegt in den *Unterschiedlichen Charakteristika der Eingabemodelle*, welche von einem NLG-System ein gewisses Maß an Flexibilität fordern. Ein BPMN-Modell kann beispielsweise Informationen über die ausführende Rolle einer Aktivität in Form von Swimlanes beinhalten. In diesem Fall können die Beschreibungen der einzelnen Aktivitäten in Aktiv-Form und mit der entsprechenden Rolle als Subjekt formuliert werden. Sind diese Informationen nicht vorhanden, dann sollte eine Passivkonstruktion gewählt werden. Die Autoren des im aktuellen Abschnitt beschriebenen Ansatzes beschreiben keine konkrete Bewältigungsstrategie für die Steigerung der Flexibilität des Ansatzes. Allerdings ist diese Problematik auch für deklarative Prozessmodelle relevant. Selbige können beispielsweise ebenfalls organisatorische Informationen enthalten – was jedoch nicht obligatorisch ist. Folglich muss auch ein NLG-System für deklarative Prozessmodelle diese Form der Flexibilität gestatten. Die Idee aus [107], zwischen Aktiv- und Passivformulierung bei Prä- respektive Absenz von organisatorischen Informationen zu variieren, kann für ein solches System wiederverwendet werden.

*Unterschiedliche Charakteristika der Eingabemodelle*

### 6.1.2 *SBVR-konforme NLG für BPMN-Modelle*

Die in [118] vorgestellte Technik transformiert BPMN-Modelle in SBVR-konforme natürlichsprachliche Prozessbeschreibungen. Zwar wird eine Umsetzung des allgemeinen NLG-Pipeline-Prinzips nicht explizit beschrieben, jedoch lassen sich die einzelnen Arbeitsschritte ohne weiteres in selbiges einordnen. Das Grundprinzip der Extraktion der Informationen aus dem BPMN-Modell beruht auf einer direkten Abbildung von BPMN- auf SBVR-Elemente. Beispielsweise wird ein XOR-Gateway mit zwei Ausgängen auf das SBVR-Element *Exclusive Disjunction* abgebildet, welches in kontrolliert-natürlichsprachlicher Form als If-then-else-Konstruktion dargestellt wird. Ein denkbares Beispiel für das Resultat ist: „If Application approved then Book accomodation else Archive Rejection." Diese SBVR-Konstruktionen dienen als Nachrichten der NLG-Pipeline. Nach Transformation der einzelnen BPMN-Elemente erfolgt die Textstrukturierung, welche auf einer Analyse der Entscheidungsmöglichkeiten des Prozessmodells basiert. Ein Gateway mit einer Ja-Nein-Entscheidung wird beispielsweise so umgesetzt, dass zunächst der Positiv-Pfad und danach der Negativ-Pfad im Zieltext erscheint. Für nichtbinäre Entscheidungen werden die Beschreibungen aller Entscheidungsmöglichkeiten nacheinander angeordnet und dem Gateway-Text werden die Beschriftungen der einzelnen Entscheidungskanten angefügt. Für die Oberflächenrealisierung wird die im Standard enthaltene Notation *SBVR Structured English* angewendet. Das Ergebnis des oben genannten Beispiels wären folgende Sätze:

*Direkte Assoziation von BPMN- und SBVR-Elementen*

- „If Application is approved, it is necessary that an Accomodation is booked."
- „If Application is not approved, it is necessary that the Rejection is archived."

Diese Technik zur Generierung natürlichsprachlicher Texte ist ausschließlich für BPMN-Modelle geeignet und schränkt die Zielsprache auf SBVR-konformes Englisch ein. Weiterhin ist die Abbildung der BPMN-Modellelemente auf SBVR-Elemente nur unvollständig beschrieben und es wird keine Aussage darüber getroffen, wie eine generelle Struktur des Zieldokuments erreicht wird. Enthält ein Prozessmodell mehrere AND-Gateways, dann ist unklar, in welcher Reihenfolge die Beschreibungen der einzelnen ausgehende Pfade im Zieltext zu materialisieren sind. Durch die Einschränkung auf SBVR-konformes Englisch kann die Verständlichkeit der generierten Texte nur beschränkt verbessert werden. Der Ansatz enthält keine auf deklarative Prozessmodelle übertragbaren Komponenten.

*Nur kontrollierte natürliche Sprache*

Eine wesentliche Erkenntnis des aktuellen Abschnitts ist, dass die größten Herausforderungen für die Generierung natürlichsprachlicher Prozessbeschreibungen aus deklarativen Prozessmodellen in der ersten Phase der NLG-Pipeline, der Dokumentplanungsphase, liegen. Der Grund dafür ist, dass die Abhängigkeit von NLG-Techniken von der jeweiligen Domäne in der ersten Phase der NLG-Pipeline am größten und in der letzten am geringsten ist (Abschnitt 3.6). Folglich können Techniken, welche in den späteren Phasen der Pipeline eingesetzt werden, tendenziell wiederverwendet werden. Techniken für frühere Phasen müssen dagegen als Speziallösung der jeweiligen Domäne entwickelt werden. Aus diesem Grund konzentriert sich das Konzept zur Generierung natürlichsprachlicher Beschreibungen für deklarative Prozessmodelle besonders auf die Phase der Dokumentplanung.

## 6.2 GENERIERUNG NATÜRLICHSPRACHLICHER BESCHREIBUNGEN FÜR MULTI-PERSPEKTIVISCHE, DEKLARATIVE PROZESSMODELLE AUF BASIS DES BEDEUTUNG-TEXT-MODELLS UND RHETORISCHER RELATIONEN

Da verwandte Arbeiten sich ausschließlich mit imperativen Prozessmodelle beschäftigen (Abschnitt 6.1), beschränkt sich die in diesem Kapitel vorgestellte NLG-Technik auf die natürlichsprachliche Beschreibung *deklarativer* Prozessmodelle. Als Modellierungssprache für die Eingabemodelle wird dabei die multi-perspektivische deklarative Prozessmodellierungssprache DPIL verwendet, da diese nachweislich [161, 199, 200] über eine große Ausdrucksmächtigkeit verfügt.

Einige Konzepte können aus verwandten Arbeiten wiederverwendet werden, andere müssen jedoch angepasst oder sogar neu entwickelt werden (Tabelle 6.1). Die Generierung natürlichsprachlicher Prozessbeschreibungen für DPIL-Modelle orientiert sich allgemein am üblichen Pipeline-Modell aktueller NLG-Systeme (Abschnitt 3.6.2) und speziell an der Umsetzung für imperative Prozessmodelle [107]. Abbildung 6.3 zeigt eine Übersicht des vollständigen Generierungsprozesses.

*Vielfalt natürlichsprachlicher Formulierungen*

Dieselben Sachverhalte können oft durch unterschiedliche natürlichsprachliche Formulierungen beschrieben werden. Beispielsweise kann eine Handlung in Aktiv- oder Passivform ausgedrückt werden. Zwar können sich diese Formulierungen in der Betonung unterscheiden, die vermittelten Informationen bleiben jedoch nahezu identisch. Durch diese und andere Variabilitäten sind in einigen Konzepten der NLG-Technik alternative Lösungen denkbar. Ein Beispiel ist die Strukturierung des Zieltextes und die zugehörige Gruppierung der zu beschreibenden Prozessmodellinhalte. Daher hat die nachfolgende Beschreibung Vorschlagscharakter und dient primär der Illustration der notwendigen Schritte für die Anwendung allgemeiner NLG-Prinzipien auf deklarative Prozessmodelle.

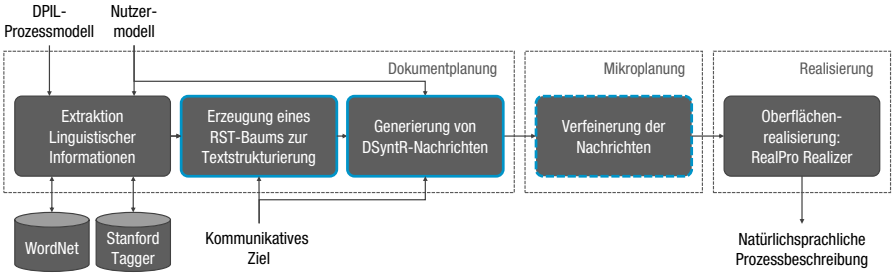



Abbildung 6.3: Architektur des NLG-Systems für DPIL-Modelle

*Allgemeines Vorgehen*

Im ersten Schritt werden, wie auch in [107], die bereits im Prozessmodell vorhandenen linguistischen Informationen extrahiert und aufbereitet. Danach werden rhetorische Relationen (Abschnitt 3.6.3) eingesetzt, um den Zieltext zu strukturieren. Als letzte Maßnahme der Dokumentplanung werden Modellinhalte sowie das kommunikative Ziel und das Nutzermodell verwendet, um Nachrichten in Form von DSyntR-Strukturen zu erzeugen (Abschnitt 3.6.4). In der Mikroplanungsphase wird die Lesbarkeit des Zieltextes durch Verschmelzung von Nachrichten sowie die Erzeugung referenzierender Ausdrücke und Diskursmarkierungen verbessert. Abschließend wird die so erzeugte Textspezifikation mittels Oberflächenrealisierung in eine natürlichsprachliche Prozessbeschreibung umgewandelt.

```
use identity SJ
use group professor          // P

process BusinessTrip {

    task Request permission   // A
    task Approve application  // B
    task Book flight          // C
    task Book accommodation   // D
    task Book transfer        // E

    document Application      // X

    ensure produces(Request permission,Application)
    advise sequence(Approve application,Book flight)
    ensure binding(Book accommodation,Book transfer)
    ensure role(Approve application,Professor)
    ensure consumes(Approve application,Application)
    ensure sequence(Book accommodation,Book transfer)
    ensure binding(Request permission,Book flight)
    advise sequence(Approve application,Book accommodation)
    advise direct(Approve application,SJ)
```

Codeausschnitt 6.1: DPIL-Modell eines vereinfachten Dienstreiseprozesses

*Beispielmodell*

Um die einzelnen Generierungsschritte zu illustrieren wird das in Codeausschnitt 6.1 dargestellte DPIL-Modell verwendet. Selbiges ist an den in Abschnitt 3.1.5 bereits erläuterten Dienstreiseprozess angelehnt, wird aber zugunsten der Übersichtlichkeit der nachfolgenden Erläuterungen stark vereinfacht. Zudem wird eine englische Übersetzung verwendet, da die eben erwähnte Technologie zur Oberflächenrealisierung auf die englische Sprache beschränkt ist. Das Modell beschreibt Zusammenhänge zwischen insgesamt fünf Aktivitäten, einem Datenobjekt sowie einer organisatorischen Ressource, welche zusammen die Entitäten des Prozesses darstellen. Alle enthaltenen Regeln sind mittels der in Abschnitt 3.1.5 formulierten Regelschablonen definiert. Somit ergibt sich die Semantik der einzelnen Regeln wie folgt (Nummerierung entspricht den Zeilennummern 14 – 22):

Z14: Im Rahmen der Bearbeitung der Aufgabe *Request permission* ist das Antragsdokument zu erzeugen.

Z15: Ein Flug sollte erst nach Genehmigung des Antrags gebucht werden.

Z16: Unterkunft und Transfer müssen von derselben Person gebucht werden.

Z17: Ein Antrag kann nur von einem Professor genehmigt werden.

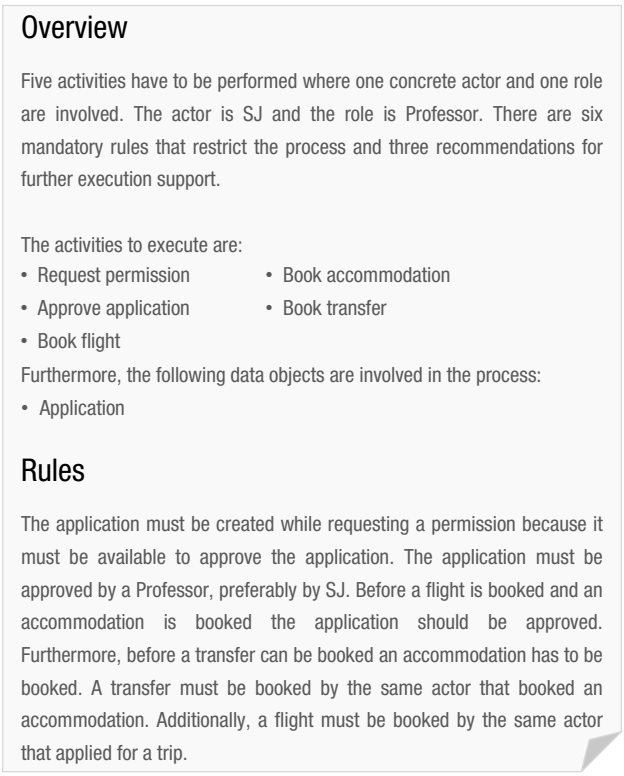
Z18: Ein Antrag kann erst genehmigt werden, wenn das zugehörige Antragsdokument erzeugt wurde.

Z19: Die Unterkunft muss vor dem Transfer gebucht werden.

Z20: Der Flug ist vom Antragsteller zu buchen.

Z21: Die Unterkunft sollte erst nach Genehmigung des Antrags gebucht werden.

Z22: Der Antrag sollte von der Ressource SJ genehmigt werden.



## Overview

Five activities have to be performed where one concrete actor and one role are involved. The actor is SJ and the role is Professor. There are six mandatory rules that restrict the process and three recommendations for further execution support.

The activities to execute are:

- Request permission
- Approve application
- Book flight
- Book accommodation
- Book transfer

Furthermore, the following data objects are involved in the process:

- Application

## Rules

The application must be created while requesting a permission because it must be available to approve the application. The application must be approved by a Professor, preferably by SJ. Before a flight is booked and an accommodation is booked the application should be approved. Furthermore, before a transfer can be booked an accommodation has to be booked. A transfer must be booked by the same actor that booked an accommodation. Additionally, a flight must be booked by the same actor that applied for a trip.

Abbildung 6.4: Beispiel für die natürlichsprachliche Beschreibung eines DPIL-Modells

Der Prozess ist in vielerlei Hinsicht vereinfacht. Allerdings kann das Prozessmodell als Zwischenstand einer Modellierungsaufgabe betrachtet werden, für den die nachfolgend erläuterte Technik zur Generierung natürlichsprachlicher Prozessbeschreibungen als Hilfsmittel dienen soll. Ein möglicher Zieltext dieser Generierung ist in Abbildung 6.4 dargestellt.

Unmittelbar nachfolgend werden die unterschiedlichen Eingabeparameter der allgemeinen NLG-Pipeline mit den Verfügbaren Informationen im Prozessmanagementkontext verknüpft. Anschließend werden die einzelnen in Abbildung 6.3 dargestellten Generierungsschritte betrachtet, welche schließlich vom Eingabemodell in Codeausschnitt 6.1 zum in Abbildung 6.4 illustrierten Beispiel-Zieltext führt.

### 6.2.1 *Eingabeparametrierung*

Im aktuellen Abschnitt werden die allgemeinen Eingabeparameterkonzepte der NLG-Pipeline (Abschnitt 3.6.2) für die Erzeugung natürlichsprachlicher Beschreibungen deklarativer Prozessmodelle konkretisiert. Wie bereits eingangs erwähnt, wird dabei von der ausdrucksstarken, multi-perspektivischen, deklarativen Prozessmodellierungssprache DPIL ausgegangen.

*Wissensbasis*

Die *Wissensbasis* setzt sich aus dem gegebenen DPIL-Modell sowie einem Modell zur Beschreibung der Organisationsstruktur zusammen. Von selbigem wird angenommen, dass es dem in [38] vorgeschlagenen organisatorischen Metamodell folgt, welches in DPIL jedoch ohnehin zur Beschreibung von Beziehungen zwischen organisatorischen Einheiten und anderen Entitäten des Prozesses benötigt wird.

Weiterhin wird angenommen, dass die in diesem Nutzermodell enthaltenen Relationstypen zwischen organisatorischen Einheiten immer der Form *HilfsverbNomen*[1] folgt. Beispiele dieser Relationstypen sind „hatRolle", *istVorgesetzter* oder auch *istMitglied*. Eine weitere Einschränkung ist, dass die Regeln des DPIL-Modells mittels eines festen Satzes an Regelschablonen formuliert sind, welche im nachfolgenden Abschnitt aufgelistet werden. Grund für diese Einschränkung ist, dass DPIL Boolesche Operatoren unterstützt, die eine beliebige Schachtelung logischer Ausdrücke zulässt. Folglich existieren theoretisch unendlich viele Regeltypen. Da sich der hier diskutierte Ansatz, wie auch jener in [107], auf eine statische Abbildung zwischen Konstrukten der Prozessmodellierungssprache auf linguistische Schablonen beschränkt, wäre diese Abbildung ohne die genannte Einschränkung zwangsläufig unvollständig.

*Kommunikatives Ziel*

Der zweite Eingabeparameter ist das *kommunikative Ziel*. Im Rahmen des Prozessmanagementlebenszyklus werden zwei wesentliche Einsatzszenarien für die hier beschriebene NLG-Technik identifiziert. Diese sind *(i)* die Beschreibung der Modellinhalte zum Modellierungs- respektive Überarbeitungszeitpunkt und *(ii)* die Erläuterung von Prozessregeln zum Ausführungszeitpunkt. Im ersten Fall dient die Generierung des natürlichsprachlichen Texts beispielsweise der Unterstützung des Validierungsprozesses und der Prozessdokumentation. Der zu generierende Text umfasst hierbei in der Regel das vollständige DPIL-Eingabemodell. Der zweite Einsatzzweck ist die Befähigung des Prozessausführungssystems, mittels natürlichsprachlicher Formulierungen die Regeln zu erklären, welche zum jeweilig aktuellen Zeitpunkt die Ausführung der nächsten Prozessschritte beeinflussen. In diesem Fall umfasst der Zieltext meist nur einen Teil des Prozessmodells.

*Nutzermodell*

Ein *Nutzermodell* ist im Kontext von DPIL in Form des bereits erwähnten Organisationsmodells gegeben. Die Festlegung der Struktur dieses Modells durch das allgemeine organisatorische Metamodell aus [38] ermöglicht es, ein generisches Konzept für den Zugriff auf die Informationen des Modells zu entwickeln. Zum Ausführungszeitpunkt können so Informationen über Position und Beziehungen des aktiven Nutzers des Systems, für den der Text erzeugt werden soll, ermittelt werden. Für die Nutzung der NLG-Technik zum Modellierungszeitpunkt stehen diese Informationen nicht automatisch zur Verfügung. An dieser Stelle kann eine Entscheidung getroffen werden, ob der zu generierende Text personalisiert werden soll oder nicht. Im ersten Fall müsste so der entsprechende Nutzer aus dem Organisationsmodell ausgewählt werden.

*Diskurshistorie*

Im Standard-Pipeline-Modell eines NLG-Systems ist ein weiterer Parameter vorgesehen – die *Diskurshistorie*. Bei mehrfachen Interaktionen mit dem NLG-System dient dieser Parameter als Informationsgrundlage für bereits kommunizierte Informationen, sodass der Zieltext in dieser Hinsicht frei von Redundanzen gehalten werden kann. Diese Problematik manifestiert sich jedoch primär bei der Erläuterung von DPIL-Regeln zum Ausführungszeitpunkt, da bei der Textgenerierung zum Modellierungszeitpunkt in der Regel das vollständige Modell beschrieben werden soll. Zum Ausführungszeitpunkt kann über die Prozesslaufzeit hinweg mehrfach der Wunsch nach natürlichsprachlichen Erläuterungen von Prozessregeln auftreten. Da dies jedoch auch in zeitlich großen Abständen erfolgen kann, darf

1 Die Trennung zwischen Hilfsverb und Nomen erfolgt am jeweiligen Binnenmajuskel.

nicht pauschal entschieden werden, welche Informationen dem Nutzer noch bekannt sind. Aus den genannten Gründen werden Optimierungsmöglichkeiten auf Basis der Diskurshistorie für den vorliegenden Ansatz explizit ausgeblendet und damit für spätere Weiterentwicklungen offengelassen.

Die in diesem Abschnitt beschriebenen Eingabeparameter umfasst somit das zu beschreibende DPIL-Modell oder Teile desselben, das kommunikative Ziel, ob die Generierung zum Modellierungs- oder Ausführungszeitpunkt erfolgt sowie Nutzerinformationen welche als Organisationsmodell vorliegen.

### 6.2.2 *Dokumentplanung*

Die Phase der Dokumentplanung besteht aus den Schritten der Extraktion linguistischer Informationen, der Generierung von Nachrichten als DSyntR-Baumstrukturen sowie die Erzeugung einer hierarchischen Strukturbeschreibung des Zieltextes auf Basis rhetorischer Relationen Abbildung 6.3. Die Eingabe für das Dokumentplanungsmodul sind die im vorherigen Abschnitt diskutierten Parameter. Das Resultat dieses Moduls ist ein Dokumentplan in Form eines rhetorischen Strukturbaums, dessen Blattknoten DSyntR-Nachrichten sind. Nachfolgend wird erläutert, wie die Eingaben in diese Ausgabe transformiert werden.

*Extraktion linguistischer Informationen*

Die *Extraktion linguistischer Informationen* kann von dem in [107] beschriebenen Ansatz übernommen werden. Folglich wird die Bezeichnung jeder einzelnen Aktivität des Modells in einzelne Wörter zerlegt, für welche jeweils die Wortart bestimmt wird. Der alleinige Einsatz des Stanford Taggers reicht in vielen Fällen nicht aus, weil dieser und vergleichbare Technologien bei kurzen Textabschnitten fehleranfälliger sind als bei längeren. Aus diesem Grund wird mit WordNet, welches eine erweiterte Form eines Thesaurus darstellt, eine zusätzliche Überprüfung durchgeführt. Weiterhin werden die verwendeten Relationstypen organisatorischer Beziehungen zerlegt. Da für diese eine Kombination aus Hilfsverb und Substantiv vorausgesetzt wird, kann die Trennung am enthaltenen Binnenmajuskel vorgenommen werden. Die Wortarten müssen demnach nicht nachträglich bestimmt werden. Falls das kommunikative Ziel die Erklärung einer oder mehrerer Regeln zur Ausführungszeit ist, werden in diesem Schritt nur die Aktivitätsbezeichnungen und Relationstypen verarbeitet, die von einer oder mehreren dieser Regeln referenziert werden. Das Ergebnis dieses Schrittes sind die einzelnen Teilwörter der relevanten Aktivitätsbezeichnungen und Relationstypen inklusiver der jeweiligen Wortart. Für die beispielhafte Aktivität „Flug buchen“ und den exemplarischen Relationstyp „istVorgesetzter“ ist das Ergebnis „Flug/COMMON_NOUN“ und „buchen/VERB“ respektive „ist/AV“ und „Vorgesetzter/COMMON_NOUN“.

*Erzeugung eines RST-Baums zur Textstrukturierung*

Im zweiten Schritt erfolgt die Ermittlung der Struktur des Zieltextes, die in Form eines *RST-Baums* (Abschnitt 3.6.4) angegeben wird. Ist das kommunikative Ziel die Beschreibung der Wirkungsweise einer oder mehrerer Regeln während der Prozessausführung, wird dieser Schritt übersprungen.

Die Blattknoten eines RST-Baums beinhaltet die einzelnen Bestandteile des Textes, während die inneren Knoten deren Anordnung bestimmen. Um inhaltlich zusammenhängende DPIL-Regeln in einer solchen Struktur gruppieren zu können, müssen sie zunächst sortiert werden. Dafür wird, angelehnt an [107], ein regelba-

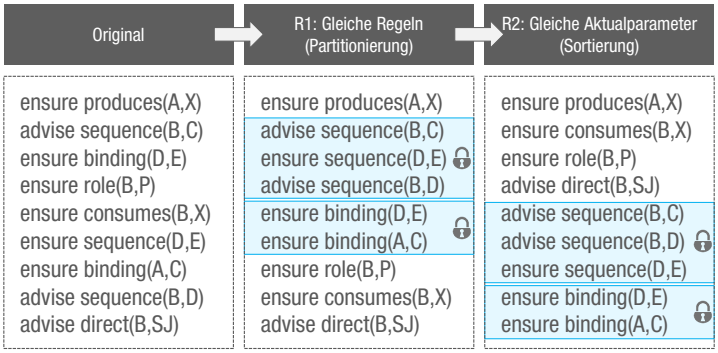



Abbildung 6.5: Anwendung der Sortierregeln auf das DPIL-Modell aus Codeausschnitt 6.1

siertes Sortierungsprinzip entwickelt. Es umfasst lediglich zwei Regeln, welche in der hier angegebenen Reihenfolge angewendet werden:

R1: *Gleiche Regelschablone:* Diese Regel betrifft Prozessregeln, welche dieselbe Regelschablone verwenden, aber unterschiedliche Aktualparameter aufweisen.

R2: *Gemeinsame Aktualparameter:* Die zweite Gruppierung betrachtet nur die Argumente der Regeln. Auch in diesem Fall entscheidet die Anzahl gemeinsamer Argumente über die Reihenfolge der Regeln innerhalb der Gruppe.

Wie beispielsweise das Regelpaar in den Zeilen 11 und 16 von Codeausschnitt 6.1 zeigt, können sich die beiden Sortierregeln überschneiden. Die Reihenfolge der Anwendung der Regeln ist zudem entscheidend für die resultierende Sortierung. Folglich ist eine Priorisierung der beiden Regeln notwendig. Eine Möglichkeit ist die Festlegung, dass die Sortierregel *R1* eine höhere Priorität als *R2* zukommt. Folglich darf nach Anwendung der Regel *R1* die Anwendung von *R2* die Sortierung nur in der Art verändern, dass *R1* noch immer Gültigkeit hat. Diese Eigenschaft eines Sortierverfahrens wird als *Stabilität* bezeichnet. Das Sortierverfahren *Bucketsort* [45] ist ein Vertreter dieser Klasse. Hierbei werden die Elemente zunächst partitioniert. In einem zweiten Schritt werden die Elemente jeder Partition nach einem beliebigen Sortierverfahren geordnet. Im letzten Schritt werden die einzelnen Gruppen wieder aufgelöst. Im aktuellen Kontext wird die Sortierregel *R1* zu Erzeugung der Partitionen verwendet, während *R2* zur Sortierung innerhalb derselben eingesetzt wird.

*Stabiles Sortierverfahren*

Zur Demonstration der Wirkungsweise dieses Verfahrens wird das Prozessmodell aus Codeausschnitt 6.1 verwendet. Die Sortierung gemäß der angegebenen Priorisierung ist in Abbildung 6.5 dargestellt. Zur kompakteren Darstellung sind die Entitäten des Prozessmodells mit den in Codeausschnitt 6.1 zugeordneten Buchstaben abgekürzt.

Im ersten Schritt werden die Regeln des DPIL-Modells partitioniert. Dabei entstehen die zwei farblich hervorgehobenen Partitionen für die `sequence`- und `binding`-Regeln sowie eine weitere für alle Regeln, deren Regelschablone lediglich einmal im Modell verwendet wird. Im zweiten Schritt werden die Regeln innerhalb jeder Partition nach absteigender Anzahl gemeinsamer Aktualparameter sortiert. In vielen Fällen – und so auch im konkreten Beispiel – ist die Anzahl

der gemeinsamen Aktualparameter der einzelnen Regeln gleich. In diesen Fällen werden die Regeln zwar ebenfalls gruppiert, die finale Sortierung ist jedoch von der Originalreihenfolge abhängig.

*Anpassung der Sortierung*

Weder die angegebene Sortierung noch die Regeln selbst sind die einzig mögliche Lösung des Problems der Textstrukturierung. Daher ist beides, wie bereits eingangs angedeutet, als exemplarischer Durchstich zu betrachten. Abhängig vom konkreten Prozessmodell und individuellen Präferenzen kann eine andere Sortierung gewünscht sein. Aus diesem Grund ist die Austauschbarkeit oder Konfigurationsmöglichkeit der Sortierungskomponente in der NLG-Systemarchitektur zu berücksichtigen.

*Anwendung der Theorie rhetorischer Strukturen*

Die Sortierung ist nur ein Vorbereitungsschritt, welcher Regeln mit sich überschneidenden Bestandteilen gruppiert. Aus der Perspektive des Textflusses besteht jedoch noch kein Zusammenhang zwischen den sortierten Regeln. Der in Abbildung 6.4 dargestellte Zieltext weist jedoch eine Reihe stilistischer Überleitungen auf, die in natürlichsprachlichen Äußerungen verwendet werden, um die Lesbarkeit eines Textes zu steigern. Diese Überleitungen sind im angegebenen Text im Wesentlichen Signalwörter, wie beispielsweise „furthermore“ und „additionally“. Diese ähneln sich semantisch und können unter anderem anzeigen, dass die zwei aufeinanderfolgenden Sätze Informationen in aufzählungsartiger Form vermitteln. Enthielte das Modell eine zweite `direct`-Regel mit vollständig anderen Argumenten, dann würden diese intuitiv eher mit einem kontrastierenden Signalwort verknüpft werden. Ein Beispiel ist das Wort „however“, welches unter anderem zur Erzeugung der folgenden Form verwendet werden kann „... preferably by SJ. However, the task ... by XY.“ Folglich müssen die Beziehungen zwischen den aufeinanderfolgenden Regeln klassifiziert werden. Hierfür wird bereits in zahlreichen Anwendungen [154] von der Theorie rhetorischer Strukturen (Abschnitt 3.6.4) Gebrauch gemacht. Die Theorie beschreibt eine Reihe rhetorischer Relationen zwischen Teilen eines Textes. Ein Teil kann dabei ein einzelner Satz aber auch eine andere rhetorische Relation sein. In der aktuellen Phase der Textplanung existieren noch keine Sätze. Stattdessen wird die RST eingesetzt, um die Beziehungen zwischen einzelnen Regeln sowie Regelgruppen zu klassifizieren. Das Ergebnis ist der bereits erwähnte Rhetorische Strukturbaum. Für den in Abbildung 6.4 dargestellten Zieltext ist ein möglicher Strukturbaum in Abbildung 6.6 dargestellt.

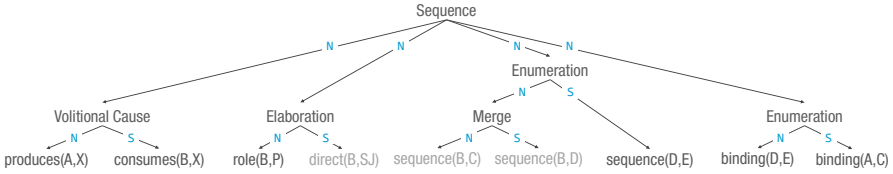



Abbildung 6.6: Rhetorischer Strukturbaum des Zieltextes in Abbildung 6.4

*Verwendete rhetorische Relationen*

Die Blattknoten des Strukturbaumes entsprechen den sortierten Regeln des DPIL-Modells. Die in hellgrau gehaltenen Regeln stellen dabei die enthaltenen Empfehlungen dar. Für benachbarte Regeln derselben Partition kann im Voraus festgelegt werden, wie ihr Zusammenhang sich im Text widerspiegeln soll. Weiter im Sinne einer exemplarischen Betrachtung werden somit folgende rhetorische Relationen verwendet:

- *Volational Cause:* Existiert eine `produces`- und eine `consumes`-Regel für dasselbe Datenobjekt, dann kann diese über ein Ursache-Wirkung-Verhältnis beschrieben werden. Ein Dokument wird erzeugt, weil es im späteren Verlauf des Prozesses benötigt wird.

- *Elaboration:* Existieren für dieselbe Aktivität sowohl eine `role`- als auch eine `direct`-Regel, dann wird mittels ersterem eine Menge von Akteuren und mit letzterem ein Element derselben beschrieben. Dieses Menge-Mitglied-Verhältnis kann mittels einer *Elaboration*-Relation abgebildet werden.

- *Merge:* Zwei `sequence`-Regeln, deren erster Aktualparameter identisch ist, beschreiben die Abhängigkeit zweier unterschiedlicher Aktivitäten von einer bestimmten anderen Aktivität. Diese Relation signalisiert den Wunsch einer Verschmelzung der beiden Regeln im Zieltext.

- *Enumeration:* Wird eine Situation in mehreren Teilen beschrieben, dann werden diese Teile unter einem Knoten vom Typ *Enumeration* vereint.

- *Sequence:* Im Unterschied zur Enumeration-Beziehung besteht in der *Sequence*-Beziehung kein besonders hervorzuhebender inhaltlicher Zusammenhang zwischen den einzelnen Argumenten. Sie ist zudem die einzige Relation mit mehreren Nuklei. Im konkreten Beispiel bildet diese Beziehung die Wurzel des Strukturbaums und beschreibt die Anordnung der einzelnen Regelpartitionen.

Die oben aufgeführte Menge an rhetorischen Relationstypen ist nicht vollständig. Vielmehr soll sie auf exemplarischem Weg verdeutlichen, in welcher Form eine rhetorische Strukturierung der Beschreibungen der Modellregeln vorgenommen werden kann. Die Menge dieser Relationstypen ist unter anderem in [119] beschrieben, kann aber nach Belieben erweitert werden. So gehören beispielsweise Merge und Enumeration nicht zum ursprünglichen Satz der Relationstypen.

Bis hierhin wird lediglich die Sortierung und rhetorische Verknüpfung der Modellregeln betrachtet. Im Zieltext ist jedoch auch ein einleitender Abschnitt erkennbar, welcher einen Überblick über die Modellinhalte liefert. Dieser einleitende Abschnitt variiert weit schwächer als die Menge der im Modell enthaltenen Regeln. Der Grund dafür ist, dass die für den Überblick verwendeten Sätze von Anzahl und Art der im Modell enthaltenen Entitäten und Konstrukte unabhängig sind. Folglich kann dieser Teil als Textschablone mit Platzhaltern für die dynamischen Inhalte vorgefertigt werden. Ein kontextbezogenes Beispiel für eine solche Textschablone ist im oberen Teil von Abbildung 6.8 dargestellt. Alle farblich hervorgehobenen Bestandteile der Schablone sind Platzhalter für dynamische Inhalte. So wird für `#ofActivities` beispielsweise die Anzahl der im DPIL-Modell enthaltenen Aktivitäten eingesetzt. Der Platzhalter `actorsAndRoles` wird dagegen durch eine Aufzählung aller am Prozess beteiligten Akteure und Rollen ersetzt. Dagegen ist `actorsActors` ein Vertreter einer anderen Art Platzhalter, bei der in Abhängigkeit der Anzahl der betroffenen Modellelemente entweder eine Singular- oder eine Pluralform des Wortes eingesetzt wird. Bei einem einzelnen Akteur ist das beispielsweise „actor“ und andernfalls „actors“.

*Einleitungstext: Schablone und Platzhalter*

Im dritten Schritt der Dokumentplanungsphase erfolgt die *Generierung* von DSyntR-Nachrichten, deren Struktur im Rahmen des Bedeutung-Text-Modells beschrieben ist (Abschnitt 3.6.3). Da hinsichtlich der zu beschreibenden DPIL-Regeln angenommen wird, dass diese mittels eines festgelegten Satzes an Regelschablonen definiert werden, kann für jede dieser DPIL-Schablonen jeweils eine DSyntR-Nachrichtenschablone angegeben werden. Diese sind in Tabelle 6.2 angegeben. Jeder der DPIL-Regeln, die in dem im vorherigen Schritt erzeugten RST-Baum enthalten ist, wird im aktuellen Schritt durch die zugehörige DSyntR-Nachricht ersetzt.

*Ersetzung der Regeln durch Nachrichten im Strukturbaum*

Die DSyntR-Nachrichtenschablonen beinhalten sowohl vorgegebene Textteile als auch Anfragen für Informationen aus den DPIL-Modellelementen. Diese Anfragen sind in einer Pseudocode-Notation angegeben und können mit Modell-Anfragesprachen wie beispielsweise OCL umgesetzt werden. Im Fall der `direct` -Regelschablone enthält die zugehörige DSyntR-Nachrichtenschablone beispielsweise das Wort „by" als Präposition sowie „class:verb, tense:pres, voice:pass" als statische Grammeme des Hauptverbs. Die dynamischen Inhalte sind das Hauptverb selbst, das syntaktische Subjekt (I) sowie das syntaktische Objekt (II) des Satzes.

*Zugriffe auf Modellinhalte*

In den Pseudocode-Angaben für dynamische Inhalte sind folgende Operatoren und Markierungen enthalten:

- Mit @ wird ein Funktionsaufruf eingeleitet. Funktionsaufrufe können Argumente annehmen und geschachtelt werden.
- Mit *:* wird der Zugriff auf das zu beschreibende DPIL-Modellelement eingeleitet. Ist das Modellelement eine Hierarchie von Modellelementen, dann kann auf die untergeordneten Modellelemente entweder durch direkte Navigation mit dem Punktoperator oder durch transitive Suche mit dem #-Operator zugegriffen werden.
- Der Operator *&* ermöglicht den Zugriff auf einen Aktualparameter über den Namen des entsprechenden Formalparameters.

In den DSyntR-Nachrichtenschablonen sind verschiedene Funktionen angegeben, deren Eingaben, Semantik und Ausgaben in Tabelle 6.3 aufgeführt sind. Die enthaltene Beschreibung ist bewusst informell gehalten, da beispielsweise für die Ermittlung von Wortarten zahlreiche verschiedenartige Ansätze existieren. Das Nachschlagen in einem Wörterbuch, in einem Thesaurus oder vergleichbaren lexikalischen Datenbanken und die Markierung mit Analysewerkzeugen wie *Part-of-Speech-Taggern* sind nur zwei Möglichkeiten. Einerseits sind die Herausforderungen der Entwicklung der in Tabelle 6.3 nicht für die Prozessdomäne spezifisch und andererseits existieren bereits zahlreiche Lösungen, die auf Grundlage von [107] nachweislich wiederverwendet werden können. Folglich wird für den verbleibenden Teil dieses Kapitels davon ausgegangen, dass für jede der in Tabelle 6.3 aufgeführten Funktionen eine Umsetzung vorhanden ist.

Die oben angegebene Pseudocode-Sprache dient lediglich der kompakteren Beschreibung und muss für andere DPIL-Regelschablonen möglicherweise erweitert werden.

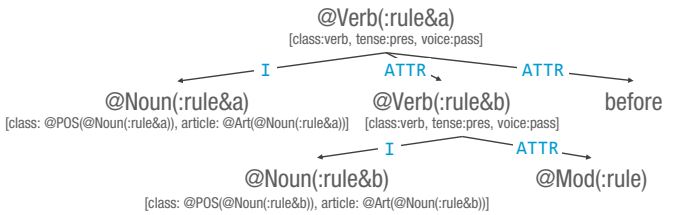
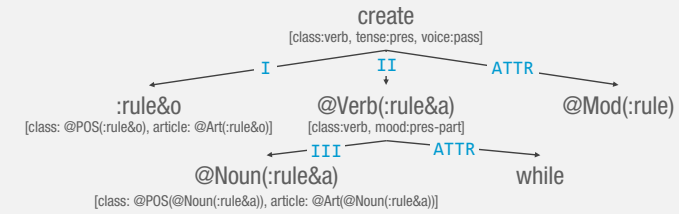
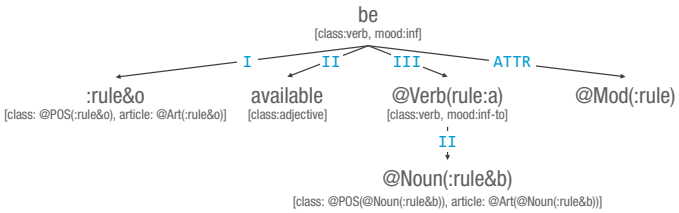
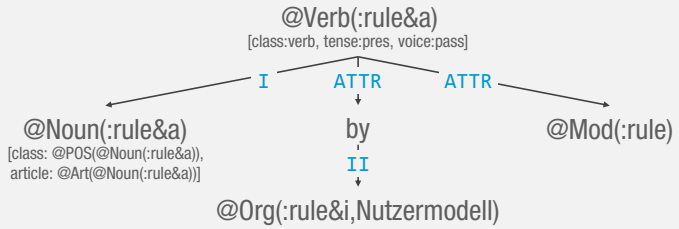
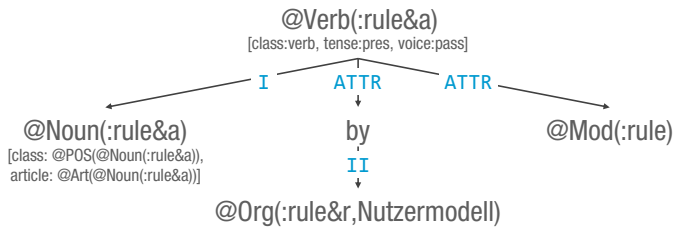
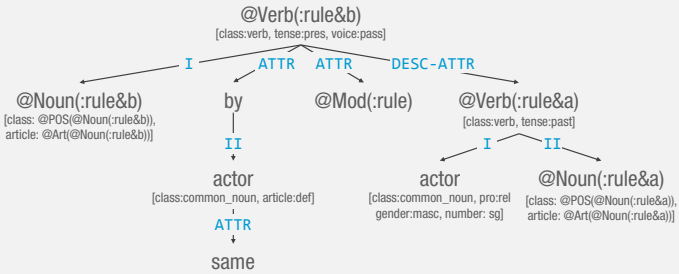
| DPIL-Makro | DSyntR-Nachrichtenschablone |
|---|---|
| `sequence(a,b)` |  |
| `produces(a,o)` |  |
| `consumes(a,o)` |  |
| `direct(a,i)` |  |
| `role(a,r)` |  |
| `binding(a,b)` |  |



Tabelle 6.2: Abbildung von DPIL-Makros auf Schablonen für DSyntR-Nachrichten

| | Eingabe | Semantik | Ausgabe |
|---|---|---|---|
| Art | Substantiv | Ermittelt auf Basis der Art eines Substantivs den zugehörigen Artikeltyp. Ist es ein Eigenname, wird kein Artikel angegeben (no-art). Ist es ein allgemeines Substantiv, dann wird ein bestimmter Artikel vergeben, wenn das Substantiv der Name einer organisatorischen Ressource oder eines Datenobjekts des DPIL-Modells ist (def). Andernfalls wird ein unbestimmter Artikel zurückgegeben (indef). | Grammemwert für Artikeltyp |
| Mod | Regel | Ermittelt auf Basis der Regelmodalität die lexikalische Entsprechung im Zielsatz. Für optionale Regeln wird „should" und für verpflichtende Regeln wird „must" zurückgegeben. | Modalverb |
| Noun | Task oder RelationType | Erfragt das in einem Task oder in einem RelationType enthaltene Substantiv, welches im Schritt der Extraktion linguistischer Informationen bestimmt wird. | Substantiv |
| Org | Identity oder Group, Nutzermodell | Ermittelt auf Basis der Identity oder Group aus einer DPIL-Regel sowie der Identity des Nutzers aus dem Nutzermodell eine eventuelle Beziehung zwischen diesen. Wird kein Nutzermodell angegeben, ist das Ergebnis die Identity oder Group, die als Parameter angegeben werden. Weiterhin werden die zugehörigen Grammeme, wie beispielsweise die Wortklasse und der Artikeltyp bestimmt. | Lexem + Grammeme |
| POS | Wort | Ermittelt die Wortart des gegebenen Wortes (z.B. mit Stanford Tagger und WordNet). | Wortart |
| Verb | Task oder RelationType | Erfragt das in einem Task oder in einem RelationType enthaltene Verb, welches im Schritt der Extraktion linguistischer Informationen bestimmt wird. | Verb |

Tabelle 6.3: Funktionen für dynamische Inhalte der DSyntR-Schablonen

In Abbildung 6.7 sind sowohl die DSyntR-Schablone für die in Codeausschnitt 6.1 angegebene optionale `direct`-Regel – mit der Semantik einer Empfehlung – als auch die vollständige DSyntR-Nachricht dargestellt. Das Ergebnis der in der Schablone angegebenen Modellelementanfrage „:rule#task" für die DPIL-Regel `direct(Approve application,SJ)` ist der Name der einzigen Aktivität der Regel – also „Approve application". Der gesamte Aufruf für das Hauptverb des Satzes macht sich jedoch die im ersten Schritt der Dokumentplanung extrahierten linguistischen Informationen zunutze und selektiert das Verb im Namen der Aktivität. Für die angegebene Beispiel-Regel ist „approve" das Ergebnis. In gleicher Form wird für das syntaktische Objekt das Substantiv der Aktivitätsbezeichnung selektiert, welches im Beispiel das Wort „application" ist. Das Nomen ist gleichzeitig der Name eines im Modell auftretenden Datenobjekts, weswegen für das Objekt der bestimmte Artikel ausgewählt wird. Folglich handelt es sich bei dem

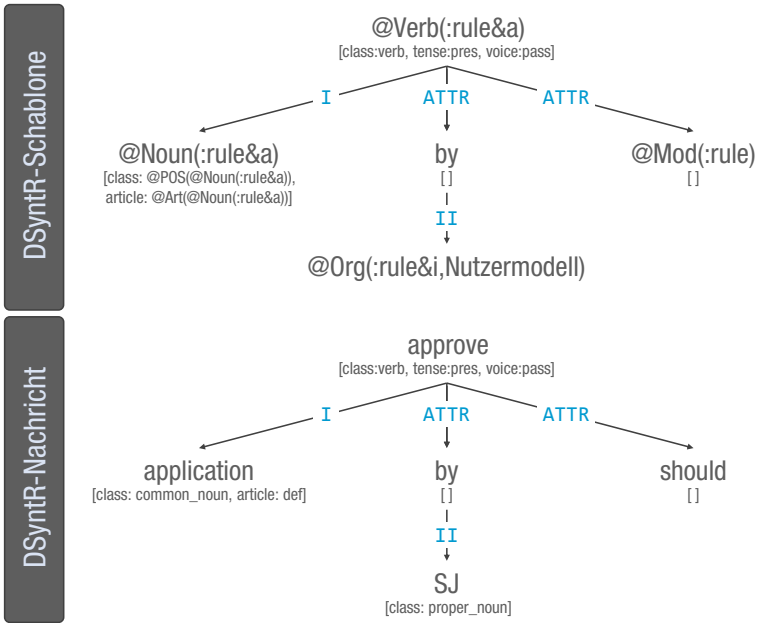



Abbildung 6.7: DSyntR-Nachrichtenschablone und -Nachricht für DPIL-Empfehlung `direct(Approve application,SJ)`

Substantiv auch nicht um einen Eigennamen, sondern ein allgemeines Nomen. Als Objekt des Satzes wird der Name der in die DPIL-Regelschablone eingesetzten organisatorischen Ressource ermittelt, was im konkreten Beispiel „SJ“ sein kann. Ist das kommunikative Ziel die Beschreibung einer Regel zur Ausführungszeit und wird über die „Org“-Funktion ermittelt, dass die organisatorische Ressource in einer Beziehung zum Nutzer steht, könnte auch eine Kombination aus Lexemen in den Baum eingefügt werden, welche z.B. die Phrase „ihr Vorgesetzter“ beschreibt. Schließlich entscheidet die Modalität der Regel über den letzten dynamischen Inhalt – das Modalverb im rechten Teil des Baums. Da es sich im konkreten Beispiel um eine optionale Regel, also eine Empfehlung handelt, ist das Ergebnis „should“. Übergibt man diese Nachricht direkt an den Oberflächenrealisierer, dann ist das Ergebnis der Generierung „The application should be approved by SJ.“ Falls das kommunikative Ziel die Erläuterung einer DPIL-Regel zum Ausführungszeitpunkt ist, könnte der Vorgang damit auch abgeschlossen sein. Das gilt aber nur dann, wenn die `direct`-Regel die einzige zu beschreibende Regel für den aktuellen Prozesszustand und den aktuellen Nutzer darstellt.

*Zwei Teile: Überblick und Beschreibung der Regeln*

Somit liegen nun zwei voneinander unabhängige, strukturierte Teilbeschreibungen des Zieltexts vor: *(i)* Der Überblick über das Modell und *(ii)* die Beschreibung der DPIL-Regeln. McKeown beschreibt in [123] einen schemabasierten Ansatz, welcher feste Text-Rahmenstrukturen und rhetorische Relationen kombiniert. Ein einfaches Schema, welches Überschriften vordefiniert und die beiden genannten Teilbeschreibungen an der korrekten Stelle einfügt, ist in Abbildung 6.8 skizziert.

Bis hierhin wird skizziert, wie für ein gegebenes DPIL-Modell und verschiedene Wertkonstellationen der übrigen Eingabeparameter DSyntR-Nachrichten erzeugt werden können. Dabei werden im ersten Schritt bereits vorhandene natürlichsprachliche Informationen analysiert und zur Anreicherung der Nachrichten

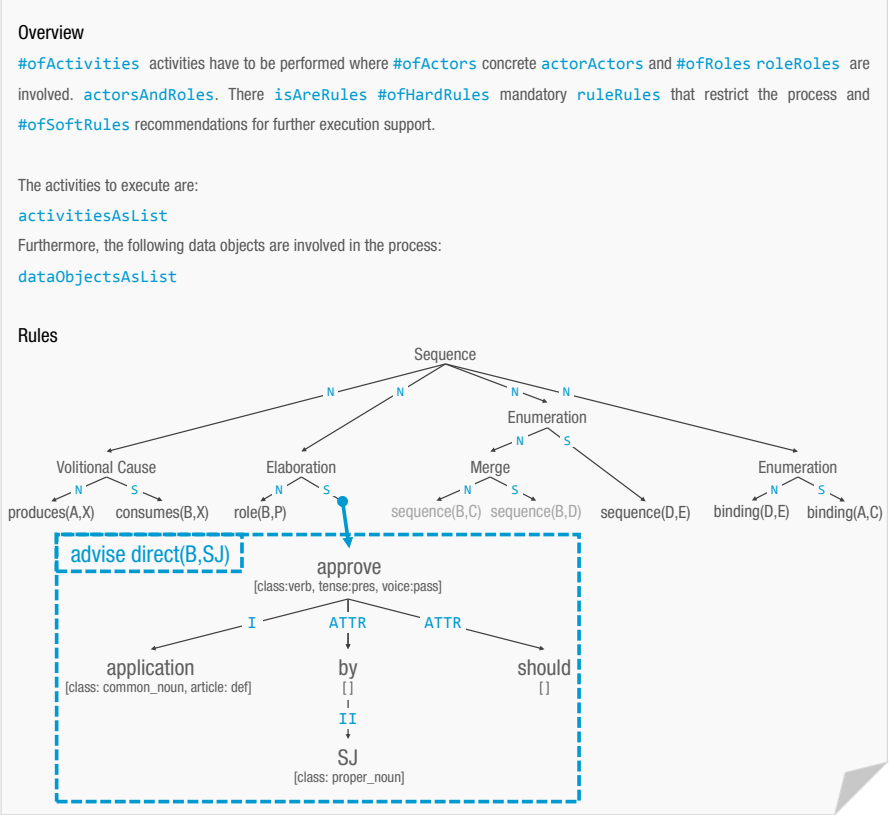



Abbildung 6.8: Beispiel-Dokumentplan als erstes Bindeglied zwischen DPIL-Modell (Codeausschnitt 6.1) und Zieltext (Abbildung 6.4)

.

*Textstrukturierung mittels RST-Baum*

verwendet. Eine dieser Analysen ist beispielsweise die Zerlegung von Aktivitätsbezeichnern in Substantive und Verben. Jede der DSyntR-Nachrichten repräsentiert die Spezifikation der natürlichsprachlichen Beschreibung *einer* DPIL-Regel. Letztere werden im zweiten Schritt der Dokumentplanung in einem RST-Baum organisiert. Dieser wird wiederum in ein Textschema eingebettet, welches die Rahmenstruktur des resultierenden Dokuments festlegt. Der daraus resultierende Dokumentplan – angedeutet in Abbildung 6.8 – ist das Ergebnis der ersten Phase der NLG-Pipeline und dient als Eingabe für die sich anschließende Phase der Mikroplanung. Als Vereinfachung wird in der Darstellung des Dokumentplans darauf verzichtet, alle DPIL-Regeln durch die entsprechende Nachrichtenschablone zu ersetzen. Der tatsächliche Dokumentplan enthält an dieser Stelle jedoch eine vollständige rhetorische Strukturierung der Nachrichten *aller* DPIL-Regeln.

### 6.2.3 *Mikroplanung: Lexikalisierung durch Ersetzung abfragebasierter Platzhalter und Satzgestaltung durch partielle linguistische Schablonen*

Die zu den einzelnen DPIL-Regeln gehörenden Nachrichten beschreiben individuelle natürlichsprachliche Sätze aber keinen zusammenhängenden Text. Eine bloße Aneinanderreihung dieser Sätze hätte eher den Charakter einer Aufzählung und würde keinen natürlich wirkenden Textfluss erzeugen. Enthält ein DPIL-Modell

beispielsweise zahlreiche `direct`-Regeln, dann würden diese immer mit der in Abbildung 6.7 dargestellten linguistischen Schablone beschrieben werden. Das Ergebnis wäre eine Form wie „The application should be approved by SJ. The reply should be checked by XY. The bill should be payed by LA…". Im aktuellen Abschnitt wird – erneut exemplarisch – beschrieben, wie der Textfluss automatisiert verbessert werden kann. Durch die mit der Verwendung von DSyntR-Nachrichten gewährleistete Abstraktion von der Domäne deklarativer Prozessmodelle ist es möglich, existierende und allgemeinere Verfahren zur Verbesserung des Textflusses anzuwenden. Dies beinhaltet, wie in [154] im Allgemeinen und [107] im Speziellen vorgeschlagen wird, die *Aggregation* von Nachrichten sowie die Einführung von *referenzierenden Ausdrücken* und von *Diskursmarkierungen*. Da es sich hierbei um Probleme der *Optimierung* der Textlesbarkeit handelt, jedoch keine generelle Voraussetzung für die Verständlichkeit des Textes darstellt, sind die nachfolgenden Erläuterungen, dem Rahmen der vorliegenden Arbeit entsprechend, knapp gehalten.

*Aggregation, referenzierende Ausdrücke und Diskursmarkierungen*

Unter *Aggregation* ist im NLG-Kontext die Abbildung der Dokumentstruktur aus dem Dokumentplan auf linguistische Strukturen und textuelle Elemente [154, S. 56] zu verstehen. Beispielsweise beschreibt der Dokumentplan aus Abbildung 6.8 eine *Elaboration*-Beziehung zwischen den Nachrichten für die Regeln `role(B,P)` und `direct(B,SJ)`. Ohne eine Aggregation wären die folgenden zwei aufeinanderfolgenden Sätze im resultierenden Text enthalten:

*Aggregation*

- „The application must be approved by a Professor."
- „The application should be approved by SJ."

Im exemplarischen Zieltext (Abbildung 6.4) ist jedoch folgende Form zu erkennen: „The application must be approved by a Professor and preferably by SJ." Nachfolgend wird ein einfaches Prinzip beschrieben, wie derartige Verschmelzungen natürlichsprachlicher Sätze mit inhaltlichen Überschneidungen automatisiert werden können.

Im Dokumentplan sind die Relationen zwischen einzelnen DSyntR-Nachrichten klassifiziert. Die Semantik dieser Klassen ist in der Theorie rhetorischer Strukturen festgelegt. Die oben genannte Beispiel-Relation *Elaboration* beschreibt demnach eine allgemeinere (Nukleus, hier: DSyntR-Nachricht für `ensure role(B,P)`) und eine spezifischere Regel (Satellit, hier: DSyntR-Nachricht für `advise direct(B,SJ)`) Regel. Die Verschmelzung der beiden Regeln ist in Abbildung 6.9 dargestellt.

Die Aggregation erfolgt hier musterbasiert und in zwei Phasen: *(i)* Erkennung des Aggregationspotentials und *(ii)* deren Durchführung. Für die beiden Beispielregeln ist ein Aggregationspotential erkannt worden, da eine *Elaboration*-Relation zwischen ihnen besteht und sie die Aktivität B (*Approve application*) als gemeinsames Argument aufweisen. Für die Aggregationsdurchführung werden vorab Fragmente in den DSyntR-Nachrichtenschablonen markiert, die für die jeweilige Regel-Konstellation geeignet sind. Diese sind in der Abbildung durch einen Rahmen hervorgehoben. Im zweiten Schritt der Aggregationsdurchführung wird ein neuer Knoten erzeugt, welcher im aktuellen Beispiel als *ATTR*-Zweig an den Wurzelknoten der Nachrichtenschablone für die `role`-Regel angehängt

*Musterbasiert, zweiphasig*

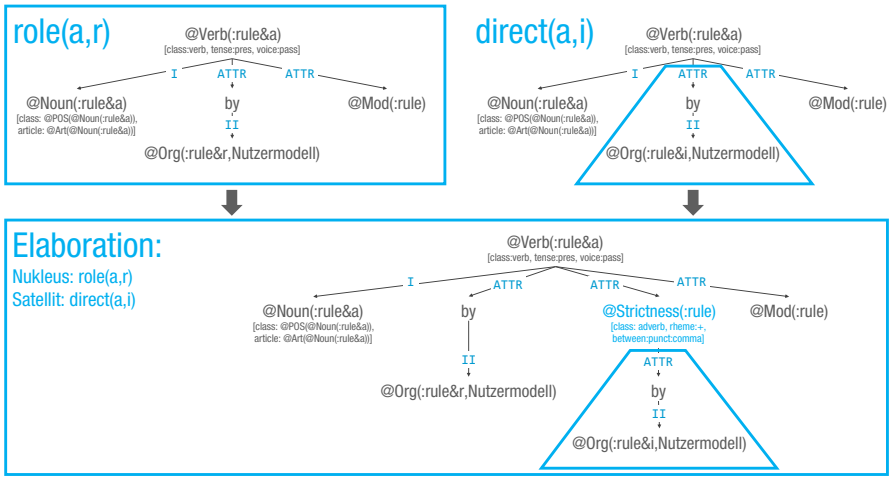



Abbildung 6.9: Aggregationsbeispiel für die Regeln `role` und `direct`

wird. Das zugehörige DSyntR-Fragment der `direct`-Nachrichtenschablone wird danach an diesen Knoten angefügt. Damit ist die Aggregation für das konkrete Beispiel abgeschlossen.

Die übrigen Aggregationen erfolgen analog. Die beiden verpflichtenden Regeln `produces(A,X)` und `consumes(B,X)` sind über eine *Volitional-Cause*-Relation verbunden. Das Aggregationspotential ist erneut durch ein gemeinsames Argument, das Datenobjekt X (*Application*), gegeben. Diese Relation beschreibt eine Ursache-Wirkung-Beziehung. Das Datenobjekt muss produziert werden, weil es für die Bewältigung von Aufgabe C (*Book flight*) benötigt wird. Das Aggregationsergebnis ist im ersten Satz der Regelbeschreibungen von Abbildung 6.4 dargestellt. Es suggeriert, dass in der Aggregationsdurchführung in diesem Fall die vollständigen DSyntR-Schablonen beider Nachrichten verknüpft werden. Dies wird hier über die Einführung eines neuen Knotens mit der Bezeichnung *because* realisiert. Unter selbigem wird die Nachrichtenschablone der `consumes`-Regel eingehängt. In dieser wird abschließend der Knoten, welcher das Datenobjekt benennt, durch ein Pronomen ersetzt. Dies erfolgt, ähnlich wie in der DSyntR-Schablone, für die `binding`-Regel über das Hinzufügen eines Grammems zum Lexem. In diesem Fall ist es das Grammem *pro:pro*, welches festlegt, dass der Knoten, der eigentlich ein Nomen beschreibt, als Pronomen realisiert werden soll. Diese sind sogenannte *referenzierende Ausdrücke*, da sie eine Referenz auf ein anderes Wort im Diskurs darstellen. Ohne diesen referenzierenden Ausdruck würde durch die Mehrfachnennung des Datenobjekts eine Wortwiederholung entstehen.

*Referenzierende Ausdrücke*

Die letzte Aggregation wird für die beiden Empfehlungsregeln `sequence(B,C)` und `sequence(B,D)` vorgenommen – auch hier wieder auf Basis gemeinsamer Aktualparameter. Die Nachrichtenschablonen sind hierbei identisch. Die vollständige Schablone bildet dabei eines der Fragmente für die Aggregationsdurchführung. Das andere ist der Teilbaum derselben Schablone, welcher aus dem Wurzelknoten und dem durch die `I`-Kante verbundenen Nomen-Knoten besteht. Die Verschmelzung erfolgt durch die Einführung einer Konjunktion als *COORD-Zweig* am Wurzelknoten des größeren Fragments.

Generell gilt im Fall der Aggregation von zwei Nachrichten, dass das Resultat, dass das Resultat den Knoten ersetzt, welcher die rhetorische Beziehung zwischen den verschmolzenen Nachrichten beschreibt.

*Diskursmarkierungen*

Als letzter Schritt zur Verbesserung der Lesbarkeit des Zieltexts können Diskursmarkierungen eingeführt werden. Das aktuelle Konzept beinhaltet auch dafür ein regelbasiertes Verfahren. Im Beispieltext (Abbildung 6.4) sind die Diskursmarkierungen „furthermore" und „additionally" verwendet worden. In dem in [107] beschriebenen NLG-Ansatz für imperative Prozessmodelle wird eine Liste mit geeigneten Markierungen vordefiniert und an geeigneter Stelle zufällig eingefügt. Dieses Prinzip wird für den aktuell beschriebenen Ansatz übernommen. Geeignete Stellen werden wiederum über die Art der Beziehungen zwischen Teilen der rhetorischen Struktur identifiziert. Im aktuellen Entwicklungsstand werden lediglich für Relationen des Typs *Enumeration* Diskursmarkierungen eingefügt. Diese werden aus einer Menge von drei Markierungen, bestehend aus „furthermore", „in addition" und „additionally", vollkommen zufällig ausgewählt. Dieses Verfahren ist stark vereinfacht und dient lediglich zur Bewertung der Realisierbarkeit derartiger Textmodifikationen.

### 6.2.4 *Oberflächenrealisierung: Umsetzung syntaktischer Abhängigkeitsbäume in natürlichsprachlichen Text*

Viele existierende Ansätze aus sehr unterschiedlichen Domänen zeigen, dass die Oberflächenrealisierung als weitestgehend domänenunabhängig betrachtet werden kann [107, 154]. Ziel dieses letzten Schrittes der NLG-Pipeline ist es, die vollständig lexikalisierten und möglicherweise aggregierten DSyntR-Nachrichten in ihre linguistische Oberflächenform zu überführen – in natürlichsprachliche Sätze.

*Schritt 1: Überführung in morphologische Repräsentation*

Die Transformation erfolgt über zwei Ebenen (Abschnitt 3.6.3). Die DSyntR-Nachrichten werden zunächst in ihre morphologische Repräsentation überführt. Dazu wird die Baumstruktur linearisiert. Die resultierende Reihenfolge der einzelnen Knoten des DSyntR-Baums richtet sich einerseits nach den Kantenbeschriftungen und andererseits nach den Grammemen der einzelnen Lexeme. Beispielsweise bedeuten die beiden Beschriftungen `I` und `II` in der Regel eine Subjekt-Objekt-Form. Zusätzlich werden beispielsweise die Grammeme, welche den Typ des Artikels eines Nomens bestimmen, zur Materialisierung des korrekten Artikels verwendet. Einzig die Grammeme, welche zur Bildung der Flexionsformen der einzelnen Wörter benötigt werden, bleiben in dieser Struktur erhalten. Die morphologische Repräsentation der in Abbildung 6.7 dargestellten `direct`-Regel ist beispielsweise: *The application should approve*$_{[\mathtt{pres,pass}]}$ *by SJ.* Im letzten Schritt werden die noch verbliebenen Grammeme zur Bildung der Flexionsformen verwendet. Die Grammeme [pres, pass] führen dazu, dass aus *approve* die Oberflächenform *be approved* wird.

*Schritt 2: Bildung der Flexionsformen*

Die Reihenfolge, in der die so entstandenen morphologischen Repräsentationen in den Zieltext überführt werden, entspricht der Reihenfolge der Blattknoten des Dokumentplans, beginnend mit dem äußerst linken Knoten. Die Typen der verbliebenen rhetorischen Relationen haben hier keine Relevanz mehr, bieten aber für

zukünftige Weiterentwicklungen das Potential für eine *semantische* Bestimmung der Reihenfolge der resultierenden natürlichsprachlichen Sätze.

Mit diesem letzten Schritt ist die Transformation des DPIL-Modells in einen natürlichsprachlichen Zieltext abgeschlossen.

### 6.2.5 *Ansatzspezifische Annahmen und Erweiterungsmöglichkeiten*

Die Untersuchungen des vorab beschriebenen Ansatzes zur Generierung natürlichsprachlicher Beschreibungen multi-perspektivischer, deklarativer Prozessmodelle (Abschnitt 6.2) haben den Charakter eines exemplarischen Durchstichs. Einige Herausforderungen wurden daher vereinfacht dargestellt oder ausgeblendet. Die dafür notwendigen, stützenden Annahmen werden im aktuellen Abschnitt festgelegt, lassen sich aber in einigen Fällen durch weiterführende Untersuchungen eliminieren. Zudem werden auch zwei Bedingungen für die Sinnhaftigkeit des Einsatzes der NLG-Technik beschrieben.

Im aktuellen Stadium der Analyse des beschriebenen NLG-Ansatzes müssen folgende Annahmen gelten, damit selbiger prinzipiell anzuwenden ist:

*Beschränkung des Regelkatalogs*
- Das mittels natürlicher Sprache zu beschreibende DPIL-Modell beinhaltet ausschließlich Regeln, für die in Tabelle 6.2 eine Abbildungsregel auf eine DSyntR-Nachricht verfügbar ist.

*Festes organisatorisches Metamodell*
- Organisationsstrukturen sind auf Basis des organisatorischen Metamodells nach Bussler [38] definiert.

*Festes Format für Aktivitätsbezeichner*
- Bezeichner von Aktivitäten eines DPIL-Modells enthalten immer eine Kombination aus Verb und Nomen, welche das Prädikat und das Objekt eines natürlichsprachlichen Satzes bilden können. Beispiele für valide Kombinationen sind somit *approve application* oder *book flight*. Gegenbeispiele sind unter anderem *approve*, *go forward* oder *delivery*. Diese Annahme entspricht auch gängigen Richtlinien für die Prozessmodellierung [126].

*Festes Format für Relationsbezeichner*
- Für die Bezeichner von Relationen zwischen Organisationseinheiten (DPIL-Prädikate) wird eine Schreibweise mit Binnenmajuskel verwendet und setzt sich aus einem Hilfsverb und einem Nomen zusammen. Valide Beispiele sind somit *hasRole* oder *isParent*. Relationsbezeichner wie *memberOf* und *supervisorOf* werden zwar von DPIL, nicht aber von der hier vorgestellten NLG-Technik unterstützt.

*Verlust der Eindeutigkeit akzeptabel*
- Der Verlust der Eindeutigkeit, welcher zwangsläufig mit dem Wechsel von einer formalen zu einer natürlichen Sprache einhergeht, wird vom Nutzer akzeptiert.

*Vollständigkeit vor sprachlicher Flexibilität*
- Die Vollständigkeit der natürlichsprachlichen Beschreibung wird als wichtiger eingestuft als sprachliche Individualität und ein natürlich anmutender Textfluss.

Durch die oben genannten, fundamentalen Annahmen und Einschränkungen des NLG-Ansatzes liegen bereits zahlreiche Erweiterungsmöglichkeiten auf der Hand.

Die starre Abbildung zwischen statischen DPIL-Regelschablonen auf linguistische Nachrichten bietet den Vorteil einer geringen Fehleranfälligkeit. Allerdings kann auf Basis statischer Abbildungen nie *vollständig* garantiert werden, dass ein DPIL-Modell in natürliche Sprache transformiert werden kann. Grund dafür ist die Möglichkeit der Schachtelung Boolescher Ausdrücke, welche theoretisch die Definition einer unendlichen Anzahl unterschiedlicher Regelschablonen ermöglicht. Aus diesem Grund ist die Abbildung atomarer Teilausdrücke der Sprache DPIL auf beispielsweise partielle DSyntR-Schablonen denkbar – und für das Erreichen einer garantiert vollständigen Transformation auch notwendig. Solange durch die Kombination dieser partiellen Nachrichten eine vollständige DSyntR-Nachricht erzeugt werden kann, kann der verbleibende Teil der NLG-Pipeline beibehalten werden.

*Erweiterung: Abbildung atomarer Teilausdrücke*

Das für die Modellierung von Organisationsstrukturen angenommene Metamodell zeichnet sich durch große Flexibilität aus. Allerdings existieren keine Daten über die praktische Verbreitung und Übertragbarkeit seiner der Konzepte. Aus diesem Grund ist seine Austauschbarkeit von zentraler Bedeutung. Im vorab vorgestellten Konzept wird dies dadurch berücksichtigt, dass in den Anfragen für Lexeme und Grammeme vom konkreten Metamodell abstrahiert wird. Beim Austausch desselben muss somit lediglich jede beschriebene Anfrage auf die Struktur des gewünschten Ziel-Metamodells abgebildet werden. Voraussetzung dafür ist, dass letzteres eine vergleichbare Ausdrucksmächtigkeit aufweist, wie das bisher angenommene Metamodell.

*Austausch des Metamodells*

Auch im Hinblick auf die Verbesserung der linguistischen Flexibilität kann der Ansatz an zahlreichen Stellen erweitert werden. Beispielsweise ist die Einschränkung der Bezeichner von Aktivitäten und Relationen auf ein Verb-Nomen- respektive Hilfsverb-Nomen-Muster nur getroffen worden, da der Einsatz von Techniken zur Verarbeitung natürlicher Sprache für die Analyse *beliebiger* Aktivitätsbezeichner nicht betrachtet wird. Werkzeuge zur Erkennung von Wortarten, Abhängigkeiten zwischen Wörtern sowie Wortbedeutungen weisen zwar bei zu kurzen Textabschnitten starke Schwächen auf, jedoch sind diese Techniken generisch konzipiert und nutzen in der Regel kein Domänenwissen. Hier wäre folglich der Grad der Verbesserung der Analyse von natürlichsprachlichen Prozessmodellanteilen unter Einbeziehung des Kontextwissens der Prozessdomäne zu untersuchen. Beispielsweise ist die Wahrscheinlichkeit hoch, dass ein Aktivitätsbezeichner ein Verb enthält.

*Nutzung von Domänenwissen für flexiblere Bezeichner*

Abseits der Aufhebung derzeit getroffener Einschränkungen ist auch die Erweiterung der Anwendungsdomäne der Technik zu berücksichtigen. Aktuelle NLG-Systeme unterscheiden beispielsweise zwischen verschiedenen kommunikativen Zielen. Die vorab beschriebene Technik unterstützt hier einerseits die Erklärung eines Prozessmodells zum Modellierungs- aber auch die Erläuterung von Modellteilen zum Ausführungszeitpunkt. Diese Generierung nutzt als Informationsquelle für relevante Prozessregeln ausschließlich das Prozessmodell. Liegen historische Aufzeichnungen von Prozessausführungen vor, so ist eine mögliche Erweiterung die Filterung der Regeln, die für diese historischen Aufzeichnungen irrelevant sind. Dies kann dazu genutzt werden, die Menge der Regeln einzuschränken, für die natürlichsprachliche Beschreibungen erzeugt werden sollen. Einerseits ist es hier möglich, die NLG-Technik auf die Erläuterung *prototypischer* Prozessaus-

*NLG für prototypische oder invalide Prozessausführungen*

führungen zu fokussieren und andererseits kann sie zur Erklärung identifizierter *Abweichungen* von dem im Modell spezifizierten Verhalten eingesetzt werden.

Ziel dieses Kapitels ist ein exemplarischer Durchstich für die Konzeption eines Systems zur Generierung natürlichsprachlicher Prozessbeschreibungen für multi-perspektivische, deklarative Prozessmodelle. Existierende Verfahren eignen sich zu großen Teilen lediglich für imperative Prozessmodelle. Durch den üblicherweise stark modularen Aufbau derartiger Systeme können jedoch einige Komponenten wiederverwendet werden. Als größte Herausforderungen sind vor allem die Sortierung und Gruppierung der in deklarativen Prozessmodellen enthaltenen Regeln zu bewältigen. Dies ist notwendig, um den zu generierenden Text sinnvoll zu strukturieren. Imperative Modelle basieren grundsätzlich auf der Semantik gerichteter Graphen, sodass eine Struktur und Reihenfolge der Beschreibung des Modells nativ gegeben ist. Durch Vergleiche der Bestandteile der Regeln deklarativer Prozessmodelle wird diese Struktur künstlich erzeugt. Durch das Bedeutung-Text-Modell und die Theorie rhetorischer Strukturen ist zudem ein Austauschformat der Ergebnisse der Zwischenschritte des NLG-Systems definiert. Dies bietet das Potential für den Austausch oder die Optimierung einzelner Komponenten. Eine zugehörige Implementierung zeigt die Realisierbarkeit dieses Gesamtkonzepts (Abschnitt 7.5).

# Teil III

# IMPLEMENTIERUNG, EVALUATION UND ZUSAMMENFASSUNG

Im dritten Teil der vorliegenden Arbeit wird die prototypische Implementierung der im vorherigen Teil beschriebenen Lösungskonzepte vorgestellt. Diese Implementierungen werden gleichzeitig zur Evaluation der Ansätze instrumentalisiert. Weiterhin wird für den bereits beschriebenen Spurgenerator und auch für das generische Translationsprinzip für Prozessmodelle die korrekte Funktionsweise geprüft. Neben anderen ist die Verbesserung der Verständlichkeit von Prozessmodellen einer der wesentlichen Einsatzzwecke aller drei im vorherigen Teil vorgestellten Techniken. Bezüglich ihrer Nützlichkeit bietet eine Umfrage mit Repräsentanten der Zielgruppe erste, qualitative Einsichten. Aufbauend auf den Erkenntnissen aller Teile der Evaluation werden die nächsten Schritte im Problembereich der vorliegenden Arbeit und auch Anknüpfungspunkte potentieller Folgearbeiten diskutiert.

# 7

# ECLIPSE- UND RAPIDMINER-BASIERTE PROTOTYPISCHE IMPLEMENTIERUNG

In Teil ii wird das Konzept für die drei Techniken MuDePS, SiMiTra und NL4DP vorgestellt. Ziel des aktuellen Kapitels ist es, einerseits die Realisierbarkeit der ausgearbeiteten Konzepte durch die Beschreibung einer prototypischen Implementierung nachzuweisen. Andererseits stellt das Kapitel auch eine Anleitung zur Bedienung und Erweiterung der Technologien zur Verfügung, um in nachfolgenden Forschungsarbeiten auf diesen Prototypen aufbauen zu können.

Zunächst werden zwei Plattformen vorgestellt, in welche die Prototypen integriert werden (Abschnitt 7.1 und Abschnitt 7.2). Die im Zuge dessen entworfenen Architekturen sowie Bedienung und Erweiterungsmechanismen für die drei Kernbeiträge dieser Arbeit werden in folgender Reihenfolge beschrieben:

- Abschnitt 7.3: Prototypische Umsetzung des MuDePS-Spurgenerators für multi-perspektivische, deklarative Prozessmodelle (Konzept: Abschnitt 4.2) als Erweiterung einer Data-Mining-Plattform, *Implementierung: Spurgenerator*
- Abschnitt 7.4: Umsetzung des SiMiTra-Translationsansatzes für Prozessmodellierungssprachen (Konzept: Abschnitt 5.2) als Analyse-Prozessschablonen einer Data-Mining-Plattform, *Implementierung: Translationsprinzip*
- Abschnitt 7.5: Prototypische Umsetzung des NL4DP-NLG-Systems für multi-perspektivische, deklarative Prozessmodelle (Konzept: Abschnitt 6.2) als Erweiterung eines für die Modellierung derartiger Modelle verwendeten Editors. *Implementierung: NLG-Technik*

Jede der drei Realisierungen sowie einige Beispieldaten stehen unter `http://mps.kppq.de` zum Download zur Verfügung. Für die Verwendung der Werkzeuge können große Teile dieses Kapitels übersprungen werden. Einzig die mit Bedienung bezeichneten Abschnitte sind als komprimiertes Handbuch für die jeweilige Technik zu verstehen und sollten dementsprechend als Einstiegshilfe verwendet werden.

## 7.1 DAS ECLIPSE MODELING FRAMEWORK, OCL UND ACCELEO: EIN RAHMENWERK FÜR MODELLGETRIEBENE SOFTWAREENTWICKLUNG

Die Techniken MuDePS und NL4DP verwerten ein DPIL-Modell. Das in Abschnitt 3.1.5 vorgestellte Metamodell der deklarativen Prozessmodellierungssprache DPIL ist mittels des *Eclipse Modeling Frameworks (EMF)* [172] entwickelt worden. Im aktuellen Abschnitt werden Technologien beschrieben, welche derartige Modelle verarbeiten können. Diese Technologien sind die Basis für die im Rahmen dieser Arbeit entwickelten Ansätze zur Spurgenerierung (Abschnitt 4.2) und Generierung natürlichsprachlicher Prozessbeschreibungen (Abschnitt 6.2). *Eclipse Modeling Framework*

EMF ist ein Modellierungsrahmenwerk aus dem Eclipse-Umfeld, das sich an Teilen des OMG-Standards *Meta Object Facility (MOF)* orientiert. Es beinhaltet unter anderem ein generisches, selbstbeschreibendes Meta-Metamodell – *Ecore* genannt – sowie Mechanismen zur Generierung von Java-Code. Ersteres erlaubt die Spezifikation von strukturellen[1] Metamodellen, die beispielsweise als abstrakte Syntax von DSLs dienen können. Die Generierung von Java-Code erlaubt die automatische Übersetzung der Modelle in ausführbaren Struktur-Code sowie die Erzeugung eines Standard-Editors zur Erstellung von Modellen, welche Instanzen des mit Ecore spezifizierten Metamodells sind. Dieser Editor wird wiederum als Eclipse-Erweiterung generiert, welcher über die Standard-Erweiterungsmechanismen in die Entwicklungsumgebung eingebunden werden können. Dabei machen sich die generierten Editoren die Eclipse-Infrastruktur für beispielsweise Syntaxhervorhebung, Referenzauflösung und Autovervollständigung zunutze. Zudem implementiert EMF das ebenfalls standardisierte Austauschformat *XML Metadata Interchange (XMI)* und zugehörige Serialisierungs-, Deserialisierungs und Persistenzmechanismen. Folglich kann jedes Modell, das konform zu einem Metamodell ist, welches wiederum konform zum Ecore-Meta-Metamodell ist, automatisch persistent gespeichert werden. Der für die vorliegende Arbeit zentrale Aspekt ist jedoch, dass das EMF Schnittstellen bereitstellt, die von anderen Technologien genutzt werden können, um ein Modell in eine gewünschte Zielrepräsentation zu transformieren. In der vorliegenden Arbeit werden ausschließlich Modell-zu-Text-Transformationen spezifiziert. Dies ermöglicht die Übersetzung beliebiger Modelle in ebenso beliebige textuelle Repräsentationen.

*Ecore*

Für die Entwicklung von Modell-zu-Text-Transformationen im EMF-Umfeld wurde von der OMG ein weiterer Standard verabschiedet – die sogenannte *MOF Model to Text Transformation Language (Mof2Text)* [140]. Dieser Standard beschreibt eine auf Schablonen basierende Transformationssprache für die Umwandlung von EMF-basierten Modellen in Text. Mit *Acceleo*[2] existiert auch eine Implementierung, welche wiederum als Eclipse-Erweiterung vorliegt. Damit lässt sich in Eclipse-Entwicklungsumgebungen eine vollständige Werkzeugkette von der Definition eines neuen Metamodells und Modell-zu-Text-Transformationen, bis hin zur operativen Verwendung des neuen Metamodells zur Erzeugung von Modellen und deren Transformation in beliebige Textrepräsentationen entwickeln. Dieser Umstand wird in der vorliegenden Arbeit einerseits genutzt, um die Abbildung von DPIL-Modellen auf Alloy-konforme textuelle Repräsentationen zum Zwecke der Simulation von deklarativen Prozessmodellen zu implementieren. Andererseits dient derselbe Mechanismus zur Realisierung des ebenfalls in der vorliegenden Arbeit beschriebenen NLG-Ansatzes, um für DPIL-Modelle natürlichsprachliche Beschreibungen zu erzeugen. Acceleos Syntax wird in den entsprechenden Implementierungsabschnitten falls notwendig am konkreten Beispiel erläutert.

*Mof2Text*

*Acceleo*

Orthogonal zum eben beschriebenen Standard für Modell-zu-Text-Transformationen existiert, ursprünglich als Erweiterung für die UML angedacht, die *Object Constraint Language (OCL)* [188]. OCL ist eine deklarative Abfrage- und Regel-

*OCL*

1 Strukturelle Modellierung beschreiben statische Zusammenhänge zwischen Entitäten, wie beispielsweise in Klassendiagrammen. Der Gegenpart wird Verhaltensmodellierung genannt, wozu beispielsweise auch die Prozessmodellierung zählt.

2 `http://www.eclipse.org/acceleo`, zuletzt aufgerufen: 13.04.2017

sprache, die sich besonders in Anwendungsbereich der Modelltransformationen etabliert hat. Sie erlaubt es einerseits, Abfragen an MOF-kompatible Metamodelle und Modelle zu formulieren. Andererseits bietet sie auch Sprachkonstrukte zur Formulierung von Regeln, welche *nicht*-strukturelle Beschränkungen von Metamodellen und Modellen beschreiben können. Beispielsweise bietet EMF die Möglichkeit, einer Entität ein Attribut zuzuordnen, welches einen ganzzahligen Datentyp aufweist. Möchte man den Wertebereich dieses Attributs jedoch auf den Bereich der *positiven* Ganzzahlen beschränken, dann fehlen in EMF die Beschreibungsmöglichkeiten dafür. OCL komplementiert an dieser Stelle das rein strukturelle Meta-Metamodell Ecore. Die Verwendung von OCL ist jedoch nicht nur auf statische Einschränkung von Attributwerten ausgerichtet, sondern ist beispielsweise auch in der Lage, dynamische Zusammenhänge zu modellieren. Angenommen, es werden in einem Metamodell die Entitäten A, B, C und D modelliert, wobei A ein ganzzahliges Attribut aufweist, B abstrakt ist und C und D Spezialisierungen der Entität B sind. A weist zudem eine Referenz zu Entität B auf, sodass in einer Instanz des Metamodells – also in einem Modell – Instanzen der Entität A mit Instanzen der Entitäten C und D assoziiert werden können. Bis zu diesem Punkt ließen sich diese Anforderungen mit den rein strukturellen Beschreibungsmitteln von EMF ohne weiteres umsetzen. Anders verhält es sich bei folgender beispielhafter Erweiterung der Anforderungen: Die Entscheidung, ob eine Instanz von C oder eine Instanz von D mit einer Instanz von A assoziiert wird, soll davon abhängig sein, ob das ganzzahlige Attribut der A-Instanz positiv oder negativ ist. Mit EMF ließe sich diese Anforderung nicht umsetzen. OCL bietet jedoch die notwendigen Sprachmittel für Abfragen bezüglich der Modellstruktur, der Instanztypen und auch für die Formulierung der datenabhängigen Einschränkungen von Assoziationen.

*OCL als Anfrage- und Regelsprache in Acceleo*

OCL wird auch von Acceleo verwendet. Letzteres stellt Sprachmittel für die deklarative Formulierung *struktureller* Suchmuster in Modellen bereit. Acceleo gestattet für *nicht-strukturelle* Suchmuster die Definition von OCL-Ausdrücken. Diese werden in den Acceleo-Abbildungsregeln zur Modell-zu-Text-Transformation benötigt. Dabei wird die Modellhierarchie durchlaufen und jedes angetroffene Modellelement wird mit den Suchmustern abgeglichen. Zur Aktivierung der zugehörigen Transformationsregel, muss das Modellelement sowohl zum strukturellen Suchmuster passen als auch die in OCL formulierten Einschränkungen befolgen.

Da die formalen Spezifikationen der drei oben beschriebenen Technologien sehr umfangreich sind, wird an dieser Stelle auf die angegebene Literatur verwiesen.

Sowohl MuDePS (Abschnitt 4.2) als auch NL4DP (Abschnitt 6.2) erfordert die Transformation eines DPIL-Modells in eine andere Repräsentation. Ziel dieses Abschnitts ist es, eine Kombination aus Technologien zu identifizieren, welche zur Implementierung dieser Transformationen verwendet werden können. Da DPIL auf dem Ecore-Metamodell basiert, ist eine Auswahl darauf aufbauender Technologien naheliegend. Diesbezüglich bildet Eclipse die Implementierungsplattform und eine generische Nutzerschnittstelle. Acceleo baut auf dieser auf und gestattet die Definition von Modell-zu-Text-Transformationen auf Basis von Schablonen. Dies geschieht auf Basis von strukturellen Mustervergleichen von DPIL-Modellelementen und Schablonen-Mustern. OCL ergänzt die Möglichkeiten des Strukturvergleichs durch Hilfsmittel zur Formulierung zusätzlicher, dynamischer Prüfbedingungen.

*Eclipse als Implementierungsplattform*

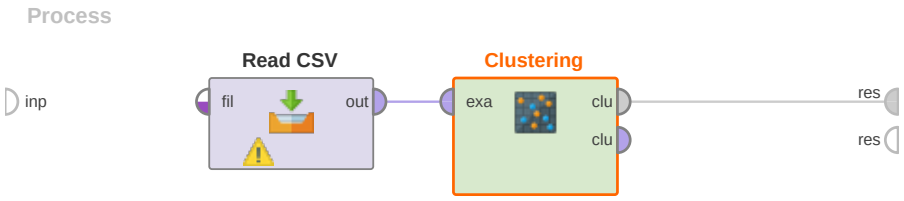



Abbildung 7.1: Operatoren im RapidMiner-Prozess

## 7.2 RAPIDMINER: QUELLOFFENE PLATTFORM FÜR DIE ORGANISATION KOMPLEXER ARBEITSABLÄUFE IM BEREICH DATA SCIENCE

*Rapidminer*

*RapidMiner*[3] ist eine quelloffene Plattform für die Bearbeitung von Data-Mining-Problemen [105]. Diese bietet Unterstützung für alle Schritte des maschinellen Lernens. Dazu zählen beispielsweise die Vorbereitung der zu analysierenden Daten, die Verarbeitung der Daten, die Präsentation der Ergebnisse sowie Schritte zur Validierung und Optimierung. Sowohl der in Abschnitt 5.2 vorgestellte Translationsansatz für Prozessmodelle als auch Teile des in Abschnitt 4.2 beschriebenen Konzepts eines Spurgenerators werden als Erweiterungen dieser Plattform umgesetzt.

*Prozessdiagramme*

In RapidMiner werden die Arbeitsabläufe zur Bewältigung von Data-Mining-Problemen in Form von *Prozessdiagrammen* grafisch modelliert. Nachdem der gewünschte Analyseprozess modelliert ist, kann dieser gestartet und observiert werden, bis der Prozess endet und zahlreiche Möglichkeiten der Ergebnisdarstellung angeboten werden. Die Grundbausteine jedes dieser Prozesse sind sogenannte *Operatoren*, wie sie in Abbildung 7.1 dargestellt sind.

*Operatoren*

Operatoren bilden die kleinste funktionelle Einheit im RapidMiner-Arbeitsprozess. Sie setzen sich aus den folgenden Bestandteilen zusammen [105]:

*Eingabe- und Ausgabe-Pins*

- *Pins:* Pins dienen zur Kommunikation mit anderen Operatoren. Eingabe-Pins (engl. *input pins*) ermöglichen das Empfangen von Daten. Ausgabe-Pins (engl. *input pins*) stellen die Ergebnisse der Anwendung des Operators zur Weiterverarbeitung zur Verfügung. Verbindet man den Ausgabe-Pin eines Operators mit dem Eingabe-Pin eines anderen Operators, entsteht ein Datenfluss. Ein Operator kann über beliebig viele Pins verfügen. Jeder Pin ist getypt, um die formale Kompatibilität zweier Operatoren prüfen zu können. In Abbildung 7.1 nimmt der Operator „Read CSV“ die Rolle des Datenproduzenten für den konsumierenden Operator „Clustering“ ein.

*Datentransformation*

- *Transformation:* Jeder Operator transformiert Daten. Diese können über einen Eingabe-Pin bereitgestellt werden oder von prozessexternen Quellen bezogen werden. In der Beispielabbildung (Abbildung 7.1) kann ist die Transformation des linken Operators das Einlesen einer CSV-Datei und die Überführung der Daten in die RapidMiner-interne Datenstruktur. Der rechte Operator nutzt diese Daten zur Clusteranalyse; die Transformation ist die Umwandlung kommaseparierter Daten in ein Clustermodell.

3 `http://rapidminer.com/`, zuletzt aufgerufen: 14.04.2017

- *Parameter:* Die Operatoren sind in vielen Fällen parametrierbar. Der Operator „Read CSV" verfügt beispielsweise über einen Pfadparameter, der zu der zu extrahierenden CSV-Datei führt. Das kann genutzt werden, wenn die Daten aus einer prozessexternen Quelle bezogen werden sollen. Weiterhin enthält der Operator einen Parameter zur Konfiguration des Trennzeichens zwischen den einzelnen Datenspalten der CSV-Datei. *Parameter*

Spezielle Operatoren können andere Operatoren *beinhalten*. Diese werden *Superoperatoren* genannt, welche die Modellierung von Subprozessen ermöglichen. Jeder RapidMiner-Prozess kann selbst als Operator betrachtet werden. In Abbildung 7.1 sind aus diesem Grund am linken und rechten Rand ebenfalls Eingabe- respektive Ausgabe-Pins zu erkennen. *Superoperatoren*

Die RapidMiner-Plattform unterstützt den Datenanalysten während des Prozessdesigns auch bei der Vermeidung syntaktischer Fehler. Wie bereits erwähnt, sind Pins getypt. Stimmt der Typ des Ausgabe-Pins eines Operators nicht mit dem geforderten Typ des angeschlossenen Eingabe-Pins überein, signalisiert die Plattform dem Nutzer einen Fehler. Der im linken Operator von Abbildung 7.1 angezeigte Fehler rührt allerdings daher, dass weder der Eingabe-Pin des Operators noch sein Pfadparameter mit einer gültigen CSV-Datei verknüpft sind.

Operatoren und die Möglichkeit, diese zu verknüpfen, sind die wichtigsten Bausteine bei der Verwendung der RapidMiner-Plattform. Sowohl der in Abschnitt 4.2 beschriebene Spurgenerator als auch das Translationskonzept aus Abschnitt 5.2 werden als Operator respektive Operator-Kombinationen implementiert.

Ziel des aktuellen Abschnitts ist es, eine Plattform zu identifizieren, in welcher das generische Translationsprinzip (Abschnitt 5.2) realisiert werden kann. Hierbei ist es wichtig, ohne größeren Aufwand, beliebige Spurgeneratoren mit ebenso beliebigen Process-Mining-Werkzeugen als Komponenten eines Übersetzungssystems kombinieren zu können. Die in diesem Abschnitt beschriebene RapidMiner-Plattform unterstützt nativ die Kommunikation zwischen sogenannten Operatoren, welche beliebige Funktionalität kapseln können. Diese Operatoren lassen sich beliebig kombinieren, solange sie bezüglich des Typs von Ein- und Ausgabedaten konform sind. Da Spurgeneratoren und Process-Mining-Werkzeuge Ereignisprotokolle erzeugen respektive verarbeiten, ist eine solche Prüfung der Typenkonformität gewünscht. Folglich eignet sich die RapidMiner-Plattform als Rahmenwerk für eine Umsetzung des generischen Übersetzungsprinzips (Abschnitt 7.3).

## 7.3 MUDEPS ALS RAPIDMINER-ERWEITERUNGEN

Für das in Kapitel 4 vorgestellte Konzept eines Spurgenerators für multi-perspektivische, deklarative Prozessmodelle – MuDePS (Abbildung 7.2) – wird nachfolgend die zugehörige technische Umsetzung beschrieben.

MuDePS ist als Operator-Erweiterung der in Abschnitt 7.2 beschriebenen Data-Mining-Plattform realisiert. Die Gründe für diese Entscheidung sind:

- MuDePS soll auch für den in Kapitel 5 beschriebenen Translationsansatz für Prozessmodelle als Komponente verwendet werden können. Die Umsetzung dieses Translationsansatzes – SiMiTra – basiert auf der RapidMiner-Plattform, was in Abschnitt 7.4 begründet wird. *Abhängigkeit von SiMiTra*

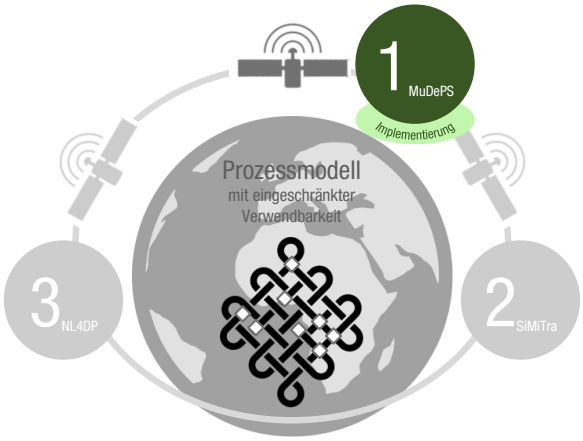



Abbildung 7.2: MuDePS (Implementierung)

*Grafische Nutzeroberfläche und Operator-Verknüpfung*

- RapidMiner bietet ein gut dokumentiertes und schlankes Erweiterungskonzept an, was im Architekturteil dieses Abschnitts näher erläutert wird. Ein wesentlicher Vorteil hierbei ist, dass lediglich die Kernfunktionalität des Spurgenerators sowie deren Anbindung an Schnittstellen der RapidMiner-Plattform implementiert werden müssen. Eine grafische Nutzeroberfläche und die Möglichkeit der Anbindung an andere Operatoren ist dadurch ohne weitere Arbeitsschritte möglich.

*Wiederverwendung von RapidMiner-Operatoren zur Simulationsdurchführung*

- RapidMiner verfügt über eine Reihe von (Super-)Operatoren, welche eine Automatisierung größerer Simulationsdurchführungen erleichtern. Dazu zählt beispielsweise der *Schleifen-Operator* (engl. *Loop operator*). Mit dessen Hilfe können unter anderem Ereignisprotokolle für unterschiedliche maximale Spurlängen erzeugt werden, ohne diese Funktionalität eigens implementieren zu müssen.

*Existierende Process-Mining-Operatoren*

- Es existieren bereits RapidMiner-Erweiterungen für einige Process-Mining-Technologien der ProM-Plattform [121], welche in der Erweiterung *RapidProM* gekapselt sind. Dazu gehören unter anderem Operatoren zur Vorverarbeitung der Ereignisprotokolle wie beispielsweise ein Operator zur Erzeugung eines künstlichen Ereignisses für Prozessstart und -ende. Auch eine informative Präsentationsfunktionalität für Ereignisprotokoll ist in dieser Erweiterung enthalten. Durch die Umsetzung von MuDePS als RapidMiner-Erweiterung können die Funktionalität des Spurgenerators und der Operatoren zur Spurmodifikation leicht kombiniert werden.

Die aktuelle Umsetzung des Spurgenerators erlaubt die Generierung von Prozessausführungsspuren auf Basis eines gegebenen DPIL-Modells. Wie in Abschnitt 4.2 erläutert, muss dieses jedoch zunächst auf eine zu Alloy konforme Repräsentation überführt werden. Dies ist mit Hilfe des in Abschnitt 7.1 beschriebenen Modell-zu-Text-Transformationsrahmenwerks Acceleo realisiert. Da diese Transformation zum aktuellen Zeitpunkt – im Sinne einer prototypischen Entwicklung – eine Eins-zu-Eins-Abbildung gemäß der im Konzeptteil aufgeführten Abbildungstabellen (Tabelle 4.3 in Abschnitt 4.2.4) darstellt, wird in der aktuellen Implementierungsbeschreibung auf eine detailliertere Betrachtung verzichtet.

Grundlegend basiert die Implementierung dieser Abbildung jedoch auf Schablonen des zu generierenden Zieltexts – in diesem Fall Alloy-Codeausschnitte – und diese setzen sich wie in Codeausschnitt 7.1 angedeutet zusammen.

*Einbettung von Alloy-Schablonen in Acceleo-Schablonen*

```
[template public rule(r : ProcessRule) ? <Guard> post (trim())]
   <Alloy Schablone>
[/template]
```

Codeausschnitt 7.1: Transformation von DPIL-Modellen in Alloy-konforme Repräsentation mittels Acceleo

Eine Modell-zu-Text-Transformation kann in Acceleo rein deklarativ gestaltet sein. Das bringt den Vorteil kompakterer Beschreibungen bei der strukturellen Mustersuche. Anstelle von zahlreichen, geschachtelten *IF-THEN-ELSE*-Blöcken imperativer Sprachen wird in Acceleo eine Schablone definiert, welche Modellelemente als Parameter akzeptiert. Allen Schablonen für die Transformation eines DPIL-Modells in die Sprache Alloy erlauben genau einen Parameter vom Typ *ProcessRule* (Abschnitt 3.1.5) und werden *rule* genannt. In klassischen objektorientierten Programmiersprachen würde der Compiler hier einen Fehler erkennen, da sich die „Methodensignaturen" nicht unterscheiden. Acceleo bietet hier aber die Möglichkeit, zusätzlich sogenannte *Guard*-Ausdrücke zu verwenden. Diese Ausdrücke werden mittels OCL formuliert und dienen der strukturellen Mustersuche. Diese gleicht die Regel-Rümpfe der in Abschnitt 4.2 beschriebenen Makros mit den im jeweiligen Prozessmodell auftretenden Regeln ab. Stimmen in einem Fall Vorlage und konkrete Instanz überein, dann wird im Zieltext der Rumpf der Acceleo-Schablone eingefügt. Dieser ist im oben skizzierten Beispiel mit *<Alloy Schablone>* markiert. Diese Schablonen sind Aufrufe des mittels struktureller Mustersuche ermittelten Alloy-Prädikats. Die derzeit unterstützten Prädikate sind in Tabelle 4.3 von Abschnitt 4.2.4 dargestellt. Zusätzlich zu diesem Aufruf wird, falls nicht bereits vorhanden, das Prädikat selbst in den generierten Alloy-Code eingefügt. Ist die strukturelle Mustersuche ergebnislos, wird die entsprechende DPIL-Konstruktion ignoriert. Die Abbildung der Prozessentitäten, wie beispielsweise Aktivitäten, Ressourcen und Datenobjekte erfolgt nach demselben Prinzip.

*Selektion von Schablonen durch Guard-Ausdrücke*

MuDePS selbst sowie einige Beispieldaten sind unter `http://mps.kppq.de` zum Download verfügbar.

### 7.3.1 *Architektur*

Die MuDePS-Architektur muss einerseits den Anforderungen der RapidMiner-Plattform genügen und andererseits die Implementierung von Alloy Spezifikationssprache für Simulationsmodelle bedienen. Im aktuellen Abschnitt wird eine mögliche und gleichzeitig die im Rahmen der vorliegenden Arbeit verwendete Architektur vorgestellt.

In Abbildung 7.3 sind die wesentlichen Architekturbestandteile der MuDePS-Implementierung als UML-2.0-Klassendiagramm abgebildet, wobei die konkrete Umsetzung auf Java basiert. Die grauen Klassen und Pakete des Diagramms gehören dabei entweder zur Erweiterungsschnittstelle der RapidMiner-Plattform oder zur Alloy-Implementierung. Die übrigen Modellelemente (blau) stellen dagegen die eigens entwickelten Bestandteile der MuDePS-Umsetzung dar.

*Gesamtarchitektur*

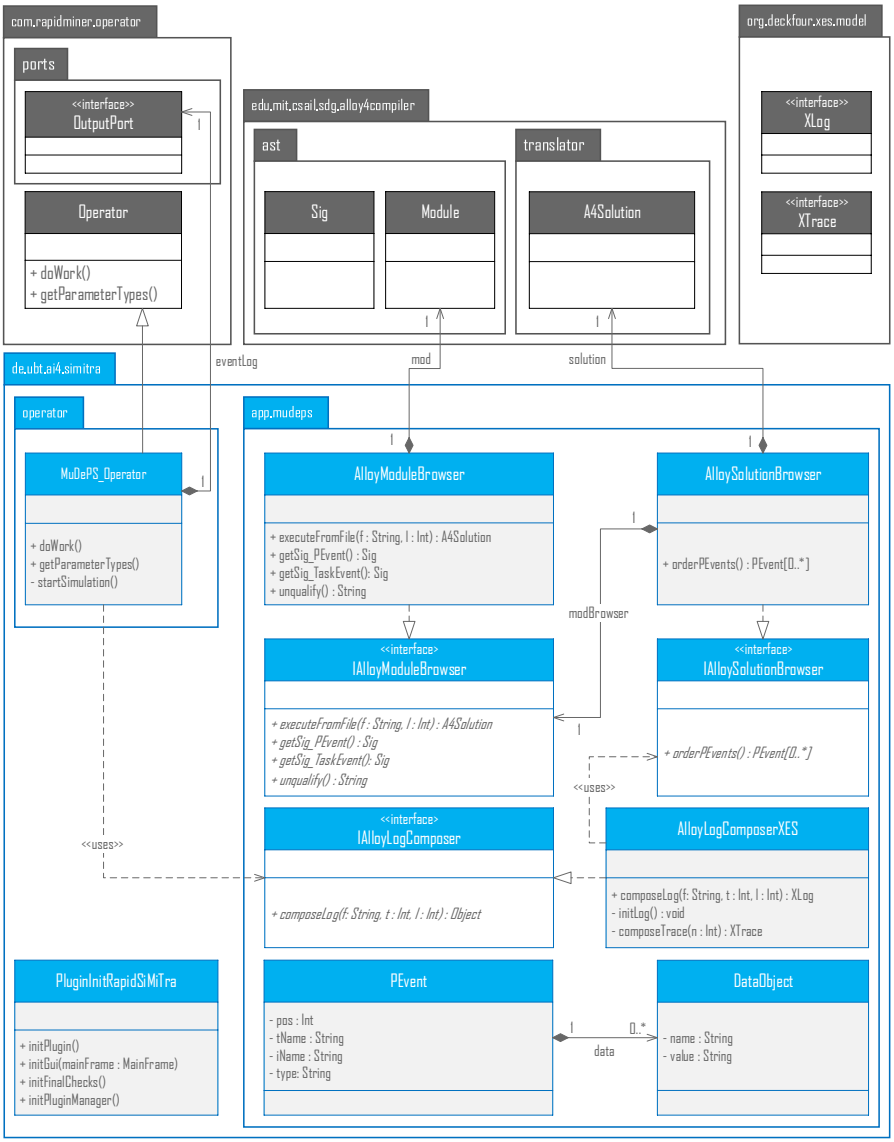



Abbildung 7.3: MuDePS: Architektur und RapidMiner-Integration

*Erweiterungsstruktur und Operator-Kapselung*

Die Klasse *PluginInitRapidSiMiTra* dient der RapidMiner-Plattform nach dem *Convention-over-Configuration-Prinzip* als Muster zur Erkennung von gültigen Erweiterungen. Der Aufbau der Klasse ist obligatorisch, kann aber mittels eines zur Verfügung stehenden Schablonen-Projekts[4] automatisch erzeugt werden. Als zweiter Anknüpfungspunkt zur RapidMiner-Plattform dient die Klasse *MuDePS_Operator* der Bereitstellung des Spurgenerators als Operator. Die Methode *doWork()* beinhaltet dabei die Logik für das Laufzeitverhalten des Operators – was hier vor allem den Start der Simulationsdurchführung beinhaltet. Dazu werden vorab die jeweiligen Aktualparameter abgerufen, wozu beispielsweise Spurlänge und -anzahl zählen. Das Ergebnis des Aufrufs dieser Methode wird dann entsprechend an die

4 https://github.com/rapidminer/rapidminer-extension-template, zuletzt aufgerufen: 08.05.2017

Ausgabe-Pins (*OutputPort*) geliefert, was im Falle des MuDePS-Operators ein Ereignisprotokoll ist.

Der MuDePS-Operator nutzt die zur Verfügung stehende Implementierung der Schnittstelle *IAlloyLogComposer*. Die derzeitige Umsetzung, namentlich *AlloyLogComposerXES*, liefert das Simulationsresultat als XES-Objekt (*XLog*) zurück, welches sich aus einzelnen Spuren (*XTrace*) zusammensetzt. Sie macht dabei Gebrauch von der Umsetzung einer weiteren Schnittstelle namens *IAlloySolutionBrowser*. Diese übernimmt die Aufgabe, ein Simulationsresultat, welches in einer zu Alloy konformen Repräsentation vorliegt, in die Zwischenstrukturen *PEvent* und *DataObject* zu überführen. Um diese Herausforderung bewältigen zu können, wird jedoch Wissen über das zugrundeliegende Alloy-Modell benötigt (*Module*), welches das DPIL-Prozessmodell repräsentiert. Grund dafür ist, dass die Abfragesprache der Alloy-Resultate wiederum Alloy ist, sodass bekannt sein muss, welche Signaturen (*Sig*), Prädikate und Funktionen zur Verfügung stehen. Dieses Wissen wird von der Umsetzung der Schnittstelle *IAlloyModuleBrowser* geliefert. Zudem ist diese Implementierung auch für den Start des Alloy-Analysewerkzeugs verantwortlich, welches wiederum die gesuchten Resultate der Spurgenerierung (*A4Solution*) zurückliefert. Der Start des Analysewerkzeugs erfolgt, wie auch bei der Verwendung von Alloy als unabhängiges System, über Kommandos, welche den Lösungsraum durch die Anzahl möglicher Atome pro Signatur begrenzen.

*Resultat als XES-Objekt*

*Einbettung des Alloy-Analysewerkzeugs*

Die in diesem Abschnitt vorgestellte Architektur des Spurgenerators für multiperspektivische, deklarative Prozessmodelle verbindet die Plattformfunktionalität der RapidMiner-Lösung mit dem Logikrahmenwerk Alloy. Dabei werden im Wesentlichen die Rückruffunktionen der Operator-Schnittstelle auf Kommandos des Alloy-Rahmenwerks abgebildet. Als Datengrundlage dient dabei im Wesentlichen die Alloy-Repräsentation des betreffenden DPIL-Modells. In Abschnitt 4.2 werden jedoch verschiedenste Parameter und Verwendungsmodi des Spurgenerators konzeptionell diskutiert. Da sich Einfluss und Einstellung dieser Parameter durch die Beschreibung ihrer Verwendung besser erläutern lassen als mittels eines Architekturbildes, werden die fraglichen Konfigurationsmöglichkeiten von MuDePS im Folgeabschnitt erklärt.

### 7.3.2 *Bedienung*

Um den Spurgenerator für DPIL-Modelle zu verwenden, sind im Wesentlichen zwei Schritte notwendig: *(i)* Das DPIL-Modell muss in eine Alloy-konforme Repräsentation konvertiert werden und *(ii)* der Spurgenerator muss konfiguriert und gestartet werden. Ziel des aktuellen Abschnittes ist es, diese beiden Schritte zu erläutern.

Die Transformation eines DPIL-Modells in eine gegenüber Alloy-konforme Repräsentation nach Vorlage von Abschnitt 4.2 ist als Modell-zu-Text-Transformation und mittels der Acceleo-Implementierung des Mof2Text-Standards implementiert. Da es sich bei dieser Technologie um ein auf Eclipse basierendes Rahmenwerk handelt, ist das resultierende Werkzeug als Eclipse-Erweiterung umgesetzt worden. Mittels den eingangs erwähnten Acceleo-Schablonen werden so musterbasiert DPIL-Modellkonstrukte in Alloy-Spezifikationen transformiert. Diese Transforma-

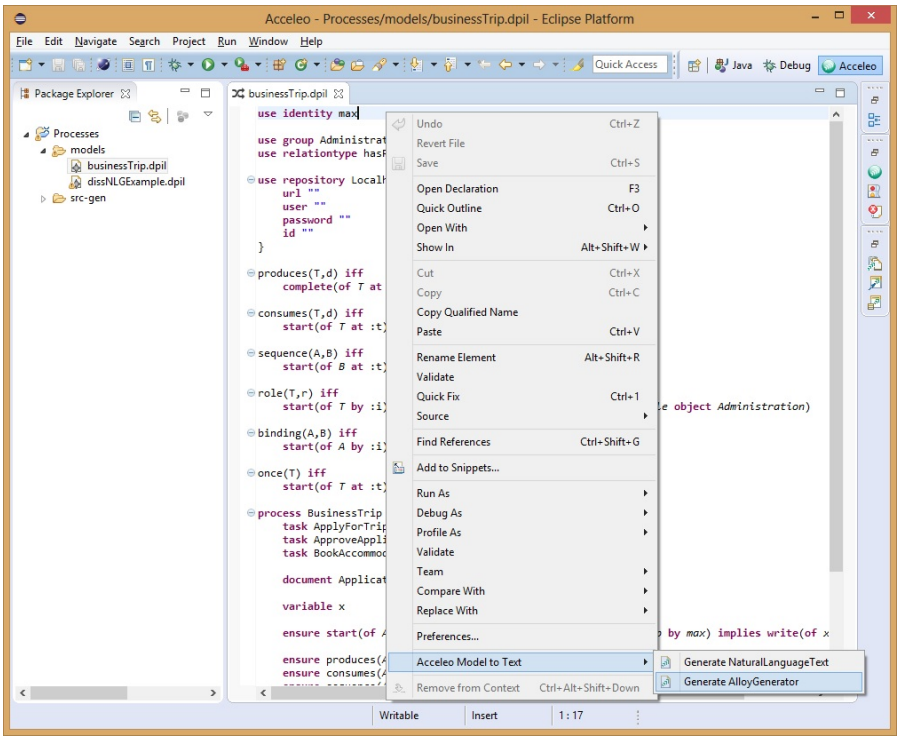



Abbildung 7.4: MuDePS: Transformation eines DPIL-Modells in Alloy

*Start über Kontextmenü*

tion kann im DPIL-Editor durch den Aufruf des Kontextmenüs und die Wahl des Eintrags *Generate AlloyGenerator* initiiert werden (Abbildung 7.4).

Das Ergebnis dieses ersten Schrittes ist eine Alloy-Repräsentation des Eingabe-DPIL-Modells, welche als *.als*-Datei im konfigurierbaren Zielordner abgelegt wird.

*Parametrierung über Operator-Detailansicht*

Im zweiten Schritt wird mittels des beschriebenen RapidMiner-Operators die eigentliche Spurgenerierung gestartet. Im Konzeptteil der Arbeit werden allerdings zahlreiche Parameter identifiziert, die den Rahmen der Spurgenerierung festlegen. In der Detailansicht des Operators können diese konfiguriert werden, was in Abbildung 7.5 dargestellt ist. Die Bedeutung der einzelnen Parameter kann anhand der Markierungen (*P1, P2, ...*) in Abschnitt 4.2 nachgeschlagen werden.

*Unterteilung in Experten- und Nicht-Expertenparameter*

Mit steigender Anzahl der Möglichkeiten zur Parametrierung des Spurgenerators kann die Verwendung desselben schwieriger werden. Aus diesem Grund sind die Parameter – konform mit der dafür vorgesehenen RapidMiner-Schnittstelle – in Experten- und Nicht-Expertenparameter klassifiziert. Die Parameter *P1* bis *P5* gehören dabei zu den Nicht-Expertenparametern. *P1* bezeichnet dabei den Parameter, welcher mit dem Pfad zu der in Schritt eins erzeugten Alloy-Repräsentation des DPIL-Modells assoziiert wird. Alle übrigen Parameter sind einerseits optional und andererseits nur im Expertenmodus sichtbar, welcher in Abbildung 7.5 aktiviert ist. Für die Verwendung des Operators im Translationskontext ist dieser Modus dagegen deaktiviert, was später in Abbildung 7.12 dargestellt ist. Der Start des Spurgenerators erfolgt über die Standard-Schaltfläche der RapidMiner-Plattform, welche den modellierten RapidMiner-Prozess ausführt. Der MuDePS-Operator erzeugt daraufhin unter Berücksichtigung der angegebenen Parametrierung ein Ereignisprotokoll und übermittelt dies an den Operator zum Export eines XES-

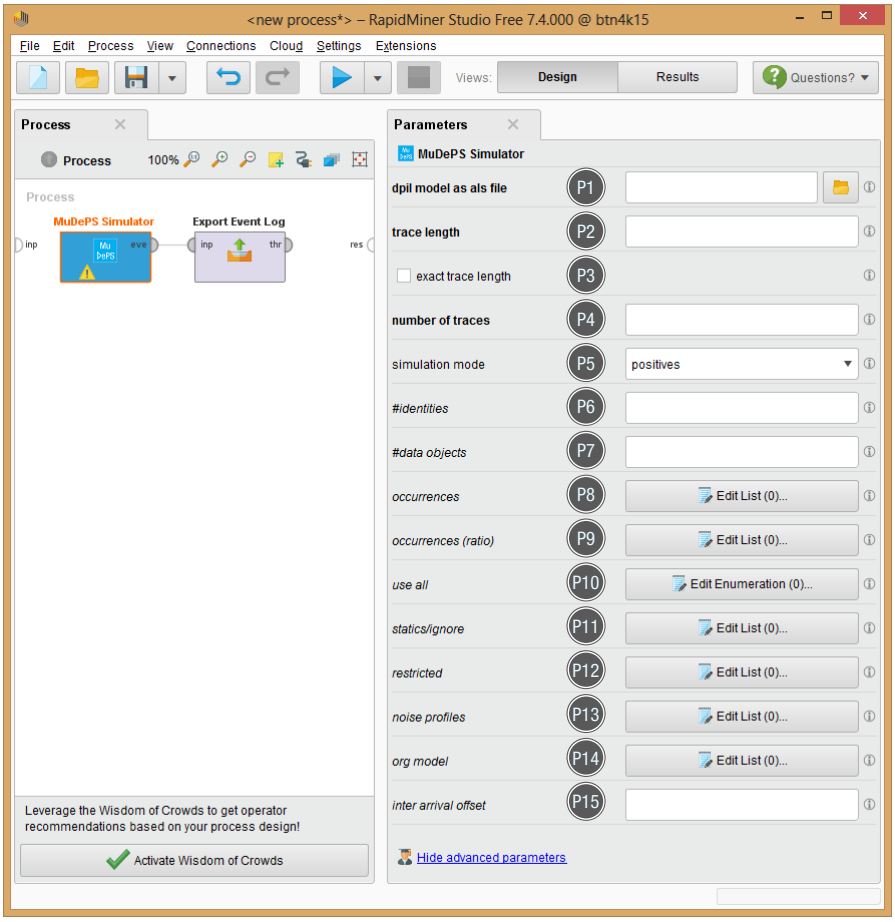



Abbildung 7.5: MuDePS: RapidMiner-Operator und Parameter

Ereignisprotokolls. Dieser zweite Operator kann durch jeden beliebigen anderen Operator ersetzt werden, welcher Ereignisprotokolle als RapidProM-interne Datenstruktur *XLogProM* als Eingabe akzeptiert.

Wie die Übersicht über verwandte Ansätze in Abschnitt 4.1 zeigt, existiert auch Bedarf für nicht-deterministische Parametrierungen eines Spurgenerators. Ein Beispiel dafür ist die Ankunftsrate von Prozessinstanzen. In Abschnitt 4.2 wird deshalb die Entwicklung eines Konzepts beschrieben, um im deterministischen, Alloy-basierten Simulationsansatz Zufallsvariablen berücksichtigen zu können. Dieses Konzept beinhaltet die Trennung der eigentlichen Spurgenerierung von der Erzeugung einer Belegung für alle Zufallsvariablen. In der RapidMiner-Plattform wird diese Trennung von Konfigurator und Alloy-basierter Spurgenerierung mittels des Operators *Generate Macro* umgesetzt. Der Operator entspricht dabei dem externen Konfigurator und eine beispielhafte Konfiguration ist in Abbildung 7.6 dargestellt. Der als *RandomActivityRatios* bezeichnete Konfigurator definiert zwei Makros, deren Auflösung als Funktionsausdruck angegeben wird. Das erste Makro des Beispiels beschreibt einen Zufallswert auf Basis einer Gleichverteilung (*rand()*). Die übrigen Bestandteile beschränken den Wertebereich der Zufallsvariablen auf natürliche Zahlen zwischen 1 und 4. Das Makro symbolisiert das Häufigkeitsverhältnis zwischen den beiden Aktivitäten *A* und *B*, hat aber noch

*Generierung von Belegungen für Zufallszahlen*

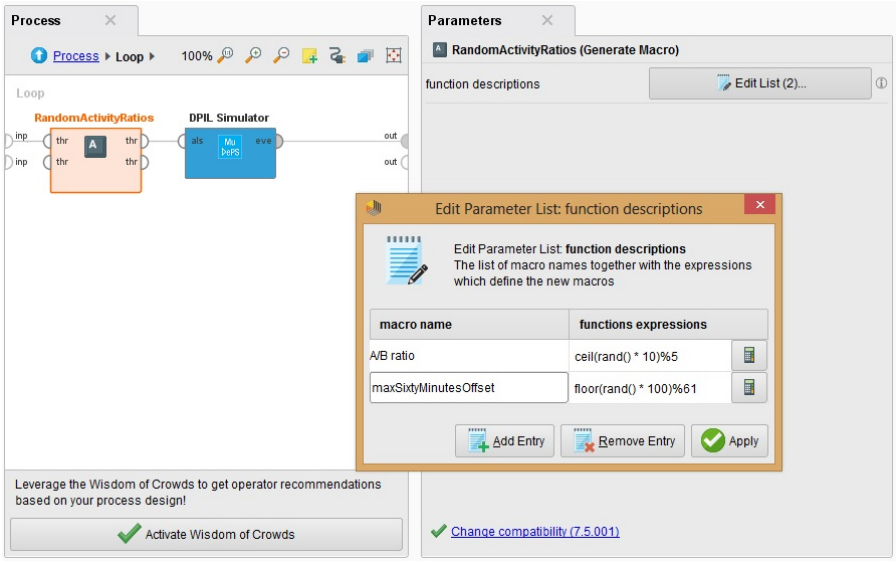



Abbildung 7.6: MuDePS: Generierung von Zufallswerten

keinerlei Auswirkungen. Erst in seiner Verwendung (Abbildung 7.7) wird der generierte Zufallswert während der Spurgenerierung eingesetzt.

*Verwendung von Zufallswerten*

Um einen Zufallswert zu verwenden, muss dieser über den Namen des zugehörigen Makros an der richtigen Stelle referenziert werden. In Abbildung 7.7 wird die Makroreferenz (*%{A/B ratio}*) zur Konfiguration des Häufigkeitsverhältnisses des Auftretens der beiden Aktivitäten verwendet. Analog wird in Abbildung 7.6 ein Makro für die Erzeugung einer zufälligen Verzögerung der Ankunftsrate neuer Prozessinstanzen definiert. Dieses wird ebenfalls in Abbildung 7.7 referenziert (*%{maxSixtyMinutesOffset}*), um die Ankunftsrate um maximal sechzig Minuten zu verzögern.

Die genannten Beispiele zeigen lediglich rudimentär die Konfiguration des Spurgenerators unter Verwendung von Zufallswerten. In der RapidMiner-Plattform ist zum aktuellen Zeitpunkt die Erzeugung von Zufallswerten lediglich auf Basis einer Gleichverteilung möglich. Im Erweiterungsteil wird beschrieben, wie zusätzliche Wahrscheinlichkeitsfunktionen nach Vorlage von Tabelle 3.3 zur Verfügung gestellt werden können.

### 7.3.3 *Erweiterung*

MuDePS basiert einerseits auf einer Modell-zu-Text-Transformation, welche DPIL-Modelle in die Sprache Alloy übersetzt und andererseits auf einer Übersetzung der einzustellenden Parameter auf ergänzende Alloy-Konstrukte. Ersteres ist mittels Acceleo realisiert und letzteres mit Hilfe der Programmierschnittstelle der Alloy-Bibliothek. Beide Aspekte sind von dem verwendeten Alloy-basierten Metamodell für Ereignisketten abhängig. Das Exportformat ist derzeit auf XES festgelegt. Im aktuellen Abschnitt werden die vorgesehenen Erweiterungsmöglichkeiten aller genannten Aspekte des Spurgenerators überblicksartig erläutert.

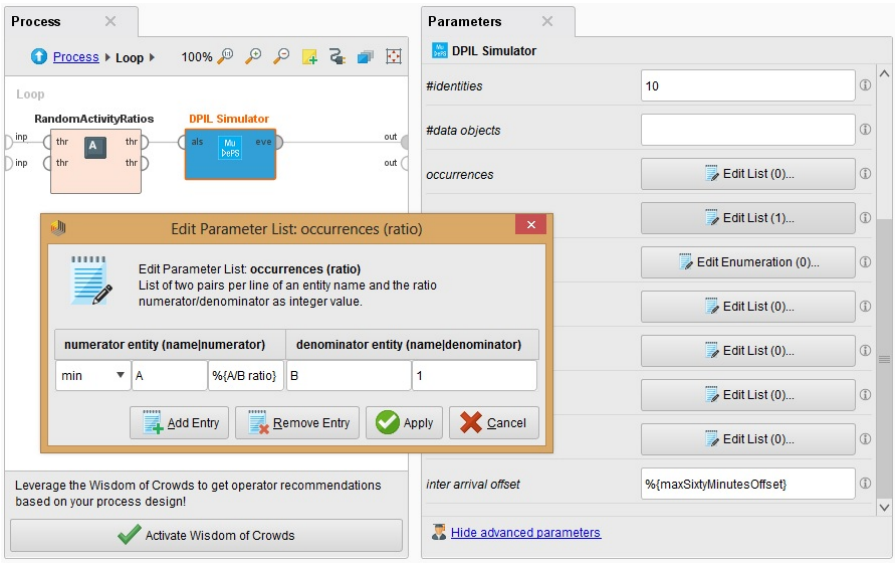



Abbildung 7.7: MuDePS: Verwendung von Zufallswerten

*Erweiterung: Zusätzliche DPIL-Regelschablonen*

Wie in Abschnitt 4.2 dargelegt wird, basiert die Transformation von DPIL-Regeln in eine Alloy-konforme Repräsentation auf struktureller Mustersuche. Folglich ist es für die erfolgreiche Transformation eines DPIL-Modells notwendig, dass für jede darin enthaltene Regel eine passende Musterregel und die zugehörige Alloy-Repräsentation vorhanden ist. Derzeit ist es möglich, die in Tabelle 4.3 beschriebenen Regeltypen für die Spurgenerierung zu berücksichtigen. Soll die Menge dieser Regeltypen erweitert werden, dann ist dafür die Erweiterung der Acceleo-basierten Modell-zu-Text-Transformation notwendig. Das Vorgehen bei derartigen Erweiterungen kann aus Codeausschnitt 7.1 und der dazugehörigen Beschreibung abgeleitet werden. Die neue Abbildungsvorschrift wird in einer neuen Schablone gekapselt, deren Name *rule* ist und welche eine *ProcessRule* als Formalparameter erhält. Das zu suchende Muster ist als Guard-Ausdruck zu beschreiben. Der zu generierende Alloy-Code bildet den Rumpf der Schablone. Damit ist die Unterstützung der neuen DPIL-Regel seitens des MuDePS-Spurgenerators sichergestellt.

*Erweiterung: Komplexerer Lebenszyklus von Aktivitäten*

Mit den beiden Schnittstellen *IAlloyModuleBrowser* und *IAlloySolutionBrowser* ist eine Möglichkeit geschaffen worden, um Erweiterungen an dem in Alloy formulierten Metamodell für Ereignisketten vorzunehmen. Möchte man beispielsweise ein komplexeres Modell des Lebenszyklus von Aktivitäten verwenden, dann muss die Methode *orderPEvents* der Schnittstelle *IAlloySolutionBrowser* neu implementiert werden. Das Überschreiben der Methode *unqualify* von *IAlloyModuleBrowser* ist dann notwendig, wenn die Namen des Metamodell-Moduls für Ereignisketten geändert wird. Auch die Methode *executeFromFile* muss überarbeitet werden, da mit einem komplexeren Lebenszyklus in den meisten Fällen auch die Zahl der Ereignisse pro Aktivitätsdurchführung zunimmt. Dafür muss jedoch die Anzahl der zur Verfügung stehenden Positionsindizes in der Ereigniskette erhöht werden. Bei einer Änderung des Metamodells für Ereignisketten ist zusätzlich zu berücksichtigen, dass die Abbildungsschablonen für DPIL-Regeln ungültig werden können, was dann Anpassungen am Acceleo-basierten Alloy-Generator erfordert.

Die Schnittstelle *IAlloyLogComposer* sowie die beiden Zwischenstrukturen *PEvent* und *DataObject* werden benötigt, um das Serialisieren und Persistieren eines Ereignisprotokolls vom Parsen der Alloy-Resultate zu trennen. Somit ist es möglich, ein anderes Protokollformat – beispielsweise MXML – zu verwenden, ohne die Verarbeitung der Alloy-Resultate verändern zu müssen. Dazu muss lediglich eine alternative Umsetzung der erwähnten Schnittstelle entwickelt werden.

*Erweiterung: Alternatives Protokollformat*

Abseits des Austauschs von Komponenten des Alloy-basierten Spurgenerators, ist auch die Erweiterung von Standard-Operatoren der RapidMiner-Plattform möglich. Dies ist notwendig, um Wertebelegungen für an der Spurgenerierung beteiligten Zufallsvariablen mittels komplexerer Wahrscheinlichkeitsfunktionen erzeugen zu können. Zum aktuellen Zeitpunkt wird diesbezüglich nur die Gleichverteilung unterstützt. Diese wird über einen entsprechenden Funktionsausdruck bei der Definition eines Makros verwendet. Für die Definition von Makros auf Basis alternativer Wahrscheinlichkeitsfunktionen, muss für jede diese Funktionen ein entsprechender Parser implementiert und registriert werden. Die Registrierung erfolgt über den Aufruf der Methode *register(...)* der Klasse *ExpressionRegistry*. Alternativ können einige Wahrscheinlichkeitsfunktionen auch mit dem verfügbaren Funktionsumfang nachgebildet werden. Beispielsweise können Verfahren wie die *Box-Muller-Methode* [167] verwendet werden, um standardnormalverteilte Zufallszahlen auf Basis zweier Zufallsvariablen zu erzeugen, welche auf einer Gleichverteilung basieren.

*Erweiterung: Komplexere Wahrscheinlichkeitsfunktionen*

Das vorrangige Ziel von Abschnitt 7.3 ist es, mittels der Beschreibung einer prototypischen Implementierung die Realisierbarkeit des in dieser Arbeit vorgestellten Ansatzes zur Spurgenerierung nachzuweisen. Das entsprechende Konzept (Abschnitt 4.2) beinhaltet unter anderem eine Liste von Eigenschaften und Funktionen, welche als Anforderungskatalog für diese prototypische Entwicklung dient. In der beschriebenen Implementierung werden alle Anforderungen umgesetzt oder explizit als Erweiterungspunkt ausgewiesen. Eine detailliertere Aufstellung findet sich im Evaluationsteil der vorliegenden Arbeit (Abschnitt 8.1.1).

## 7.4 SIMITRA ALS RAPIDMINER-ERWEITERUNGEN

Der in der vorliegende Arbeit vorgestellte Translationsansatz für Prozessmodelle (Abschnitt 5.2 und Abbildung 7.8) basiert auf der Kombination unterschiedlicher Techniken zur Spurgenerierung mit verschiedenen Process-Mining-Techniken. Da bisher keine vergleichbaren Ansätze existieren, ist es von zentraler Bedeutung, das Potential dieses Vorgehens abschätzen zu können. Ziel dieses Abschnitts ist es, eine generische Implementierung in Form einer RapidMiner-Erweiterung vorzustellen.

*RapidMiner als Implementierungsgrundlage*

Unterschiedliche Spurgeneratoren und unterschiedliche Process-Mining-Werkzeuge liefern potentiell unterschiedliche Ergebnisse, sodass aus einer willkürlichen Kombination *eines* Spurgenerators mit *einem* Process-Mining-Werkzeug kaum Rückschlüsse auf das generelle Potential dieses Prinzips gezogen werden können. Folglich ist es wichtig, flexibel verschiedene dieser Paarungen testen zu können. Jede dieser Technologien ist zudem parametrierbar, sodass eine Wiederholbarkeit der Versuche mit gleichen Eingabedaten aber unterschiedlichen Parametrierungen

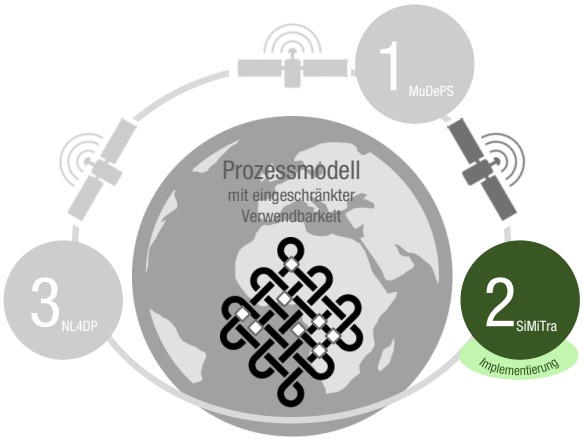



Abbildung 7.8: SiMiTra (Implementierung)

obligatorisch ist. Vor diesem Hintergrund eignet sich die RapidMiner-Plattform als Implementierungsgrundlage. Die Gründe sind nachfolgend aufgelistet:

- Durch das generische Operator-Prinzip der Plattform können existierende Spurgeneratoren und Process-Mining-Techniken leicht integriert werden. Die Möglichkeit der Verknüpfung der gewünschten Technologiepaare ist dadurch bereits geschaffen, da die Verknüpfung von Operatoren eine der Grundfunktionen des RapidMiner-Editors ist.

  *Flexible Verknüpfung der Komponenten*

- Der Erweiterungsmechanismus der Plattform ermöglicht die Erstellung von RapidMiner-Prozessschablonen. Diese können beispielsweise vorkonfigurierte Operator-Ketten beinhalten, worauf die im anschließenden Abschnitt beschriebene Architektur aufbaut.

  *Prozessschablonen*

- Bereits vorhandene Superoperatoren – vor allem der bereits erwähnte Schleifenoperator – ermöglichen es, Experimente mit verschiedenen Parametrierungen zu wiederholen.

  *Wiederholbarkeit der Experimente*

- Mit der RapidProM-Erweiterung stehen auch Operatoren für die Qualitätsprüfung der Translationsresultate zur Verfügung. Diese können nach Belieben in den Translationsprozess integriert werden.

  *Wiederverwendung von Operatoren zur Qualitätsprüfung*

- Es stehen außerdem Operatoren zur Verfügung, um Informationen – beispielsweise Ereignisprotokolle realer Prozessausführungen – aus externen Quellen zu beziehen. Zu diesen Quellen gehören beispielsweise Datenbankensysteme, XML-Dateien und auch Excel-Tabellen.

  *Zugriff auf externe Quellen*

- RapidMiner stellt zusätzlich Meta-Informationen über die Durchführung der Translation zur Verfügung. Dazu zählt beispielsweise die Durchlaufzeit, was eine Analyse der Leistungsfähigkeit des Translationsansatzes begünstigt.

  *Messung der Durchlaufzeiten*

In den nachfolgenden Abschnitten wird einerseits die Architektur des im Rahmen der vorliegenden Arbeit vorgestellten Translationssystems beschrieben und andererseits die Verwendung des resultierenden Werkzeugs. Durch den flexiblen Erweiterungsmechanismus der zugrundeliegenden RapidMiner-Plattform ist die

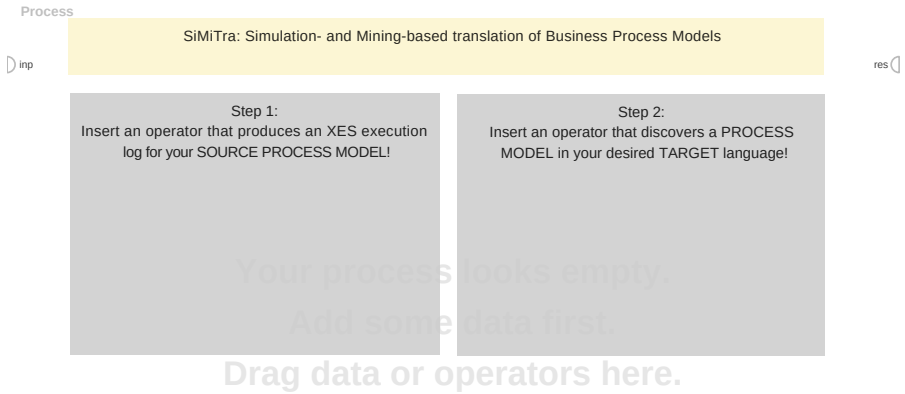



Abbildung 7.9: SiMiTra: Grundgerüst der Prozessschablone

Erweiterbarkeit des Systems ohne weiteres möglich, was im Anschluss an die Bedienungshinweise überblicksweise diskutiert wird.

### 7.4.1 *Architektur*

*Implementierung durch Wiederverwendung existierender Operatoren*

Die Architektur für die Umsetzung des Translationskonzepts aus Abschnitt 5.2 basiert zwar ebenfalls auf der RapidMiner-Plattform, ist jedoch im Gegensatz zum Spurgenerator MuDePS keine Erweiterung. Stattdessen werden die von der Plattform und einer Process-Mining-Erweiterung angebotenen Operatoren genutzt, um den Translationsprozess zu implementieren. Der aktuelle Abschnitt beschreibt die sich daraus ergebende Architektur.

*Implementierungsvorlage als RapidMiner-Prozess*

Jedes dem in Abschnitt 5.2 beschriebenen Konzept entsprechende Translationssystem besteht aus einem Spurgenerator für das Quellmodell und einem Process-Mining-Werkzeug für die Zielsprache. Übertragen auf die RapidMiner-Plattform bedeutet das, dass jedes Translationssystem einem RapidMiner-Prozessdiagramm entspricht, welches je einen Operator für Spurgenerierung und Process Mining enthält. Hierfür kann in der Plattform eine Prozessschablone bereitgestellt werden. Ohne Festlegung des konkreten Sprachpaares kann lediglich eine leere Prozessschablone angeboten werden, die um Hinweise bezüglich ihrer Verwendung angereichert ist. Eine solche Schablone ist in Abbildung 7.9 dargestellt.

*Beispielimplementierungen*

Eine derartige leere Prozessschablone kann lediglich als Anleitung dienen, ist jedoch keine wirkliche Implementierung des in Abschnitt 5.2 vorgestellten Konzepts. Aus diesem Grund wird nicht nur die Blankoschablone des Translationssystems angeboten, sondern auch Implementierungen des Translationskonzepts für konkrete Sprachpaare. In Abbildung 7.10 ist das am Beispiel einer Translation von DPIL in die Petri-Netz-Notation dargestellt.

Die Bereitstellung der Schablonen für konkrete Sprachpaare bietet mehrere Vorteile. Einerseits reduziert sich der Aufwand des Nutzers darauf, die korrekte Schablone und das zu übersetzende Prozessmodell auszuwählen. Andererseits können die beiden zu verwendenden Operatoren vorkonfiguriert werden, sodass sie bestmöglich harmonieren.

*RapidMiner-Prozessschablonen*

Prozessschablonen sind als ZIP-Archive in die Plattform zu integrieren. Erneut wird hier vom Convention-over-Configuration-Prinzip ausgegangen. Das bedeutet,

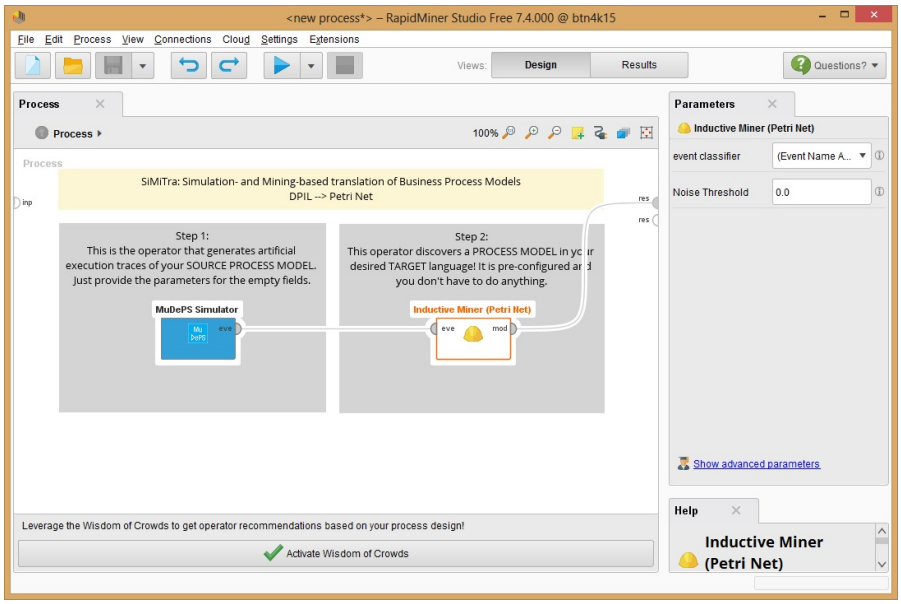



Abbildung 7.10: SiMiTra: Prozessschablone für die Translation von DPIL-Modellen in Petri-Netze

dass die Inhalte des Archivs und auch dessen Struktur reglementiert sind und so eine zusätzliche Konfiguration überflüssig machen. Die Prozessschablone – als einer der Bestandteile des Archivs – kann wie gewohnt mittels RapidMiner erzeugt und durch eine entsprechende Exportfunktion extrahiert werden. Neben einem Symbol zur Darstellung enthält das Archiv auch eine Konfigurationsdatei für die Festlegung der Beschreibung und des Namens der Schablone. Schließlich müssen die so erzeugten Vorlagen für die RapidMiner-Plattform bekannt gemacht werden. Das kann beispielsweise über eine Erweiterung erfolgen, wie es auch im Rahmen der vorliegenden Arbeit umgesetzt wird. Hierbei müssen lediglich Pfad und Name jeder Schablone in der *initPlugin()*-Methode der Klasse *PluginInitRapidSiMiTra* registriert werden. Für detailliertere Informationen zur Erstellung von RapidMiner-Prozessschablonen verweist der Autor auf die offizielle Hilfeseite[5].

### 7.4.2 *Bedienung*

Da die Bedienung des Translationssystems durch die Nutzeroberfläche der RapidMiner-Plattform bedingt wird, werden im aktuellen Abschnitt nur die nötigen Arbeitsschritte erläutert, um die jeweilige Prozessschablone verwenden zu können.

Abbildung 7.11 zeigt eine Ansicht, die sich dem Nutzer beim Start von RapidMiner präsentiert. Auf der rechten Seite befindet sich eine Auflistung der derzeit verfügbaren Prozessschablonen – darunter auch die im Rahmen dieser Arbeit entwickelten. Folglich ist der *erste Schritt* die Auswahl der passenden Schablone für das aktuelle Translationsproblem. Steht keine solche Schablone zur Verfügung,

*Schritt 1: Wahl der Schablone*

5 https://docs.rapidminer.com/developers/how-to/custom-templates.html, zuletzt aufgerufen: 11.05.2017

dann kann noch auf die Blankoschablone für derartige Translationssysteme zurückgegriffen werden. Die nachfolgenden Erläuterungen basieren auf der Schablone für die Translation von DPIL-Modellen in eine Petri-Netz-Repräsentation.

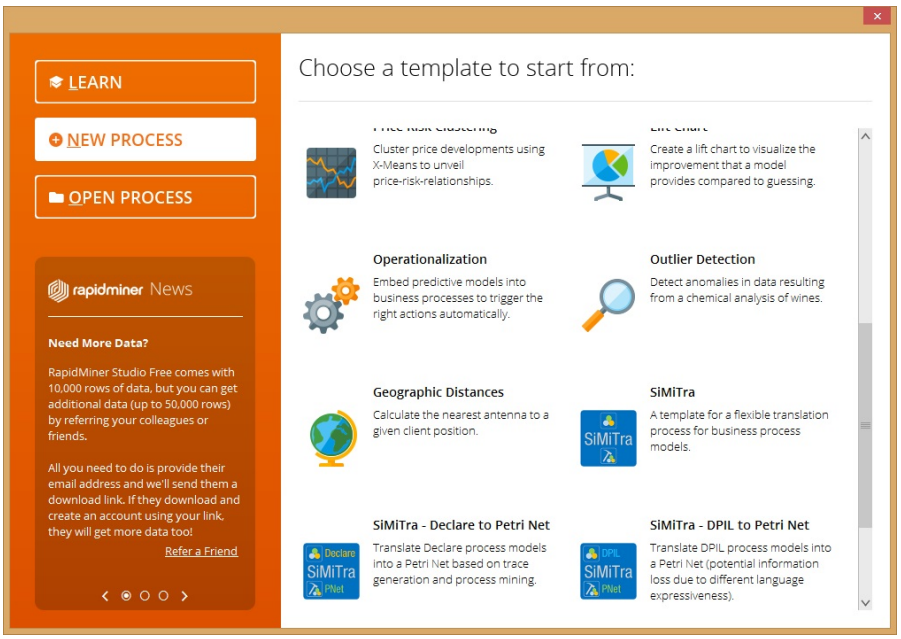



Abbildung 7.11: SiMiTra: Prozessschablonen

*Schritt 2: Konfiguration des Spurgenerators*

Der *zweite Schritt* ist die Konfiguration des Operators für die Spurgenerierung. Für das aktuelle Übersetzungsbeispiel handelt es sich konkret um den bereits aus Abschnitt 7.3 bekannten MuDePS-Operator. Für den Einsatz im Translationskontext müssen lediglich die Basisparameter konfiguriert werden, welche in Abbildung 7.12 dargestellt sind. Das sind einerseits die Alloy-Repräsentation des DPIL-Prozessmodells, die Anzahl der zu generierenden Spuren sowie die maximale Länge jeder Prozessausführungsspur.

*Schritt 3: Konfiguration des Process-Mining-Werkzeugs*

Im *dritten Schritt* ist das Process-Mining-Werkzeug zu konfigurieren, welches aus den künstlich erzeugten Prozessausführungsspuren ein Prozessmodell in der gewünschten Zielsprache erzeugt. Für das aktuelle Beispiel wird willkürlich die RapidMiner-Implementierung des *Inductive Miner*[106] ausgewählt, deren Parameter in Abbildung 7.10 dargestellt sind. Der Operator ist in diesem Fall vollständig vorkonfiguriert, sodass der Nutzer keine Einstellungen vornehmen muss. Im Gegensatz zu Anwendungen auf reale Ereignisprotokolle bieten künstliche den Vorteil der Rauschfreiheit. Aus diesem Grund wird die Rauschschwelle des Process-Mining-Werkzeugs auf Null gesetzt. Somit ist sichergestellt, dass jedes protokollierte Verhalten des künstlichen Ereignisprotokolls im Modell berücksichtigt wird.

*Schritt 4: Ausführung der Translation*

Der *vierte* und letzte *Schritt* ist die Ausführung der Translation. Dies geschieht mittels der allgemeinen Start-Schaltfläche der RapidMiner-Plattform, welche den kompletten Prozess ausführt. Das Ergebnis – für das aktuelle Beispiel ein Petri-Netz – wird abschließend in einer gesonderten Ansicht präsentiert. Mithilfe eines weiteren RapidProM-Operators ist es zudem möglich, das erzeugte Petri-Netz als *.pnml*-Datei zu exportieren und in mit diesem Format vertrauten Drittapplikationen weiterzuverwenden. Damit ist die Translation abgeschlossen.

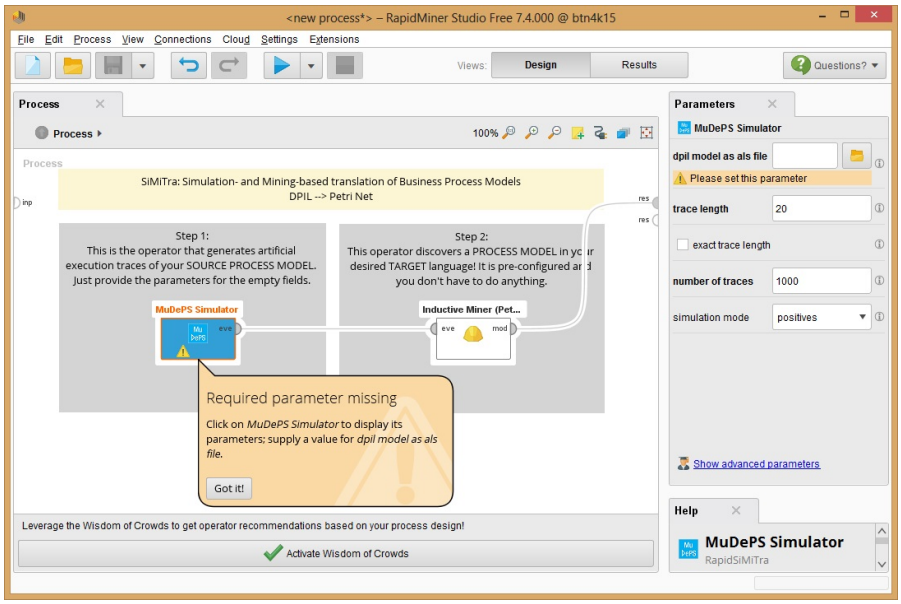



Abbildung 7.12: SiMiTra: Verwendung von MuDePS im Translationskontext

### 7.4.3 *Erweiterung*

*Zusätzliche Modellierungssprachen und Erweiterung des Translationsprozesses*

Erweiterungsmöglichkeiten der SiMiTra-Technologie lassen sich in zwei Klassen unterteilen: *(i)* Erweiterung um zusätzliche Prozessmodellierungssprachen und *(ii)* Ergänzung bestehender Translationsprozesse. Das Vorgehen bei derartigen Erweiterungen wird nachfolgend erläutert.

Ein Nachteil der Entscheidung, die RapidMiner-Plattform als Rückgrat der SiMiTra-Architektur zu verwenden ist, dass es so notwendig ist, dass jeder benötigte Spurgenerator und jede benötigte Process-Mining-Technik als RapidMiner-Operator vorliegt. Zwar existieren für viele der gängigen Prozessmodellierungssprachen Technologien für die Spurgenerierung und das Process Mining. Diese liegen jedoch in den meisten Fällen nicht als RapidMiner-Erweiterung vor. Folglich besteht die Hauptaufgabe bei der Erweiterung um zusätzliche Prozessmodellierungssprachen darin, diese RapidMiner-externen Technologien in Operatoren zu Kapseln. Das Vorgehen dabei ist bereits in Abschnitt 7.3 am Beispiel des Spurgenerator-Operators für DPIL-Modelle erläutert. Im Wesentlichen muss somit die grundlegende Struktur von RapidMiner-Erweiterungen eingehalten und es müssen die entsprechenden Basisklassen erweitert werden. Innerhalb der neuen Operator-Klasse gilt es dann, die zu konfigurierenden Parameter der gekapselten Technologie in der Nutzeroberfläche der Plattform zu visualisieren. Bei der Erweiterung von Operatoren um zusätzliche Parameter muss die Methode *getParameterTypes* der Operator-Basisklasse überschrieben werden, wie es in Codeausschnitt 7.2 dargestellt ist.

Der Codeausschnitt zeigt die Registrierung eines Parameters mit dem Namen „process model“, einer Beschreibung der Bedeutung des Parameters und einem Binärwert, welcher anzeigt, ob es sich um einen Expertenparameter handelt. Da es sich bei diesem Parameter um die Auswahl einer Datei aus einer externen Quelle handelt, ist er als *ParameterTypeFile* registriert und die Auswahl gültiger

```
@Override
public List<ParameterType> getParameterTypes() {
   List<ParameterType> pt = super.getParameterTypes();
   pt.add(new ParameterTypeFile("process model",
      "Choose process model as Alloy als file.", false,
         new String[] { "als" }));

   return pt;
}
```

Codeausschnitt 7.2: Erweiterung eines RapidMiner-Operators um zusätzliche Parameter

Dateierweiterungen ist auf „als" begrenzt. Möchte man einen weiteren Parameter registrieren, so muss dieser lediglich analog zum Beispiel in die Parameterliste eingetragen werden. So ist es möglich, die meist als externe Bibliotheken vorliegenden Spurgeneratoren und Process-Mining-Werkzeuge zu parametrieren und damit ihre Funktionalität in der RapidMiner-Plattform zur Verfügung zu stellen.

Die zweite Erweiterungsklasse umfasst alle Änderungen, bei dem die vordefinierten Prozessschablonen für eine Translation um zusätzliche Operatoren erweitert werden sollen. Da es möglich ist, individuell zusätzliche Operatoren zu definieren, kann die nachfolgende Beschreibung nur beispielhaft sein. Die RapidMiner-Plattform und die Operatoren der RapidProM-Erweiterung ermöglichen die Erweiterung der Prozessschablonen unter anderem um die folgende Funktionalität:

*Durchlaufzeitmessung*

- *Messung der Durchlaufzeiten:* Der *Log*-Operator der RapidMiner-Plattform erlaubt das Protokollieren von Prozess- und Operator-Durchlaufzeiten. Damit ist es möglich, bei Bedarf für konkrete Translationsprobleme den Zeitaufwand zu messen.

*Konformitätsprüfung*

- *Konformitätsprüfungen:* Die RapidProM-Erweiterung liefert einige Operatoren, welche dazu verwendet werden können, die Übereinstimmung zwischen einem Ereignisprotokoll und einem Prozessmodell zu bewerten. Davon wird im Evaluationsteil der vorliegenden Arbeit Gebrauch gemacht.

*Automatisierte Wiederholung*

- *Automatisierte Wiederholungen:* Der Schleifen-Superoperator der Plattform ermöglicht die automatisierte Wiederholung von RapidMiner-Prozessen mit unterschiedlichen Parametrierungen. Das eröffnet im aktuellen Kontext und in Kombination mit den Operatoren zur Konformitätsprüfung beispielsweise die Möglichkeit, die Qualität der Translationsergebnisse in Abhängigkeit von Spurlänge und -anzahl zu prüfen.

*Ergebnisselektion*

- *Selektion des besten Resultats:* Häufig existieren für ein gewähltes Sprachpaar mehrere Spurgeneratoren und/oder Process-Mining-Techniken. Ist das der Fall, so kann mittels des *Branch*-Operators entschieden werden, welcher Operator jeweils im Translationsprozess aktiv werden soll. In Kombination mit den Operatoren der Konformitätsprüfung kann so die Übersetzung für jede verfügbare Kombination aus Spurgenerator und Process-Mining-Werkzeug einzeln durchgeführt und das Ergebnis mit der besten Protokoll-Modell-Konformität ausgewählt werden.

Generell ist der Erweiterungsmechanismus von SiMiTra Teil der grundlegenden Modellierungsfunktionalität der RapidMiner-Plattform. Folglich sind den Erweiterungsmöglichkeiten – bei Verfügbarkeit der entsprechenden Operatoren – kaum Grenzen gesetzt. Für Detailinformationen zum RapidMiner-Erweiterungsmechanismus sei auf die zugehörige Plattform-Dokumentation[6] verwiesen.

Das Hauptziel von Abschnitt 7.4 ist, die Realisierbarkeit des generischen Prinzips zur Translation von Prozessmodellen (Abschnitt 5.2) nachzuweisen. Dieses basiert im Wesentlichen auf der Schaffung einer Infrastruktur, um beliebige Spurgeneratoren mit ebenso beliebigen Process-Mining-Technologien kombinieren zu können. Durch die Adaption und Kapselung bestehender Verfahren in RapidMiner-Operatoren bietet die RapidMiner-Plattform diese Infrastruktur nativ. Zusätzlich können weitere Operatoren eingesetzt werden, um beispielsweise Ereignisprotokolle zu modifizieren oder Übersetzungsergebnisse beliebig zu exportieren. Gleichzeitig können beispielsweise Prozessmodelle aus verschiedenen Quellen importiert werden. Durch die grafische RapidMiner-Nutzerschnittstelle zur Organisation von Analyseprozessen sind zudem komfortable Interaktionsmöglichkeiten gegeben. Für eine zusätzliche Unterstützung des Nutzers können zudem auch Schablonen für Operator-Kombinationen angeboten werden. Die notwendige Kapselung jeder Komponente in einem Operator ist somit der einzige Implementierungsaufwand.

## 7.5 NL4DP ALS ECLIPSE-ERWEITERUNG

Im verbleibenden Teil dieses Kapitels wird die Umsetzung des in Abschnitt 6.2 beschriebenen NLG-Ansatzes erläutert (Abbildung 7.13). Wie der vorab vorgestellte Spurgenerator, basiert auch die Umsetzung dieses Ansatzes zu großen Teilen auf der Mof2Text-Standard-Implementierung Acceleo.

*Implementierung des NLG-Ansatzes auf Basis von Acceleo*

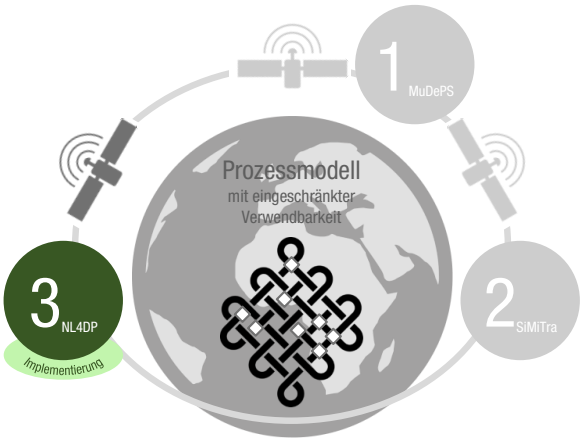



Abbildung 7.13: NL4DP (Implementierung)

Vorrangiges Ziel dieses Abschnitts ist es, die Integration des NLG-Systems in die existierende Werkzeugunterstützung für DPIL-Prozessmodelle zu erläutern. Das bedeutet, dass die Technik als Eclipse-Erweiterung umgesetzt wird, da auch das zugehörige DPIL-Modellierungswerkzeug so realisiert ist.

6 https://docs.rapidminer.com/developers/creating-your-own-extension/, zuletzt aufgerufen: 15.05.2017

### 7.5.1 *Architektur*

Gegenstand dieses Abschnitts ist es, die Umsetzung der in Abschnitt 6.2 beschriebenen Pipeline-Architektur zur Generierung natürlichsprachlicher Texte für DPIL-Modelle zu beschreiben. Aufgrund der Reduktion der Untersuchungen auf einen exemplarischen Durchstich wird auch in diesem Abschnitt lediglich exemplarisch auf die Umsetzung eingegangen.

Das Pipeline-Modell sieht im Wesentlichen sechs Schritte und drei Eingabeparameter vor. Die Umsetzung der Architektur erfolgt nach dem in Abbildung 7.14 dargestellten Schema.

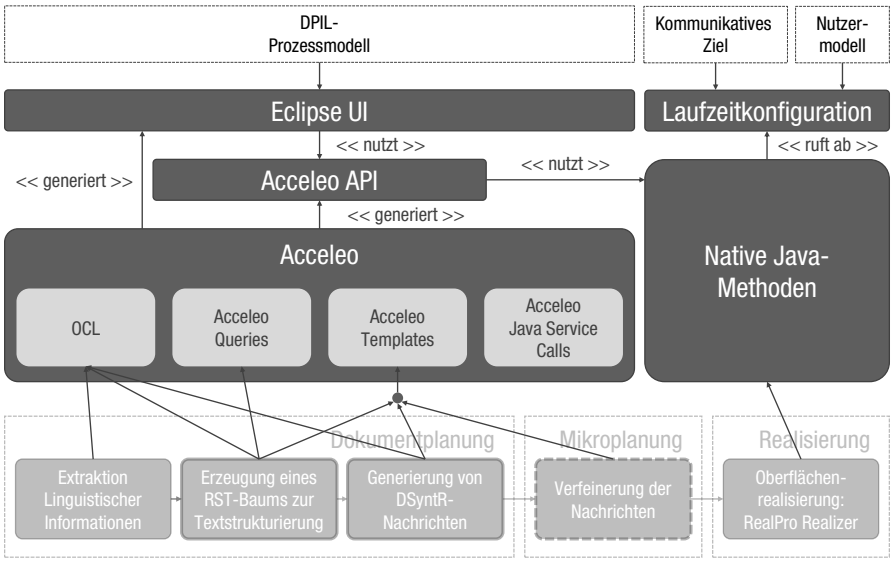



Abbildung 7.14: Operatoren im RapidMiner-Prozess

Wie aus der Abbildung dargestellt ist, basiert das NLG-System nahezu vollständig auf Acceleo. Dies ist eine Abweichung von der üblichen Vorgehensweise modellgetriebener Softwareentwicklung. Die Erzeugung eines RST-Baums und die Transformation von DPIL-Regeln in DSyntR-Baumstrukturen würde ein Implementierung basierend auf Modell-zu-Modell-Transformationen vorschlagen – und keine direkte Modell-zu-Text-Transformation, wie es mit Acceleo der Fall ist. Grund ist, dass sowohl RST-Bäume als auch DSyntR-Nachrichten eigentlich *Modelle* sind. Voraussetzung für die Definition einer Modell-zu-Modell-Transformation ist jedoch die Verfügbarkeit der Metamodelle für alle Modelle. Da dem Autor jedoch für RST-Bäume und DSyntR-Nachrichten keine solchen bekannt sind und die Definition derselben Grenzen und Fokus dieser Arbeit überschreiten würde, wird stattdessen ein rein auf Modell-zu-Text-Transformationen basierender Ansatz realisiert. In Konsequenz dazu werden einige Schritte der NLG-Pipeline vereinfacht.

*Fehlende Metamodelle*

Die *Extraktion Linguistischer Informationen* ist im aktuellen Stand der Implementierung einfach gehalten. Durch die in Abschnitt 6.2 getroffene Annahme, dass Aktivitätsbezeichner immer aus Nomen und Verb bestehen, ist vorerst keine Wortklassifikation durch Technologien wie WordNet oder den Stanford Tagger notwendig. Stattdessen kann der Bezeichner einfach durch Zeichenkettenauftrennung am Leerzeichen in die zwei gesuchten Wörter zerlegt werden.

*Extraktion linguistischer Informationen*

*Textstrukturierung*

Ziel des zweiten Schrittes ist die *Erzeugung eines RST-Baums zur Textstrukturierung*. Dafür ist einerseits die Gruppierung der Regeln gemäß des in Abschnitt 6.2 beschriebenen Verfahrens notwendig und andererseits müssen mögliche rhetorische Relationen zwischen den so gruppierten Regeln identifiziert werden. Im Konzept für ersteres ist zunächst eine Gruppierung der DPIL-Regeln basierend auf der verwendeten Regelschablone vorgesehen. Anschließend werden die so gebildeten Gruppen intern anhand der Anzahl gleicher Parameter geordnet. In Acceleo kann diese Gruppierung mittels der OCL-Funktion *select* gebildet werden, wobei der zugehörige Filterausdruck den Abgleich des DPIL-Makro-Namens beinhaltet. Das Ergebnis sind eine oder mehrere Mengen von DPIL-Regeln, welche jeweils dasselbe Makro referenzieren. Diese Mengen werden anschließend mittels eines zweiten OCL-Ausdrucks – *sortedBy* – absteigend nach der Anzahl gleicher Parameter sortiert.

*Verschmelzung mit nachfolgenden Pipeline-Schritten*

Damit ist zwar eine Gruppierung und gruppeninterne Sortierung vorgenommen worden, der zu erzeugende RST-Baum ist jedoch weiterhin ausstehend. Da durch die fehlenden Metamodelle von RST-Bäumen und DSyntR-Nachrichten auf eine Modell-zu-Text-Transformation anstelle einer Modell-zu-Modell-Transformation zurückgegriffen werden musste, wird die Erstellung des RST-Baums mit den zwei darauffolgenden Pipeline-Schritten verschmolzen. Der Grund dafür ist, dass sowohl der RST-Baum als auch die noch zu erzeugenden DSyntR-Nachrichten nicht als Modell vorliegen können, sodass auf eine schrittweise Transformation auf Textebene zurückgegriffen werden müsste. Um den sich daraus ergebenden Text-Parse-Aufwand zu umgehen, sind die Schritte der RST-Baum-Erzeugung, der Generierung von DSyntR-Nachrichten und der Verfeinerung der Nachrichten durch die Bereitstellung differenzierender Acceleo-Abbildungsschablonen zusammengefasst. Zu den in Tabelle 6.2 dargestellten Abbildungsvorschriften werden für die in Abbildung 6.6 dargestellten Beispielrelationen Schablonen für aggregierte DSyntR-Nachrichten ergänzt. Die Selektion der passenden Schablone erfolgt demnach stufenweise, indem zunächst nach einer geeigneten aggregierten Schablone gesucht und erst in zweiter Instanz auf eine isolierte Schablone zurückgegriffen wird. Die Entscheidung, welche Acceleo-Schablone zur Transformation einer oder mehrerer DPIL-Regeln verwendet wird, ist vom positiven Resultat bei der Prüfung des jeweiligen Guard-Ausdrucks abhängig. Da durch die Gruppierung anhand des Makro-Namens bereits eine inhaltliche Beziehung zwischen den Regeln sichergestellt ist, enthalten diese Guard-Ausdrücke lediglich Überprüfungen bezüglich der von einer aggregierten Schablone erwarteten Parameterüberschneidungen. Ist eine solche Guard-Prüfung erfolgreich, wird statt zweier einzelner DSyntR-Nachrichten die im Rumpf der Acceleo-Schablone enthaltene aggregierte Form erzeugt.

*Abbildung von DPIL-Regelschablonen auf DSyntR-Nachrichtenschablonen*

DPIL-Regelschablonen werden auf DSyntR-Nachrichtenschablonen abgebildet (Abschnitt 6.2). Letztere enthalten Abfragen für die dynamischen Inhalte aus dem jeweiligen Quell-DPIL-Modell. Diese Abfragen können, wie auch die eben erwähnten Guard-Ausdrücke, mittels OCL beschrieben werden. Zur besseren Lesbarkeit erlaubt Acceleo zudem den Aufbau hierarchischer Schablonen. Somit wird jeder dynamische Anteil wiederum durch eine eigene Schablone abgefragt. Das ist in Codeausschnitt 7.3 anhand einer Darstellung der Implementierung der *@Mod(:rule)*-Abfrage verdeutlicht.

```
[template public mod(r : ProcessRule) ? (r.type = ProcessRuleType::Hard)]
   must
[/template]
[template public mod(r : ProcessRule) ? (r.type = ProcessRuleType::Soft)]
   should
[/template]
```

Codeausschnitt 7.3: Schablonen dynamischer Inhalte für DSyntR-Nachrichtenschablonen

In der Rahmenschablone für die jeweilige DSyntR-Nachricht kann an jeder beliebigen Stelle eine der *mod*-Schablonen verwendet werden, um die Modalität der fraglichen DPIL-Regel auf einen passenden natürlichsprachlichen Ausdruck abzubilden. Die Guard-Ausdrücke werden dazu verwendet, um je nach Typ der Prozessregel entweder „must" oder *should* zurückzuliefern. In ähnlicher Art und Weise sind alle übrigen Abfragen implementiert.

Im letzten Schritt erfolgt die Oberflächenrealisierung der erzeugten DSyntR-Nachrichten. Diese wird in Acceleo als sogenannte *Nachbedingung* (engl. *post condition*) realisiert. Diese Nachbedingungen enthalten Funktionsaufrufe, welche den durch die Acceleo-Schablone erzeugten Text einer Nachbearbeitung unterziehen. Dieser Mechanismus wird hier genutzt, um die Oberflächenrealisierung zu initiieren. Dies geschieht über einen sogenannten *Acceleo Java Service Call*, welcher den Aufruf von reinen Java-Methoden aus den deklarativen Acceleo-Spezifikationen heraus gestattet. Im Falle der Oberflächenrealisierung wird hier der *RealPro Realizer*[7] verwendet, um aus den erstellten DSyntR-Nachrichten natürlichsprachliche Sätze zu erzeugen. Damit ist die Generierung natürlichsprachlicher Beschreibungen des Eingabe-DPIL-Modells für ein bestimmtes kommunikatives Ziel und Nutzermodell (Abschnitt 6.2.1) abgeschlossen.

*Oberflächenrealisierung mit dem RealPro Realizer*

### 7.5.2 *Bedienung*

Da auch die im Rahmen dieser Arbeit entwickelte NLG-Technik für DPIL-Modelle auf Acceleo basiert, ähnelt die Bedienung dieses Werkzeugs dem ersten Schritt bei der Verwendung des weiter oben beschriebenen Spurgenerators. Ziel des aktuellen Abschnitts ist es, einige ausgewählte Hinweise zur Bedienung des NLG-Werkzeugs bereitzustellen.

Für NL4DP existieren zwei Ausführungsmodi. Einerseits kann das NLG-Werkzeug wie gewohnt über das bereits bei der Bedienung des Spurgenerators erwähnte Kontextmenü des DPIL-Editors gestartet werden (Abbildung 7.15). Andererseits besteht die Möglichkeit, es als sogenannte *Standalone-Applikation* zu starten. Das geschieht über beispielsweise einen Kommandozeilenaufruf oder auch den programmatischen Aufruf der entsprechenden *main*-Methode. In beiden Fällen sind einerseits der absolute Pfad zum zu transformierenden DPIL-Modell und andererseits der Zielordner für das generierte Artefakt als Parameter zu übergeben. Dies ist eine Anforderung des Acceleo-Rahmenwerks.

*Start über Kontextmenü des DPIL-Editors oder als eigenständige Applikation*

Unabhängig davon, welcher Ausführungsmodus verwendet wird, werden seitens des Acceleo-basierten Generators für natürlichsprachliche Texte zwei weitere Parameter berücksichtigt: *(i)* Das Kommunikative Ziel und *(ii)* das Nutzermodell. Hierfür

7 http://www.cogentex.com/technology/realpro/, zuletzt aufgerufen: 17.05.2017

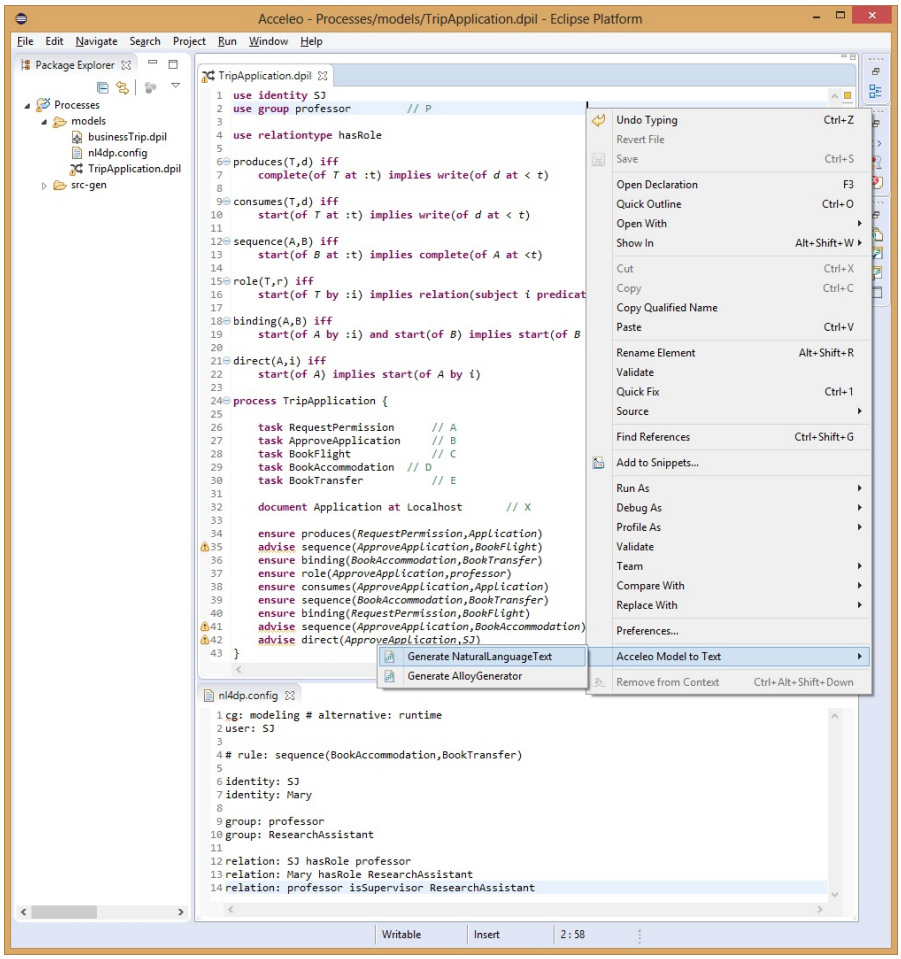



Abbildung 7.15: Verwendung von NL4DP für die Generierung natürlichsprachlicher Beschreibungen von DPIL-Modellen

ist es in der aktuellen Implementierung Konvention, eine Konfigurationsdatei mit dem Namen „nl4dp.config" in dem Verzeichnis anzulegen, in welchem sich auch das Quell-DPIL-Modell befindet. Ein Beispiel für den Inhalt einer solche Konfigurationsdatei ist ebenfalls in Abbildung 7.15 (unterer Bereich) dargestellt. In der ersten Zeile ist das kommunikative Ziel auf die Beschreibung der Modellinhalte zum Modellierungszeitpunkt festgelegt (*modeling*). Ändert man diese Einstellung auf *runtime*, dann ist das der Hinweis für das NLG-Werkzeug, eine natürlichsprachliche Erklärung für Prozessregeln zum Zeitpunkt der Prozessausführung zu liefern. Welche Regeln konkret erläutert werden sollen, wird mittels des Schlüsselwortes *rule* festgelegt (Zeile 4). Falls die Konfiguration des kommunikativen Ziels jedoch *modeling* ist, dann werden alle *rule*-Einträge ignoriert.

*Externe Konfiguration des kommunikativen Ziels*

Hinter dem Schlüsselwort *user* kann der Nutzer festgelegt werden, für den der natürlichsprachliche Text erzeugt wird. Die übrigen Einstellungsmöglichkeiten bilden das zu verwendende Organisationsmodell, zu dem auch dieser Nutzer gehört. Dadurch ist NL4DP in der Lage, personalisierte Textabschnitte zu erzeugen. Bei der in der Abbildung dargestellten Konfiguration würde hier beispielsweise die Erklärung der letzten Regel des Modells (`advise direct...`) personalisiert werden. Die resultierende Erläuterung der Regel in Isolation könnte dann beispielsweise

*Externe Konfiguration des Nutzermodells für personalisierte Textabschnitte*

wie folgt lauten: „The application should be approved by yourself." Diese Information erhält das System durch die Verknüpfung der zu erklärenden DPIL-Regel mit der Zielperson der NLG-Aufgabe. Wäre stattdessen „Mary" konfiguriert, dann kann der Satz auf „... by your supervisor" enden. Die Spezifikation des Nutzermodells in der NL4DP-Konfiguration ist aktuell eine Vereinfachung, die notwendig ist, da die Zielplattform *Process Navigation* [199] zum aktuellen Zeitpunkt noch keine Schnittstelle zur Abfrage des verwendeten Organisationsmodells bereitstellt.

Besonders bei der Verwendung von NL4DP zum Zeitpunkt der Prozessausführung ist eine dynamische Konfiguration mittels der beschriebenen Konfigurationsdatei ein unnötiger Umweg. Da bei der Ausführung von NL4DP als Standalone-Applikation bereits die Parameterwerte für Quellmodell und Zielordner übergeben werden müssen, besteht zusätzlich die Möglichkeit, das kommunikative Ziel, den Nutzer und die zu erklärenden Regeln per Leerzeichen getrennt als Parameter zu übergeben. Dieser Parametrierung wird gegenüber den Inhalten der Konfigurationsdatei Vorrang gewährt. Auch hier gilt, dass die so übergebenen DPIL-Regeln nur berücksichtigt werden, wenn das kommunikative Ziel *runtime* ist.

Das Ergebnis jeder Generierung ist eine Textdatei, welche den Namen des zu erläuternden Prozesses und die Dateierweiterung *.nlg* trägt.

### 7.5.3 *Erweiterung*

Da sich das im Rahmen dieser Arbeit entwickelte NLG-Werkzeug im Stadium eines exemplarischen Durchstichs befindet, steht aktuell lediglich ein Erweiterungsmechanismus zur Verfügung, welcher die Menge der DPIL-Regelschablonen ergänzt, für die eine passende DSyntR-Nachrichtenschablone vorliegt. Das Vorgehen bei einer Erweiterung dieser Art wird im aktuellen Abschnitt skizziert.

*Erweiterung: Zusätzliche DPIL-Regelschablonen*

Ähnlich wie bei Erweiterung der MuDePS-Implementierung, muss auch im Falle von NL4DP die Menge der angegebenen Transformationsregeln erweitert werden, um eine Unterstützung zusätzlicher DPIL-Regelschablonen zu ermöglichen. Dazu muss eine neue Acceleo-Schablone angelegt werden, deren Guard-Ausdruck die eindeutige Assoziation mit der gewünschten DPIL-Regelschablone sicherstellt. Im Rumpf der Schablone muss das Gerüst einer DSyntR-Nachricht angegeben werden. Dabei kann von den bereits vorhandenen Abfragen dynamischer Modellinhalte Gebrauch gemacht werden. Weiterhin ist es möglich, Schablonen für neue Nachrichten-Aggregationen anzugeben, welche die DSyntR-Nachrichten der neu hinzugekommenen DPIL-Regelschablonen vor dem Hintergrund möglicher rhetorischer Relationen verschmelzen.

Hauptziel von Abschnitt 7.5 ist es, das im Rahmen dieser Arbeit entwickelte Konzept der Generierung natürlichsprachlicher Beschreibungen von DPIL-Prozessmodellen (Abschnitt 6.2) durch eine prototypische Implementierung zu evaluieren. Diese ist im Kern eine Modell-zu-Text-Transformation. Wesentlich ist hierbei, dass aufgrund der fehlenden Metamodelle für RST-Bäume und DSyntR-Nachrichten eine *reine* Modell-zu-Text-Transformation anstelle einer Kombination aus dieser und mehrerer Modell-zu-Modell-Transformationsschritte realisiert ist. Die sich daraus ergebende Konsequenz ist die Zusammenfassung der Schritte zur Erzeugung des RST-Baums, der DSyntR-Nachrichten und der Verfeinerung von letzteren in dif-

ferenzierenden Acceleo-Abbildungsschablonen. Diese zwei Datenstrukturen, die als Zwischenrepräsentation für die Spezifikation des Zieltextes genutzt werden, sollten für zukünftige Weiterentwicklungen Modellcharakter haben. Dies bedeutet, dass die dafür notwendigen Metamodelle und gegebenenfalls auch eine eigene Modellierungssprache für einen erhöhten Modellierungskomfort entwickelt werden sollten.

# 8 EVALUATION: EIGENSCHAFTEN, KORREKTHEIT UND NÜTZLICHKEIT DER LÖSUNGEN

Mit den im vorherigen Kapitel beschriebenen Implementierungen wird die *Realisierbarkeit* der im zweiten Teil der Arbeit vorgestellten Konzepte sichergestellt. Noch ausstehend ist eine Bewertung der korrekten Funktionsweise, der Leistungsfähigkeit und, falls möglich, ein Vergleich mit ähnlichen Lösungen. Der aktuelle Abschnitt schließt diese Lücken zu großen Teilen.

In Abschnitt 8.1 respektive Abschnitt 8.2 werden die statischen Eigenschaften der Technik zur Spurgenerierung für multi-perspektivische, deklarative Prozessmodelle und die des induktiven Translationsprinzip analysiert. Zusätzlich wird für den Spurgenerator eine Laufzeitanalyse durchgeführt, um die Abhängigkeit der Berechnungszeit von der gewählten Parametrierung einschätzen zu können. Die Korrektheit der Funktionsweise des Spurgenerators und der induktiven Translationstechnik wird in Abschnitt 8.3 beziehungsweise Abschnitt 8.4. Den vorletzten Betrachtungsgegenstand dieses Kapitels bildet eine qualitative Umfrageevaluation (Abschnitt 8.5), welche erste Erkenntnisse für den potentiellen Nutzen aller drei entwickelten Konzepte liefern soll. Alle Teilevaluationen dienen weiterhin als Grundlage für eine Gesamteinschätzung (Abschnitt 9.1) der Erfüllung aller grundlegenden Anforderungen aus Abschnitt 1.2.

## 8.1 EIGENSCHAFTEN: SIMULATION MULTI-PERSPEKTIVISCHER, DEKLARATIVER PROZESSMODELLE

Ziel dieser ersten Evaluation ist die Analyse statischer Eigenschaften und des Laufzeitverhaltens des in Kapitel 4 vorgestellten Ansatzes zur Spurgenerierung. Als Grundlage für ersteres dienen Merkmale und Funktionalität verwandter Ansätze (Abschnitt 4.1). Eine Laufzeitanalyse wird in Abhängigkeit der Modellkomplexität und des Umfangs der Ereignisprotokolle durchgeführt.

### 8.1.1 *Statische Eigenschaften*

*Merkmale und Funktionen von MuDePS*

Tabelle 8.1 zeigt in kompakterer Form die bereits bekannte Übersicht über existierende Spurgeneratoren, in welche nun aber auch der im Rahmen dieser Arbeit entwickelte Spurgenerator (MuDePS) eingeordnet ist. Die Tabelle zeigt Merkmale und Funktionen, die auf den jeweiligen Ansatz zutreffen (✓), nicht (-) oder anteilig ((✓)) zutreffen. Merkmale oder Funktionalitäten können zudem irrelevant (0) sein, wie beispielsweise die Organisatorische Perspektive im Falle der Simulation rein kontrollfluss-orientierter Petri-Netze. Schließlich sind für den neu hinzugekommenen Spurgenerator auch explizite Erweiterungspunkte (E) definiert worden. Nachfolgend werden einige der Eigenschaften diskutiert, während andere in Kombination mit der Beschreibung aus Abschnitt 4.1 selbsterklärend sind.

| | PN | | | | PT | CPN | BPMN | YAWL | Declare | | DCR | Hybrid | MuDePS |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
| | [36, 97, 168, 173] | | | | [92] | [124] | [33, 34] | [156] | [41, 193] | | [122] | [194] | Kapitel 4 |
| Unterstützte Prozessperspektiven | | | | | | | | | | | | | |
| – Funktionale | ✓ | ✓ | ✓ | ✓ | ✓ | ✓ | ✓ | ✓ | ✓ | ✓ | ✓ | ✓ | ✓ |
| – Verhaltensorientierte | ✓ | ✓ | ✓ | ✓ | ✓ | ✓ | ✓ | ✓ | ✓ | ✓ | ✓ | ✓ | ✓ |
| – Datenorientierte | 0 | 0 | (✓) | 0 | 0 | ✓ | ✓ | ✓ | 0 | (✓) | ✓ | (✓) | ✓ |
| – Organisatorische | 0 | 0 | (✓) | 0 | 0 | ✓ | – | ✓ | 0 | (✓) | (✓) | (✓) | ✓ |
| – Operationale | 0 | 0 | 0 | 0 | 0 | – | – | 0 | 0 | 0 | 0 | 0 | (E) |
| Ereignistypen: Start/Ende | – | – | – | ✓ | ✓ | ✓ | ✓ | ✓ | – | – | – | – | (E) |
| Dynamische Beziehungen | – | – | (✓) | – | – | – | – | – | 0 | – | – | – | ✓ |
| Perspektivenübergr. Beziehungen | 0 | 0 | 0 | 0 | 0 | – | – | – | 0 | – | – | – | ✓ |
| Konfigurierbare Entscheidungen | ✓ | – | – | (✓) | – | ✓ | – | ✓ | – | ✓ | – | ✓ | ✓ |
| Nativ garantierte Terminierung | ✓ | – | ✓ | – | ✓ | – | ✓ | (✓) | – | – | ✓ | (✓) | – |
| Wahlweise künstliches Rauschen | | | | | | | | | | | | | |
| – Zusätzliche Aktivitäten | ✓ | – | ✓ | ✓ | (✓) | – | ✓ | – | – | – | – | – | ✓ |
| – Überspringen von Aktivitäten | ✓ | – | ✓ | ✓ | ✓ | – | ✓ | – | – | – | – | – | ✓ |
| – Verzögerung von Aktivitäten | – | – | ✓ | – | – | – | – | – | – | – | – | – | – |
| – Vertauschen von Aktivitäten | – | – | – | ✓ | – | – | ✓ | – | – | – | – | – | ✓ |
| – Unautorisierte Ressourcen | – | – | ✓ | – | – | – | – | – | – | – | – | – | ✓ |
| – Verfälschen von Datenwerten | – | – | – | – | – | – | ✓ | – | – | – | – | – | ✓ |
| – Rauschprofile | – | – | – | – | – | – | ✓ | – | – | – | – | – | ✓ |
| Konfigurierbare Prozessumgebung | – | – | ✓ | – | – | – | – | ✓ | – | – | – | – | (✓) |
| Konfigurierbarer Startzustand | – | – | – | – | – | – | – | ✓ | – | – | – | – | ✓ |
| Ressourcen-Bias | – | – | – | – | – | – | – | – | – | – | ✓ | – | ✓ |
| Export in Standardformat | ✓ | ✓ | ✓ | ✓ | ✓ | ✓ | ✓ | ✓ | ✓ | (✓) | – | (✓) | ✓ |

Tabelle 8.1: Vergleich von MuDePS mit existierenden Spurgeneratoren

*Prozessperspektiven*

Die erste Kategorie von Merkmalen setzt sich aus der Unterstützung der einzelnen Prozessperspektiven zusammen. Gemäß Tabelle 8.1 unterstützt MuDePS vier der fünf geläufigsten Prozessperspektiven. Einzig die Operationale Perspektive wird ausgespart. Jedoch ist mit der in Abschnitt 4.2 beschriebenen generischen Signatur `AssociatedElement` ein Erweiterungspunkt definiert, der die Assoziation weiterer Schlüssel-Wert-Paare mit jedem zu protokollierenden Ereignis gestattet.

*Ereignistypen*

Im bisherigen Konzept des Spurgenerators ist keine Unterstützung verschiedener Status einer Aktivitätsausführung und diesbezüglich verschiedener Ereignistypen gegeben. Dies schränkt die Aussagekraft der Simulationsergebnisse in der Form ein, dass keine parallele Ausführung zweier oder mehrerer Aktivitäten protokolliert werden kann. Außerdem ist es nicht möglich, die Durchlaufzeit durch eine Instanz einer Aktivität zu simulieren. Dies hat für die Verwendungen des Spurgenerators im Kontext der Verbesserung der Verständlichkeit jedoch keinen Einfluss. Auch das in Abschnitt 5.2 vorgestellte Prinzip zur Translation von Prozessmodellen wird dadurch nicht beeinflusst, da Process-Mining-Techniken lediglich die *Reihenfolge* der Ausführung einer Aktivität beachten. Ist die Reihenfolge zweier Aktivitäten in zwei verschiedenen Ausführungsspuren vertauscht, gehen Process-Mining-Techniken davon aus, dass die Reihenfolge dieser beiden Aktivitäten irrelevant ist. Somit kann im Modell die Möglichkeit der parallelen Ausführung vermerkt werden.

*Konfigurierbare Entscheidungen*

Konfigurierbare Entscheidungen bezeichnen im Kontext deklarativer Prozessmodelle die Priorisierung bestimmter Aktivitäten, die Festlegung statischer Spurbestandteile sowie Ausschlüsse anderer Spurbestandteile. Derartige Entscheidungen können in dem in dieser Arbeit vorgestellten Spurgenerator modelliert werden. Dazu sind verschiedene Alloy-Prädikate vordefiniert, die bei ihrer Instanziierung die konfigurierten Entscheidungen umsetzen.

*Native Terminierung*

Da MuDePS auf Werkzeugen zum Lösen von Erfüllbarkeitsproblemen aufbaut, ist eine native Terminierung unmöglich. Grund ist, dass – vor allem durch die Unterstützung der Datenorientierten Perspektive – ein nicht-endlicher Wertebereich für mögliche Variablenbelegungen modelliert werden kann. Die Erfüllbarkeit, auf der die Generierung der Spuren basiert, wäre dadurch im Allgemeinen unentscheidbar (nach Boris Trakhtenbrot (1950) als engl. Zusammenfassung [132, S. 279]). Aus diesem Grund muss der Lösungsraum begrenzt werden, was gleichzeitig die Definition eines künstlichen Terminierungkriteriums darstellt.

*Künstliches Rauschen*

Der Spurgenerator ist in der Lage, nach Bedarf künstliches Rauschen in den Ereignisprotokollen zu verursachen. Auch hierfür sind entsprechende Alloy-Prädikate definiert, die entweder einzelne Verstöße gegen das Modell oder ganze Rauschprofile definieren. Davon ausgenommen ist das künstliche Verzögern von Aktivitäten. Dies kann durch die Absenz der Unterscheidung von Start und Ende der Bearbeitung einer Aktivität nicht abgebildet werden, da die Repräsentation der Durchführungsdauer der Aktivität fehlt.

*Prozessumgebung*

Die Prozessumgebung ist nur rudimentär konfigurierbar. Lediglich die Ankunftsrate der Prozessinstanzen kann parametriert werden. Andere Aspekte wie die Verfügbarkeit der Prozessteilnehmer sind im aktuellen Zustand weder vorgesehen, noch in Alloy direkt abzubilden.

Mit der Integration des Spurgenerators in die RapidMiner-Plattform und die Wiederverwendung der RapidProM-Datenstruktur für Ereignisprotokolle ist ein Export derselben im XES-Standardformat ohne weiteres möglich.

Zusammenfassend ist zu sagen, dass der im Rahmen dieser Arbeit vorgestellte Spurgenerator etwa 76% aller identifizierten Funktionen und Merkmale abdeckt. Zählt man die definierten Erweiterungspunkte sowie rudimentäre Umsetzungen hinzu, steigert sich dieser Anteil auf rund 90%. Da vorab keine Wertigkeit für die unterschiedlichen Funktionen und Eigenschaften vereinbart sind, können die konkreten Prozentwerte lediglich qualitativ gewertet werden. In jedem Fall übersteigt somit die Funktionsunterstützung die aller aufgeführten und dem Autor bekannten Spurgeneratoren.

### 8.1.2 *Laufzeitanalyse*

Ziel des aktuellen Abschnitts ist es, die Leistungsfähigkeit des Spurgenerators durch die Erfassung der Berechnungszeiten in Abhängigkeit von verschiedenen Parametrierungen zu analysieren. Dazu werden mehrere Versuche für DPIL-Modelle unterschiedlichen Umfangs und für unterschiedliche Zielgrößen der resultierenden Ereignisprotokolle durchgeführt. Die Untersuchungen zeigen, dass die Berechnungszeiten eines Ereignisprotokolls primär von der Anzahl und Länge der zu erzeugenden Prozessausführungsspuren abhängig ist. Auf eine formale Komplexitätsanalyse wird an dieser Stelle verzichtet, da die Komplexität vom gewählten Solver für das in Alloy formulierte Erfüllbarkeitsproblem abhängig ist. Zwar gilt dies auch für die Messung der in diesem Abschnitt betrachteten Berechnungszeit, aber der nachfolgend beschriebene Versuchsaufbau ist für die Überprüfung der Laufzeit des Spurgenerators mit jedem *beliebigen* Solver geeignet.

Der Versuchsaufbau berücksichtigt insgesamt vier Dimensionen, um deren Einfluss auf die Berechnungszeit geprüft zu prüfen. Die vier Dimensionen sind:

*Anzahl der Modell-Entitäten*

- *E:* Die Anzahl der im DPIL-Modell enthaltenen Entitäten (*E*) definiert die Anzahl möglicher Ereignistypen, die sich wiederum über unterschiedliche Assoziationen der Entitäten definieren. Diese Dimension gibt letztlich Aufschluss darüber, wie viele sich in mindestens einer Assoziation unterscheidende Prozessausführungsspuren *potentiell* generiert werden können.

*Anzahl der Prozessregeln*

- *R:* Die zweite betrachtete Dimension der Modellgröße ist die Anzahl der im DPIL-Modell enthaltenen Prozessregeln (*R*). Diese selektieren in ihrer Kombination die Teilmenge der *validen*[1] aus der Menge aller potentiell zu generierenden Prozessausführungsspuren.

*Anzahl der Spuren*

- *N:* Eine Dimension des Umfangs eines Ereignisprotokolls ist die *Anzahl (N)* der maximal zu erzeugenden Prozessausführungsspuren. Dieser Parameter kann nur als obere Grenze modelliert werden, da die Anzahl der bezüglich der Prozessregeln validen Prozessausführungsspuren auch kleiner als *N* sein kann.

*Länge der Spuren*

- *L:* Die letzte Dimension ist die maximale *Länge (L)* der protokollierten Prozessausführungsspuren. Auch diese ist eine obere Grenze, da auch die Anzahl der aufeinanderfolgenden Ereignisse in einer Prozessinstanz durch Prozessregeln stärker eingeschränkt sein kann.

*Beispielmodelle*

Um die Anzahl der Modellentitäten und -regeln variieren zu können, werden Prozessmodelle unterschiedlicher Größe manuell definiert. Jede für die aktuelle Untersuchung gewählte Konstellation der Anzahl an Entitäten und der Anzahl an Regeln spiegelt sich in einem solchen Modell wider. Dabei wird darauf geachtet, die Prozessregeln so zu definieren, dass zumindest *eine* Aktivität unbegrenzt oft wiederholbar ist. Dies ist notwendig, um mit einer sehr geringen Anzahl an unterschiedlichen Aktivitäten beliebig lange Prozessausführungsspuren zu produzieren. Auch die Anzahl der generierten Spuren ist von der jeweiligen Regelkonstellation abhängig. Aus diesem Grund wird auch darauf geachtet, dass trotz der definierten Regeln die gewünschte Anzahl an unterschiedlichen Ausführungsspuren erzeugt werden können.

*Parametrierungen*

Für die obenstehenden einzelnen Einflussgrößen auf die Berechnungszeit der künstlichen Ereignisprotokolle wird jeweils eine Reihe von Parametrierungen festgelegt. Anschließend wird ein RapidMiner-Prozess[2] definiert, welcher die kombinatorische Vielfalt aller möglichen Wertkombinationen bildet und zur Konfiguration des Spurgenerators verwendet. Für jede dieser Kombinationen wird die Berechnungszeit des Spurgenerators ermittelt. Die Ergebnisse sind in Abbildung 8.1 dargestellt.

1 Eine Analyse der Berechnungszeit für nicht-valide Prozessausführungsspuren wird nicht durchgeführt, da diese Generierung auf demselben Algorithmus beruht, wie die Generierung der validen Spuren.

2 Der Prozess selbst sowie die verwendeten Beispielmodelle sind unter `http://mps.kppq.de` zum Download verfügbar.

| | | 10 | 10 | 10 | 20 | 20 | 20 | 30 | 30 | 30 | 40 | 40 | 40 | 50 | 50 | 50 | 60 | 60 | 60 | 70 | 70 | 70 | 80 | 80 | 80 | L |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
| | | 10 | 100 | 1000 | 10 | 100 | 1000 | 10 | 100 | 1000 | 10 | 100 | 1000 | 10 | 100 | 1000 | 10 | 100 | 1000 | 10 | 100 | 1000 | 10 | 100 | 1000 | N |
| 5 | 0 | 0,42 | 0,69 | 5,81 | 0,64 | 1,81 | 14,47 | 1,02 | 3,45 | 23,92 | 4,83 | 22,61 | 63,69 | 5,39 | 27,30 | 90,52 | 9,19 | 35,98 | 121,91 | 31,25 | 133,11 | 443,90 | 44,21 | 140,15 | 592,75 | |
| 5 | 5 | 0,38 | 0,94 | 7,24 | 1,38 | 3,50 | 23,62 | 3,05 | 12,30 | 50,77 | 12,41 | 50,91 | 133,21 | 18,25 | 80,05 | 235,68 | 30,28 | 119,96 | 342,49 | 83,25 | 202,79 | 942,28 | 129,60 | 325,75 | 1403,77 | |
| 5 | 10 | 0,50 | 0,56 | 0,53 | 2,36 | 3,47 | 3,67 | 6,05 | 29,34 | 28,50 | 21,56 | 105,30 | 104,64 | 37,70 | 194,68 | 195,76 | 55,47 | 340,99 | 341,57 | 138,77 | 576,49 | 610,97 | 187,68 | 554,37 | 1449,14 | |
| 5 | 15 | 0,44 | 0,44 | 0,44 | 2,75 | 3,72 | 3,74 | 7,05 | 22,83 | 21,22 | 27,00 | 121,55 | 121,80 | 53,72 | 260,22 | 269,14 | 81,47 | 339,57 | 338,50 | 198,01 | 671,04 | 667,72 | 396,46 | 693,23 | 1983,21 | |
| 5 | 20 | 0,42 | 0,45 | 0,42 | 2,88 | 3,14 | 2,99 | 7,41 | 12,92 | 11,55 | 28,14 | 60,75 | 61,11 | 53,72 | 96,18 | 95,69 | 63,88 | 157,38 | 164,04 | 202,45 | 798,23 | 784,07 | 245,48 | 775,54 | 776,85 | |
| 10 | 0 | 1,45 | 1,14 | 6,05 | 1,25 | 2,16 | 15,14 | 1,52 | 4,52 | 24,49 | 5,52 | 21,16 | 66,77 | 5,34 | 26,08 | 102,10 | 8,06 | 35,64 | 148,59 | 33,64 | 104,19 | 470,26 | 30,53 | 112,82 | 563,59 | |
| 10 | 5 | 0,58 | 1,19 | 7,53 | 1,80 | 5,34 | 25,73 | 3,97 | 12,91 | 53,31 | 10,58 | 44,22 | 138,10 | 16,83 | 76,66 | 226,48 | 17,77 | 88,85 | 332,71 | 58,40 | 170,96 | 950,24 | 101,16 | 305,55 | 1311,19 | |
| 10 | 10 | 0,89 | 1,50 | 8,38 | 1,98 | 6,83 | 28,67 | 4,59 | 15,27 | 61,02 | 14,75 | 64,83 | 173,06 | 29,85 | 95,83 | 295,14 | 37,38 | 116,75 | 453,65 | 94,66 | 237,98 | 1314,16 | 131,68 | 378,04 | 1750,23 | |
| 10 | 15 | 0,70 | 1,22 | 8,03 | 2,98 | 5,45 | 28,13 | 4,55 | 15,52 | 61,74 | 16,89 | 61,75 | 165,39 | 27,44 | 85,21 | 313,77 | 37,10 | 116,27 | 424,43 | 91,49 | 240,64 | 1236,29 | 134,45 | 405,48 | 1832,74 | |
| 10 | 20 | 0,66 | 1,31 | 9,27 | 3,45 | 5,35 | 30,09 | 6,49 | 18,50 | 65,30 | 21,08 | 66,22 | 171,29 | 31,19 | 99,35 | 282,76 | 53,90 | 175,57 | 459,99 | 112,85 | 279,13 | 1078,55 | 228,05 | 472,69 | 1693,12 | |
| 15 | 0 | 0,53 | 1,24 | 6,84 | 0,92 | 1,89 | 18,16 | 1,49 | 5,44 | 28,58 | 4,53 | 18,30 | 68,56 | 7,31 | 28,78 | 106,13 | 9,03 | 34,35 | 146,95 | 29,69 | 62,42 | 491,68 | 24,55 | 81,91 | 635,00 | |
| 15 | 5 | 0,55 | 1,42 | 28,83 | 1,59 | 5,91 | 48,78 | 4,02 | 13,16 | 78,51 | 12,41 | 44,55 | 182,87 | 19,66 | 81,51 | 280,47 | 29,80 | 105,05 | 447,71 | 88,06 | 195,57 | 1290,78 | 89,69 | 218,60 | 1583,33 | |
| 15 | 10 | 1,70 | 1,61 | 9,86 | 3,44 | 6,24 | 32,95 | 6,74 | 16,75 | 70,38 | 43,64 | 49,28 | 191,54 | 31,60 | 90,63 | 325,54 | 43,77 | 117,57 | 502,46 | 117,63 | 201,98 | 1316,20 | 151,30 | 356,49 | 2115,03 | |
| 15 | 15 | 2,22 | 1,77 | 9,52 | 3,63 | 8,41 | 34,33 | 5,92 | 19,34 | 73,31 | 21,44 | 58,50 | 186,35 | 31,17 | 94,58 | 305,28 | 43,75 | 137,13 | 511,60 | 108,80 | 241,48 | 1346,69 | 165,90 | 382,58 | 2095,00 | |
| 15 | 20 | 0,78 | 1,59 | 9,39 | 2,80 | 6,58 | 32,69 | 6,38 | 19,22 | 70,31 | 22,73 | 72,22 | 191,71 | 43,08 | 132,51 | 329,83 | 58,66 | 183,07 | 527,35 | 126,74 | 315,92 | 1370,65 | 196,62 | 541,11 | 1841,83 | |
| 20 | 0 | 0,52 | 0,99 | 6,73 | 0,80 | 2,67 | 16,78 | 1,28 | 5,27 | 29,49 | 6,31 | 20,41 | 67,68 | 7,94 | 32,91 | 114,13 | 8,98 | 33,69 | 138,51 | 29,41 | 88,71 | 570,97 | 40,06 | 81,30 | 714,59 | |
| 20 | 5 | 1,58 | 1,63 | 9,52 | 2,45 | 5,16 | 29,72 | 3,45 | 14,50 | 59,28 | 10,56 | 46,45 | 173,15 | 16,17 | 83,77 | 268,42 | 36,92 | 118,85 | 387,63 | 79,11 | 129,04 | 1066,90 | 77,58 | 192,11 | 1633,92 | |
| 20 | 10 | 0,75 | 1,53 | 9,89 | 2,24 | 6,94 | 33,71 | 4,94 | 21,44 | 76,00 | 14,29 | 67,24 | 212,28 | 30,80 | 94,96 | 344,10 | 44,78 | 134,88 | 555,68 | 95,05 | 197,33 | 1371,39 | 144,14 | 257,49 | 1997,83 | |
| 20 | 15 | 0,75 | 1,52 | 9,69 | 2,39 | 8,69 | 33,97 | 5,59 | 19,72 | 75,91 | 18,84 | 59,06 | 205,89 | 28,47 | 103,93 | 364,90 | 38,45 | 143,82 | 524,34 | 106,40 | 186,59 | 1319,93 | 175,42 | 296,10 | 2049,84 | |
| 20 | 20 | 0,70 | 1,50 | 9,53 | 3,11 | 8,36 | 34,13 | 5,74 | 17,19 | 72,33 | 23,41 | 75,03 | 210,62 | 33,80 | 113,93 | 339,67 | 73,28 | 201,83 | 534,81 | 143,88 | 347,37 | 1332,62 | 248,56 | 437,13 | 1890,98 | |
| E | R | | | | | | | | | | | | | | | | | | | | | | | | | |

Abbildung 8.1: Übersicht über Laufzeiten in Abhängigkeit von unterschiedlichen Parametrierungen

In den zweistufigen Spaltenköpfen der abgebildeten Matrix sind jeweils unterschiedliche Kombinationen aus Spurlänge und -anzahl dargestellt. Die Zeilenköpfe spiegeln folglich Kombinationen aus einer unterschiedlichen Anzahl an Entitäten und Regeln wider. Die Zelle, welche den Kreuzungspunkt einer bestimmten Spalte und einer bestimmten Zeile markiert, enthält die Berechnungszeit (in Sekunden), welche der Spurgenerator in der jeweiligen Konfiguration benötigt. Zudem sind die Zellen farblich hervorgehoben, um den Vergleich einzelner Konfigurationen zu erleichtern. Vergleichsweise kurze Berechnungszeiten sind grün hervorgehoben, während längere rot dargestellt sind. Die Farbdarstellung ist jedoch im höchsten Grade vom festgelegten Wertebereich jeder Einflussgröße abhängig und dient somit nur der initialen Orientierung. Die Versuchsergebnisse reichen von Berechnungszeiten von etwa 400 Millisekunden bis hin zu rund 35 Minuten.

*Länge und Anzahl der Prozessausführungsspuren*

Verfolgt man die Entwicklung der eingetragenen Werte in horizontaler Richtung, dann lassen sich daraus jedoch bereits erste Rückschlüsse auf die Abhängigkeit der Berechnungszeit von Länge und Anzahl der Prozessausführungsspuren ziehen. Vertikale Wertentwicklungen geben dagegen Aufschluss über die Abhängigkeit von der Anzahl der Modellentitäten und -regeln. Für die verwendeten Konfigurationen wird so ersichtlich, dass die maximale Länge jeder Ausführungsspur und die Anzahl derselben die größten Einflussfaktoren für die Berechnungszeit darstellen. Diese Einschätzung liegt darin begründet, dass im vertikalen Verlauf eine deutlich geringere Wertentwicklung stattfindet, als im horizontalen.

Dies wird auch aus einer alternativen Darstellung (Abbildung 8.2) ersichtlich. Für das dargestellte Diagramm ist für die Anzahl der zu erzeugenden Ausführungsspuren ein konstanter Wert – 100 – eingestellt. Jede Säulengruppe zeigt die Berechnungszeit in Abhängigkeit der Anzahl an Entitäten und Regeln für eine konkrete maximale Spurlänge.

*Geringer Einfluss der Anzahl an Entitäten*

Aus der Darstellung ist erkennbar, dass die Anzahl der im Modell enthaltenen Entitäten einen vergleichsweise geringen Einfluss ausüben. Vergleicht man die Säulenniveaus bei konstanter Anzahl an Regeln, dann sind diese weitestgehend ähnlich. Anders ist es im umgekehrten Fall, was zeigt, dass die Anzahl der Regeln eine größere Rolle spielt.

Um eine Vergleichsbasis für die in diesem Abschnitt beschriebene Untersuchung herstellen zu können, sind auch Kenntnisse über die verwendete Systemkonfiguration bei der Versuchsdurchführung notwendig. Die Ergebnisse des Versuchs be-

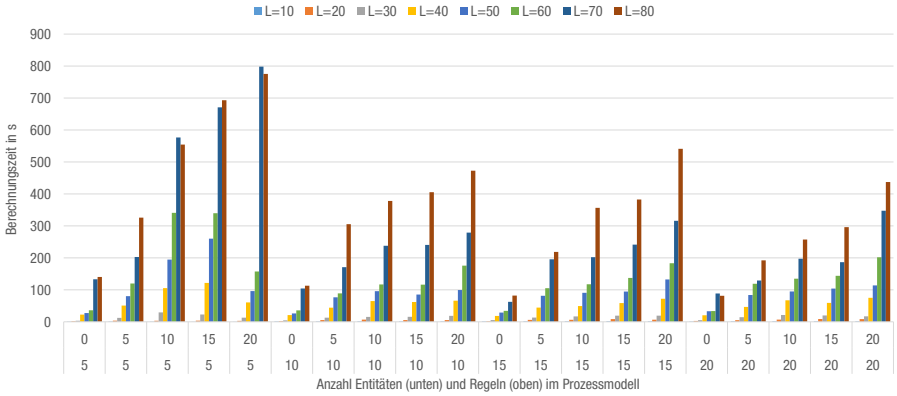



Abbildung 8.2: Berechnungszeiten in Abhängigkeit von der Anzahl der Prozessentitäten und -regeln sowie der maximalen Spurlänge bei $N = 100$

*Systemkonfiguration*

ziehen sich auf einen *Dell Latitude E6430* (Core i7-3720QM mit 8 x 2.6GHz, 16 GB memory, SSD-Festplatte und Windows 8 64 Bit). Für die Ausführung des Spurgenerators wird auf die ebenfalls Java-basierte RapidMiner-Plattform in der Version 7.5 zurückgegriffen. Diese operierte auf einer 64-Bit Java Virtual Machine mit einer maximalen Speichernutzung von 8 GB.

*Kein Vergleich mit existierenden Ansätzen sinnvoll*

Am aktuellen Abschnitt mag aufgefallen sein, dass kein Vergleich der Leistungsfähigkeit in Kapitel 4 vorgestellten Spurgenerators und existierenden Ansätzen durchgeführt wird. Zwar existiert zumindest ein weiterer solcher Ansatz im Bereich der deklarativen Prozessmodellierungssprachen, jedoch basiert dieser auf der perspektivisch stark eingeschränkten Sprache Declare. Dadurch kann, wie in Kapitel 4 dargelegt wird, für die Spurgenerierung von Techniken Gebrauch gemacht werden, die im multi-perspektivischen Fall nicht eingesetzt werden können. Ein Vergleich mit dieser Technik erbrächte die Erkenntnis, dass der Declare-Spurgenerator mit großem Abstand geringere Berechnungszeiten benötigt. Durch die deutlich geringere Abdeckung der Prozessperspektiven sowie weitere drastische Funktionsunterschiede wäre die Aussagekraft eines solchen Vergleichs äußerst gering.

Abschließend muss an dieser Stelle angemerkt werden, dass die Datengrundlage des in diesem Abschnitt beschriebenen Leistungstests potentiell „Ausreißer" beinhaltet. Durch die eigenständig gesteuerte Ressourcenverwaltung des Betriebssystems haben auch Hintergrundprozesse Einfluss auf die Berechnungszeiten. Dies hat zwar keine Auswirkungen auf die gewonnenen, qualitativen Erkenntnisse, erschwert aber die Interpretation der einzelnen Darstellungen. Folgerichtig ist geplant, den Versuch mehrfach zu wiederholen und die ermittelten Werte dadurch zu konsolidieren oder zu korrigieren.

Zweck von Abschnitt 8.1 ist die Analyse der Eigenschaften des entwickelten Spurgenerators für multi-perspektivische, deklarative Prozessmodelle (Abschnitt 4.2). Die Betrachtung der statischen Eigenschaften des Prinzips (Abschnitt 8.1.1) zeigt, dass nahezu die vollständige von vergleichbaren Ansätzen bereitgestellte Funktionalität (Abschnitt 4.1) umgesetzt ist. Testdurchläufe mit unterschiedlichen Konfigurationen ergaben zudem, dass die Laufzeit hauptsächlich von Anzahl und Länge der zu generierenden Prozessausführungsspuren aber kaum vom Umfang des Prozessmodells abhängig ist.

## 8.2 EIGENSCHAFTEN: INDUKTIVE TRANSLATION VON PROZESSMODELLEN

Ziel des aktuellen Abschnitts ist es, festzustellen, wie stark das im Rahmen der vorliegenden Arbeit entwickelte induktive Translationsprinzip die Abdeckung für Translationssysteme hinsichtlich möglicher Sprachpaarungen verbessert. In Abschnitt 5.1 wird analysiert, für welche Paarungen der verbreitetsten Prozessmodellierungssprachen eine adäquate Übersetzungstechnik existiert. Das Ergebnis zeigt, dass für etwa 10% der möglichen Paarungen direkte Übersetzungsmöglichkeiten bestehen, aus denen sich wiederum transitive Übersetzungsmöglichkeiten ergeben. Dies steigert die Abdeckung auf etwa 20%. Das Gros setzt sich dabei aus Techniken zur Translation imperativer Modelle in andere imperative Sprachen zusammen. In diesem Bereich sind etwa 45% der Sprachpaarungen durch eine direkte Translation abgedeckt und unter Nutzung transitiver Übersetzungsmöglichkeiten steigert sich dies auf rund 86%. In jedem Fall bergen transitive Übersetzungsansätze die Gefahr, dass eine oder mehrere der verwendeten Zwischenrepräsentationen über eine gegenüber der Zielsprache geringere Ausdrucksmächtigkeit verfügen. Die Folge ist ein *garantierter* Informationsverlust.

*Abdeckung durch induktives Translationsprinzip*

Das im Rahmen der vorliegenden Arbeit vorgestellte induktive Prinzip zur Prozessmodelltranslation ist naturgemäß fehlerträchtig. Eine spätere Evaluation der Korrektheit zeigt – für konkrete Sprachpaare – jedoch, dass dieser Fehler minimiert werden kann. Dadurch besteht in vielen Fällen das Potential für eine fehlerfreie Übersetzung. Die Verwendbarkeit des vorgestellten induktiven Translationsansatzes ist jedoch direkt von der Verfügbarkeit eines Spurgenerators und eines Process-Mining-Werkzeugs für das jeweilige Sprachpaar abhängig. Tabelle 8.2 zeigt eine kombinierte Darstellung der Abdeckung durch konservative Translationstechniken und den möglichen Sprachpaarungen bei Nutzung des induktiven Prinzips[3]. Dabei werden die Paarungen für das induktive Prinzip ausgeblendet, für die bereits eine konservative *direkte* Translationstechnik ($\checkmark_0$) existiert. Falls für ein Sprachpaar keine solche Technik zur Verfügung steht, jedoch das induktive Prinzip durch die Verfügbarkeit von Spurgenerator und Process-Mining-Technik angewendet werden kann, wird das in der Tabelle entsprechend vermerkt ($\checkmark_+$). Nicht jeder Spurgenerator und nicht jede Process-Mining-Technik unterstützt alle fünf Prozessperspektiven oder die Unterscheidung der zwei Ereignistypen, weswegen hier das induktive Prinzip nur mit Einschränkung anwendbar ist ($(\checkmark_{\ldots})$). Da sowohl konservative, transitive als auch induktive Translationstechniken einen Informationsverlust verursachen können, sind in Tabelle 8.2 bei Verfügbarkeit *beider* dieser Varianten auch beide gleichberechtigt dargestellt ($\checkmark_{\rightarrow|+}$). Durch die Ergänzung dieser neuen Übersetzungsmöglichkeiten vergrößert sich auch die Menge der transitiven Translationsmöglichkeiten ($\xrightarrow{\mathrm{x}}$), wobei hier in allen Fällen BPMN als Zwischensprache fungiert ($\xrightarrow{\mathrm{B}}$). In der Tabelle wird ein möglicher Unterschied in der Ausdrucksmächtigkeit von Quell- und Zielsprache der Translation nicht berücksichtigt. Eine tiefer gehende Analyse der Ausdrucksmächtigkeit würde, wie bereits in Abschnitt 5.1 angemerkt, den Rahmen der vorliegenden Arbeit sprengen.

*Gesamtabdeckung durch insgesamt verfügbare Translationssysteme*

3 Dass induktive Verfahren naturgemäß fehlerträchtig sind, wird an dieser Stelle außer Acht gelassen, da für eine diesbezüglich differenziertere Betrachtung eine ausgedehnte Versuchsreihe für *alle* möglichen Sprachpaarungen notwendig wäre.

| ↬ | UML 2 AD | BPMN | EPC | PN | CPN | PT | YAWL | CMMN | CLIMB | Declare | MP-Declare | DPIL | TLA+ | DCR | EM-BrA²CE |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
| UML 2 AD | — | ✓₀ | ✓₀ | ✓₀ | ✓₀ | ✓$_{B}$→ | ✓→ | | | ✓$_{B}$→ | ✓$_{B}$→ | ✓$_{B}$→ | | | |
| BPMN | ✓₀ | — | ✓₀ | ✓₀ | ✓₀ | ✓₊ | ✓₀ | | | ✓₊ | (✓₊) | (✓₊) | | | |
| EPC | ✓₀ | ✓₀ | — | ✓₀ | ✓→ | ✓$_{B}$→ | ✓₀ | | | ✓$_{B}$→ | ✓$_{B}$→ | ✓$_{B}$→ | | | |
| PN | ✓→ | ✓₀ | ✓→\|+ | — | ✓→\|+ | ✓₊ | ✓→\|+ | | | ✓₊ | ✓₊ | (✓₊) | | | |
| CPN | ✓→ | ✓→\|+ | ✓→\|(+) | ✓₀ | — | ✓₊ | ✓₀ | | | ✓₊ | (✓₊) | (✓₊) | | | |
| PT | ✓→ | ✓→\|+ | ✓→\|+ | ✓₀ | ✓→\|+ | — | ✓→\|+ | | | ✓₊ | ✓₊ | (✓₊) | | | |
| YAWL | ✓→ | ✓→\|+ | ✓→\|+ | ✓₀ | ✓₀ | ✓₊ | — | | | ✓₊ | ✓₊ | (✓₊) | | | |
| CMMN | | | | | | | | — | | | | | | | |
| CLIMB | | | | | | | | | — | | | | | | |
| Declare | ✓→ | ✓₀ | ✓→\|+ | ✓₀ | ✓→\|+ | ✓₊ | ✓→\|+ | | | — | ✓₊ | ✓₊ | | | |
| MP-Declare | (✓$_{B}$→) | (✓₊) | (✓₊) | (✓₊) | (✓₊) | (✓₊) | (✓₊) | | | ✓₊ | — | (✓₊) | | | |
| DPIL | ✓$_{B}$→ | (✓₊) | (✓₊) | ✓₊ | (✓₊) | ✓₊ | (✓₊) | | | ✓₊ | (✓₊) | — | | | |
| TLA+ | | | | | | | | | | | | | — | | |
| DCR | | | | | | | | | | | | | | — | |
| EM-BrA²CE | | | | | | | | | | | | | | | — |

Tabelle 8.2: Erweiterung der sprachlichen Abdeckung von Translationstechniken für konzeptuelle Prozessmodelle

*Gesamtabdeckung (nur direkt) auf 25% erhöht*

Aus Tabelle 8.2 kann abgeleitet werden, dass sich durch die Verfügbarkeit des induktiven Translationsprinzips die Gesamtabdeckung der Translationstechniken für die möglichen Sprachpaarungen auf etwa 25% erhöht, wobei lediglich konservative, direkte sowie induktive, direkte Translationsansätze berücksichtigt werden. Letztere setzen sich zudem nur aus solchen Kombinationen aus Spurgenerator und Process-Mining-Werkzeug zusammen, die hinsichtlich des jeweiligen Sprachpaars keinen Verlust an Prozessperspektiven oder Ereignistypen verursachen. Zusammen mit allen bereits vorher ermöglichten und allen neu hinzugekommenen transitiven Translationsmöglichkeiten sowie allen Möglichkeiten für beschränkte Übersetzungen erhöht sich der Gesamtanteil auf 43%. Im Bereich der imperativ-imperativen Translationen wird dadurch sogar eine vollständige Abdeckung aller möglichen Paarungen der angegebenen Sprachen erzielt. Zudem ist erkennbar, dass nun auch Translationen mit Beteiligung einiger deklarativer Prozessmodellierungssprachen durchgeführt werden können. Bestand vorher nur die Möglichkeit, Modelle aus der deklarativen Sprache Declare direkt entweder in BPMN oder Petri-Netze zu übersetzen, erweitert sich die Menge möglicher Zielsprachen auf die vollständige Menge der angegebenen imperativen Sprachen. Gleiches gilt auch für die deklarativen Sprachen MP-Declare und DPIL, für die vorher keinerlei Translationsmöglichkeiten bestanden. Aber auch die Abbildung imperativer Modelle auf deklarative Repräsentationsformen ist nun unterstützt. Mittels des induktiven Prinzips lässt sich so für imperativ-deklarative Translationen eine Abdeckung von rund 14% erzielen. Dabei sind nur direkte Übersetzungen ohne werkzeugbedingtem Informationsverlust berücksichtigt. Lässt man diese Einschränkungen außer Acht, erreicht man eine Abdeckung von etwa 37,5%. Für die letzte Gruppe – deklarativ-deklarative Sprachpaarungen – ist die Abdeckung mit etwa 10–11% am geringsten. Allerdings ist ohne das im Rahmen der vorliegenden Arbeit vorgeschlagene induktive Prinzip auch in diesem Bereich keine Translation möglich.

*Gesamtabdeckung (direkte und transitiv) auf 43% erhöht*

*Gesamtabdeckung nach Paradigmen*

*Keine Messung des Laufzeitverhaltens sinnvoll*

Im Gegensatz zur Analyse der Eigenschaften des im Rahmen der vorliegenden Arbeit entwickelten Spurgenerators wird für die ebenfalls in diesem Kontext entstandene Translationstechnik *keine* Laufzeitanalyse durchgeführt. Das ist damit zu begründen, dass diese Translationstechnik direkt vom Laufzeitverhalten und der Leistungsfähigkeit des jeweiligen Spurgenerators und der gewählten Process-Mining-Technik abhängig ist. Einerseits stehen für den Großteil dieser Techniken

bereits entsprechende Untersuchungsergebnisse zur Verfügung und andererseits würde es den Rahmen der Arbeit sprengen, das Laufzeitverhalten der Kombination aus Spurgenerator und Process-Mining-Werkzeug für *jede* beliebige Sprachpaarung zu messen. Ob eine solche Analyse der Gesamtlaufzeit überhaupt notwendig ist, entscheidet sich vor allem danach, ob die Translation *echtzeitkritisch* ist, also ohne Verzögerung darauf aufbauender Tätigkeiten erfolgen muss. Da sich die vorliegende Arbeit aber mit dem grundlegenden *Prinzip* einer induktiven Translation beschäftigt, wird eine Laufzeitanalyse für konkrete Technologiepaarungen als irrelevant betrachtet.

Ziel dieses Abschnitts ist es, zu evaluieren, inwieweit die Einführung eines induktiven Translationsprinzips für Prozessmodelle die Abdeckung möglicher Sprachpaarungen verbessert. Eine wesentliche Erkenntnis ist, dass für Übersetzungen mit je einer imperativen Prozessmodellierungssprache als Quelle und als Ziel nahezu eine vollständige Abdeckung erreicht werden kann. Dabei ist allerdings zu berücksichtigen, dass ein Teil dieser Übersetzungen durch einen transitiven Abschluss erreicht wird. Die somit an der Übersetzung zusätzlich beteiligten Sprachen können einen Informationsverlust verursachen, wenn sie weniger ausdrucksstark sind als Quell- und Zielsprache. Weiterhin unterstützt nicht jeder Spurgenerator und nicht jedes Process-Mining-Werkzeug alle Prozessperspektiven, sodass auch dadurch Informationen verloren gehen können. Mit einer Steigerung der Abdeckung von 20% auf 43% unter Berücksichtigung aller betrachteten Sprachpaarungen, ist der Unterschied dennoch signifikant.

## 8.3 KORREKTHEIT: SPURGENERIERUNG FÜR MULTI-PERSPEKTIVISCHE, DEKLARATIVE PROZESSMODELLE

Die größte mögliche Fehlerquelle des in Abschnitt 4.2 vorgestellten multi-perspektivischen, deklarativen Spurgenerators ist die Abbildung des DPIL-Modells auf eine gegenüber Alloy konforme Repräsentation. Alle nachfolgenden Schritte von der Umformung der Alloy-Spezifikation in Boolesche Formeln bis zu deren Auflösung wurden bereits im Rahmen der Entwicklung des Alloy-Rahmenwerks geprüft. Aus diesem Grund wird die Korrektheit von Alloy als gegeben angesehen, sodass im aktuellen Abschnitt lediglich die Transformation von DPIL-Modellen in Alloy untersucht wird.

### 8.3.1 *Aspekte der Korrektheit*

Die Abbildung eines DPIL-Modells auf Alloy-Sprachkonstrukte ist eine spezielle Form der Modelltransformation. Um die Korrektheit von Modelltransformationen zu prüfen, müssen mehrere Aspekte betrachtet werden [39][4]:

- *Wohlgeformtheit* des Ergebnisses der Transformation betrifft die syntaktische Korrektheit, also die Konformität des Resultats bezüglich der Zielsprache der Transformation. *Wohlgeformtheit*

4 Aktuelle Methoden zur Prüfung der Korrektheit von Modelltransformationen beinhalten auch die Prüfung der Eigenschaften der Transformationssprache. Da in der vorliegenden Arbeit von dem OMG-Standard Mof2Text ausgegangen wird, kann dieser Teil jedoch übersprungen werden.

*Modell-Syntax-Relationen*

- *Modell-Syntax-Relationen* betreffen die korrekte Assoziation von Elementen des Quellmodells mit Elementen des Zielmodells.

*Modell-Semantik-Relationen*

- *Modell-Semantik-Relationen* betreffen die korrekte Abbildung der Quellmodell-Semantik auf eine gewünschte Zielmodell-Semantik. Für Korrektheit der Spurgenerierung ist hier vor allem die Relation der semantischen Äquivalenz interessant.

*Funktionales Verhalten*

- *Funktionales Verhalten* bezeichnet die Garantie der Erfüllung bestimmter Funktionen und betrifft eher Modell-zu-Modell- als die hier relevanten Modell-zu-Text-Transformationen. Für den aktuellen Kontext ist hier besonders die Erfüllbarkeit der Transformationsregeln entscheidend.

*Syntaktische Vollständigkeit*

- *Syntaktische Vollständigkeit* ist die Überprüfung, ob eine Transformationsspezifikation das vollständige Quell- respektive Zielmodell abdeckt.

*Wohlgeformtheit durch Alloy-Schablonen*

Die Wohlgeformtheit wird durch die Verwendung des Alloy-Rahmenwerks implizit sichergestellt. Die in Abschnitt 4.2 dargestellten Abbildungsregeln für DPIL-Entitäten und -Regeln auf eine gegenüber Alloy konforme Syntax sind direkte Abbildungen auf Alloy-Konstrukte. Letztere werden mittels des Alloy-Analysewerkzeugs syntaktisch geprüft. Der im Alloy-Rahmenwerk enthaltene Compiler stellt somit die Wohlgeformtheit der Ausdrücke sicher.

*Fokus: Korrekte Modell-Semantik-Relationen*

Da die Abbildung von DPIL auf Alloy-Sprachelemente in der vorliegenden Arbeit vor dem Hintergrund der Erzeugung eines Simulationsmodells erfolgt, sind Modell-Syntax-Relationen von untergeordneter Bedeutung. Jede beliebige syntaktisch wohlgeformte Konstruktion im Zielmodell, welche gegenüber der betroffenen Konstruktion des Quellmodells semantisch äquivalent ist, ist ein korrektes Transformationsergebnis. Dies ist damit zu begründen, dass für den Zweck der Spurgenerierung lediglich die erzeugten Protokolldaten korrekt sein müssen. Welche syntaktische Konstruktion im Modell dies sicherstellt ist für diesen Verwendungszweck unerheblich. Aus diesem Grund fokussiert sich der aktuelle Abschnitt auf die Prüfungen der Modell-Semantik-Relationen. Dennoch ist auch der rein auf die Syntax bezogene Aspekt der Korrektheit durch die Verwendung von Alloy-Schablonen und ihre Assoziation mit DPIL-Regelschablonen sichergestellt.

Eine Prüfung des funktionalen Verhaltens ist im Falle des zu prüfenden Spurgenerators zweitrangig. Die erwähnte Erfüllbarkeit von Transformationsregeln ist bei Modell-zu-Text-Transformationen in jedem Fall sichergestellt. Für die spätere (Abschnitt 8.4) Untersuchung der Korrektheit des vorgestellten Translationsprinzips hat dieser Aspekt eine größere Relevanz.

Von zentraler Bedeutung ist dagegen die syntaktische Vollständigkeit der Abbildung von DPIL auf Alloy. Diese ist im aktuellen Entwicklungsstadium nicht gegeben. Stattdessen ist der Grad der Unterstützung aus den Transformationsregeln in Abschnitt 4.2 direkt abzulesen. DPIL-Entitäten können mit dem vorgestellten Prinzip in Alloy vollständig dargestellt werden. Anders verhält es sich mit dem Prinzip der DPIL-Regel-Transformationen. Diese basieren auf dem Erkennen von DPIL-Regel-Mustern und der direkten Zuordnung einer entsprechenden Alloy-Schablone. DPIL unterstützt jedoch Boolesche Verknüpfungen, sodass theoretisch beliebig viele DPIL-Regeltypen formuliert werden können. Mit einer direkten

*Syntaktische Unvollständigkeit*

Transformation über statische Muster kann somit keine syntaktische Vollständigkeit erreicht werden. Alloy eignet sich als Simulationssprache jedoch unter anderem deswegen, da es, wie DPIL selbst, zu einem Teil auf der Prädikatenlogik erster Stufe basiert. Folglich ist für zukünftige Weiterentwicklungen eine Transformation auf Basis atomarer Sprachelemente zu prüfen, was dann die Transformation auf Basis von komplexen Sprachmustern ablösen würde.

Da sich die genannten Aspekte der Korrektheit allgemein auf Modelltransformationen beziehen, haben sie prinzipiell auch für den in dieser Arbeit betrachteten Translationsansatz Gültigkeit. Hierbei bergen die Modell-Syntax-Relationen eine große Schwierigkeit, da hierfür unter anderem die syntaktisch korrekte Abbildung zwischen Konstrukten imperativer und deklarativer Prozessmodellierungssprachen bekannt sein muss. Dies ist jedoch allgemein nicht der Fall. In Abschnitt 8.4 wird auf die übrigen Aspekte detaillierter eingegangen.

### 8.3.2 *Versuchsaufbau, -durchführung und Ergebnisse*

Im aktuellen Abschnitt wird beschrieben, wie die semantische Äquivalenz eines DPIL-Konstrukts und des zugeordneten Alloy-Konstrukts sichergestellt wird. Unter semantischer Äquivalenz wird hier die Gleichheit des Ausführungsverhaltens von DPIL-Modell und Alloy-Spezifikation verstanden. Nachfolgend wird ein Versuchsaufbau erläutert, der diese Äquivalenz der Ausführungssemantik sicherstellt. Die Durchführung der Versuche zeigt eine vollständige ausführungssemantische Äquivalenz aller DPIL-Regeln, für die eine Alloy-Entsprechung formuliert ist.

Um die Korrektheit einer Abbildung zu prüfen, existieren mehrere Verfahren, beispielsweise logische Inferenz, formale Modellprüfung, Modelltests und statische Analysen. Alloy selbst ist nicht nur eine Sprache, sondern auch ein Rahmenwerk für beispielsweise die Formulierung und Durchführung von Modelltests. Somit ist es möglich, Testspezifikationen für die semantische Äquivalenz eines DPIL-Modells und einer Alloy-Spezifikation in Alloy selbst zu formulieren. Das Alloy-Rahmenwerk bietet unter anderem Mittel für die Instanzsuche, die somit die Erfüllbarkeit der Testspezifikation prüfen können. Tests bergen dadurch jedoch die Gefahr einer unvollständigen Fallabdeckung. Wird der Lösungsraum für die Instanzsuche klein gewählt, kann es sein, dass die Tests zwar positive Ergebnisse liefern, obwohl sich in einem größeren Lösungsraum Fehler manifestieren würden. Für jede der in Abschnitt 4.2 angegebenen DPIL-Regeln kann jedoch abgeleitet werden, bei welcher Größe des Lösungsraums die jeweilige Regel wirksam wird. Aus diesem Grund wird für jede der in den Abbildungstabellen aufgeführten Abbildungsregeln eine Testspezifikation bereitgestellt. Das resultierende Testverfahren wird nachfolgend an Beispiel des DPIL-Makros `response(A,B)` erläutert.

*Testspezifikationen in Alloy*

Die Semantik der `response(A,B)`-Regel ist, dass, falls Aktivität `A` ausgeführt wird, vor Beendigung des Prozesses schließlich auch `B` ausgeführt werden muss. Im Umkehrschluss gilt, dass es keine Prozessinstanz geben kann, in der zwar `A` ausgeführt wird, *ohne* jedoch später `B` auszuführen. Genau diese gegenläufigen Formulierungen werden einerseits zur Formulierung der Regel selbst und andererseits zur Formulierung ihrer Testspezifikation verwendet, was in Codeausschnitt 8.1 dargestellt ist (`assert Test..`). Zusätzlich enthält die Testspezifikation Deklara-

*Beispielhafte Testspezifikation*

```
one sig A extends Task{}
one sig B extends Task{}

fact {
  response[A,B]
}

assert Test{
  no te: TaskEvent | A in te.assoEl and #searchAfter[te,B] = 0
}

check Test for 3 TaskEvent, 1 Identity, 0 DataObject, 0 Group,
  0 Relation, 0 RelationType, 5 Int
```

Codeausschnitt 8.1: Alloy-Testspezifikation für Abbildung des `response`-Makros

tionen von zwei Aktivitäten, die von der `response`-Regel betroffen sind. Die Ausführung des zugehörigen `check`-Kommandos prüft, ob für das angegebene `assert` Gegenbeispiele im Lösungsraum von drei Ereignissen einer Ausführungsspur ermitteln kann. Mögliche Gegenbeispiele in diesem Lösungsraum sind *A*, *AA* oder auch *ABA*. Da die Instanzsuche von Alloy im angegebenen Lösungsraum vollständig ist, gilt der Test als erfolgreich, wenn *kein* Gegenbeispiel gefunden werden kann. Eine Ausführung der oben dargestellten Testspezifikation belegt, dass dies für den gewählten Lösungsraum der Fall ist. Gleiches gilt für die Resultate der Testspezifikationen[5] für alle übrigen Abbildungsregeln. Folglich ist die Korrektheit der Abbildung eines DPIL-Modells auf eine zu Alloy konforme Textspezifikation im Rahmen der gewählten Größe des Lösungsraums sichergestellt.

Ziel dieses Abschnittes ist es, die Korrektheit des multi-perspektivischen, deklarativen Ansatzes zur Spurgenerierung zu prüfen. Da die Hauptfehlerquelle in der Abbildung von DPIL-Prozessregeln auf Alloy-Konstrukte liegt, können mittels ebenfalls auf Alloy basierenden Testspezifikationen entsprechende Untersuchungen angestellt werden. Auf diese Weise sind alle derzeit unterstützten Abbildungsregeln von DPIL-Sprachelementen auf Alloy-Konstrukte getestet – mit durchgehend positiven Resultaten. Die Entwicklung der einzelnen Parametrierungen (Abschnitt 4.2.5 und Abschnitt 4.2.6) ist ebenfalls mittels Alloy-Tests geprüft. Damit sind sowohl die Erhaltung der Semantik des Eingabe-DPIL-Modells als auch die Korrektheit der Umsetzung der Laufzeitparameter des Spurgenerators sichergestellt.

*Positive Testresultate*

## 8.4 KORREKTHEIT: INDUKTIVE TRANSLATION VON PROZESSMODELLEN

Ein zentrales Qualitätskriterium bei der Übersetzung von Prozessmodellen ist die Korrektheit des erzeugten Prozessmodells in der gewünschten Zielsprache. Da das in Abschnitt 5.2 vorgestellte induktive Translationsprinzip für verschiedenste Sprachpaare verwendet werden kann, ist es nicht möglich, eine pauschale Aussage zu treffen, inwiefern das Prinzip im Allgemeinen korrekte Ergebnisse liefert. Anspruch der vorliegenden Arbeit ist es jedoch, das Prinzip exemplarisch zu testen

5 verfügbar unter: `http://mps.kppq.de`

und damit erste Erkenntnisse hinsichtlich des Potentials eines induktiven Translationsverfahres für Prozessmodelle zu gewinnen.

Da die Translation eine Spezialisierung der allgemeinen Modelltransformation darstellt, gelten auch für sie die allgemeinen Aspekte der Korrektheit (Abschnitt 8.3.1). Nachfolgend werden vordergründig Metriken beschrieben (Abschnitt 8.4.1), welche eine automatisierte Quantifizierung des Aspekts des korrekten funktionalen Verhaltens der Translationsergebnisse ermöglichen. Anschließend (Abschnitt 8.4.2) wird darauf aufbauend ein geeigneter Versuchsaufbau skizziert, mit welchem es möglich ist, die darauf folgenden Ergebnisse (Abschnitt 8.4.3) zu ermitteln. Mit diesem Versuchsaufbau wird gleichzeitig auch ein weiterer Aspekt – die Wohlgeformtheit der Ergebnisse – geprüft, da die Prüfung des funktionalen Verhaltens erneut durch Spurgeneratoren getestet wird. Ist ein durch Translation erzeugtes Prozessmodell nicht wohlgeformt, so scheitert der Spurgenerator bei dessen Interpretation. Für die Prüfung von Modell-Syntax- und Modell-Semantik-Relationen können keine allgemeingültigen Prüfkriterien ermittelt werden, da diese unmittelbar vom jeweiligen Sprachpaar abhängig sind. Der letzte Aspekt, die syntaktische Vollständigkeit, ist dagegen direkt vom verwendeten Process-Mining-Werkzeug abhängig. Beispielsweise ist es nicht möglich, mit dem Alpha-Algorithmus Inklusiv-Oder-Verzweigungen zu erzeugen, obwohl er für die Erzeugung von BPMN-Modellen eingesetzt wird. Somit ist *dieses* Translationssystem insgesamt syntaktisch unvollständig. Für die Analyse des in der vorliegenden Arbeit vorgestellten *allgemeinen* Translationsprinzips ist dieser Aspekt jedoch irrelevant.

*Fokus: Funktionales Verhalten*

*Prüfung der übrigen Aspekte*

### 8.4.1 *Metriken*

Das in Abschnitt 5.2 vorgestellte Konzept zur induktiven Translation von Prozessmodellen verarbeitet das Eingabemodell in zwei Stufen. Zunächst wird ein Spurgenerator verwendet, um künstliche, valide Ausführungsspuren des Prozesses zu erzeugen und diese in einem Ereignisprotokoll zusammenzufassen. Anschließend wird das Ereignisprotokoll von einer geeigneten Process-Mining-Technik erschlossen, welches daraufhin ein Modell in der gewünschten Zielsprache erzeugt. In beiden Fällen kommen in der Regel statistische Verfahren zum Einsatz und sind damit im Allgemeinen naturgemäß fehlerhaft[6]. Damit existieren prinzipiell zwei Fehlerquellen, welche für ein fehlerhaftes Übersetzungsergebnis verantwortlich sein können. Nachfolgend wird jedoch argumentiert, warum es nicht notwendig ist, *beide* Technologien der Werkzeugkette individuell zu evaluieren.

*Zwei Fehlerquellen*

Einerseits ist es möglich, dass der Spurgenerator qualitativ minderwertige Ereignisprotokolle erzeugt. Andererseits ist im Allgemeinen nicht auszuschließen, dass die gewählte Process-Mining-Technik aus einem Ereignisprotokoll fehlerhafte Prozessmodelle konstruiert. Eine von der gewählten Process-Mining-Technik unabhängige Bewertung des Spurgenerators ist jedoch nicht zielführend. Nimmt man beispielsweise die Vollständigkeit des Ereignisprotokolls, also die Manifestation aller möglichen Ausführungsspuren als Qualitätskriterium, so ist es mög-

6 Eine Ausnahme bildet hierbei der in Abschnitt 4.2 vorgestellte Spurgenerator, da dieser auf Wunsch deterministisch arbeitet.

lich, dass ein Spurgenerator durch wahrscheinlichkeitsbasierte Entscheidungen an Verzweigungen im Prozess nicht jeden möglichen Pfad im Protokoll erfasst. Dennoch ist es möglich, dass die gewählte Process-Mining-Technik ein vollständiges und korrektes Prozessmodell ermittelt. Grund dafür ist, dass für viele der Process-Mining-Verfahren keine Notwendigkeit besteht, dass alle vollständigen Ausführungspfade protokolliert werden, sondern nur jede mögliche paarweise Abfolge von Aktivitäten erfasst ist. Zudem steht die Entwicklung von Spurgeneratoren und Process-Mining-Techniken nicht im Fokus der vorliegenden Arbeit. Folglich wird auch deren Evaluation nicht als Aufgabe derselben verstanden. Stattdessen wird an dieser Stelle davon ausgegangen, dass die zu verwendenden Technologien der Translationsketten im Rahmen ihrer Entwicklung bereits evaluiert werden. Auf die in Tabelle 5.2 aufgelisteten Spurgeneratoren und die in Tabelle 5.3 festgehaltenen Process-Mining-Werkzeuge trifft dies zu. Aus diesem Grund ist das Ziel dieses Abschnitts, Metriken zu identifizieren, welche das *Gesamtergebnis* der Translationskette bewerten.

*Bewertung des Gesamtergebnisses*

Geht man davon aus, dass eine Translation genau dann korrekt ist, wenn Quell- und Zielmodell semantisch gleichwertig sind, dann bestünde die Evaluation aus entsprechenden Modellvergleichen. Ein formaler Vergleich von Prozessmodellen unterschiedlicher Notationen ist jedoch aufgrund des Mangels an geeigneten Verfahren als Quelle für entsprechende Metriken auszuschließen [23]. Bei der Evaluation von Process-Mining-Techniken wird dagegen geprüft, inwiefern das real beobachtete Verhalten mit dem von dem erschlossenen Prozessmodell vorgeschriebenen und unterstützten Verhalten übereinstimmt. Dieses Vorgehen wird *Konformitätsprüfung* (engl. *conformance checking*) genannt. Besonders durch die zum Teil drastischen Unterschiede in den Ausdrucksmitteln der verschiedenen Sprachpaare, ist die automatisierte Prüfung der Verhaltensähnlichkeit naheliegend und ist auch der für die vorliegende Arbeit verwendete Weg zur exemplarischen Prüfung der Korrektheit der induktiven Translation.

*Konformitätsprüfung*

Die Konformität eines Prozessmodells gegenüber des in der Realität beobachteten Verhaltens wird in Bezug auf Process-Mining-Techniken durch einen Abgleich zwischen ermitteltem Prozessmodell und zur Verfügung stehenden Ereignisprotokollen geprüft. Dabei sind zwei grundlegende Aspekte der Konformität interessant. Einer ist die Überprüfung, ob das Prozessmodell *flexibel* genug ist und das real beobachtete Verhalten *gestattet*. Der zweite Aspekt ist die Überprüfung, ob das Prozessmodell *genau* genug ist, um das in der Realität nicht beobachtete und unerwünschte Verhalten zu *verbieten*. Zur Quantifizierung dieser beiden Kriterien sind im Bereich des Process Mining zwei Metriken eingeführt worden, die als *Fitness* und *Appropriateness* bezeichnet werden.

*Flexibilität vs. Genauigkeit*

*Fitness und Appropriateness*

Je nachdem, ob ein Prozess auf Basis eines gerichteten Graphen oder auf einer Menge von Regeln abgebildet wird, werden die genannten Metriken unterschiedlich definiert. Für die vorliegende Arbeit wird für die Konformitätsprüfung des Kontrollflusses von folgender Fitness-Metrik ausgegangen [7]:

*Fitness*

$$\mathbf{fit}(L, M) = 1 - \frac{\mathbf{fcost}(L, M)}{\mathbf{move}_L(L) + |L| \times \mathbf{move}_M(M)} \tag{29}$$

Die Formel geht von der Annahme aus, dass die Fitness eines Modells maximal ($\mathbf{fit}(L, M) = 1$) ist, wenn alle Ausführungsspuren eines Ereignisproto-

kolls im Modell vollständig nachvollzogen werden können (engl. auch *replay fitness*). Kann keine der Ausführungsspuren im Modell nachvollzogen werden, ist der Wert entsprechend minimal ($\mathfrak{fit}(L, M) = 1$). Das „Nachvollziehen" erfolgt über ein sogenanntes *Alignment*, welches durch simultane Bewegungen ($\mathfrak{move}$) durch Ausführungsspur und Modell gebildet wird. Betrachtet man die beispielhafte Ausführungsspur $\sigma_L = abdeg$ eines Ereignisprotokolls $L$ und eine mögliche Ausführungssequenz eines Modells $\sigma_M = acdeh$, dann kann die Ausführungsspur bist zum Ereignis vor der Ausführung von Aktivität $b$ des Ereignisprotokolls im Modell nachvollzogen werden. Die simultane Bewegung in Modell und Ereignisprotokoll ist bis dahin also gegeben. Folglich existieren zwei Arten von Fehlern bei einem Alignment: *(i)* Bewegungen, welche nur im Ereignisprotokoll möglich sind ($\delta(x, \perp)$) und *(ii)* solche, die nur im Modell möglich sind ($\delta(\perp, y)$). Im ungünstigsten Fall sind alle Bewegungen entweder nur im Ereignisprotokoll ($\mathfrak{move}_L(L)$) oder nur im Modell ($\mathfrak{move}_M(M)$) möglich. Diese Terme dienen der Normalisierung des Fitnesswertes, also der Abstraktion vom Umfang des Modells und des Ereignisprotokolls. Der tatsächliche Fehlerwert wird mittels einer Kostenfunktion ($\mathfrak{fcost}(L, M)$) angegeben. Diese ist wie folgt definiert:

*Kostenfunktion*

$$\mathfrak{fcost}(L, M) = \sum_{\sigma_L \in L} \delta(\lambda_M(\sigma_L)) \tag{30}$$

Ausgehend von einem optimalen Alignment $\lambda_M$ berechnen sich die Kosten aus der Summe der Distanzen zwischen $\delta_M$ und dem tatsächlichen Alignment aller Prozessausführungsspuren. Das Ergebnis ist demnach das Verhältnis der tatsächlichen Kosten, ein bestimmtes Ereignisprotokoll so zu verändern, dass es mit dem optimalen Alignment mit einem spezifischen Modell übereinstimmt und den maximalen Kosten.

In [7] wird auch eine Berechnungsvorschrift für die Metrik Appropriateness angegeben. Für ihre Berechnung werden jedoch *Negativbeispiele*, also Beispiele für *nicht-valide* Prozessausführungsspuren benötigt ([6, S. 188f.]). Wie Tabelle 4.1 zeigt, ist nur etwa die Hälfte der verfügbaren Spurgeneratoren in der Lage, Negativbeispiele zu generieren. Geht man von der einschlägigen Literatur aus (z.B. [6]), dann enthalten real-historische Ereignisprotokolle Negativbeispiele nur in Form von Rauschen, welches jedoch nicht kontrollierbar ist. Aus diesem Grund wird von einer Verwendung dieser Metrik und damit auch von der *expliziten* Generierung von Negativbeispielen abgesehen. Stattdessen wird das in Abbildung 8.3 dargestellte Prinzip der *Reverse Fitness* angewendet.

*Problematik: Appropriateness erfordert Negativbeispiele*

Bei diesem Prinzip wird ein Prozessmodell mittels Spurgenerierung und Process Mining in die Zielsprache übersetzt. Steht für die Zielsprache ein Spurgenerator zur Verfügung, was für die in der nachfolgenden Konformitätsprüfung exemplarisch verwendeten Prozessmodellierungssprachen der Fall ist, dann wird dieser verwendet, um für das Zielmodell ebenfalls ein künstliches Ereignisprotokoll zu erzeugen. Ist das Zielmodell nicht exakt genug am Ereignisprotokoll des Quellmodells ausgerichtet, dann werden so auch Prozessausführungsspuren generiert, die gegenüber des Quellmodells nicht valide sind. Folglich ist die im Rahmen der vorliegenden Arbeit verwendete Appropriateness-Metrik wie folgt definiert:

*Reverse Fitness*

$$\mathfrak{app}(L, M) = \mathfrak{fit}(L_{M_t}, M_s) \tag{31}$$

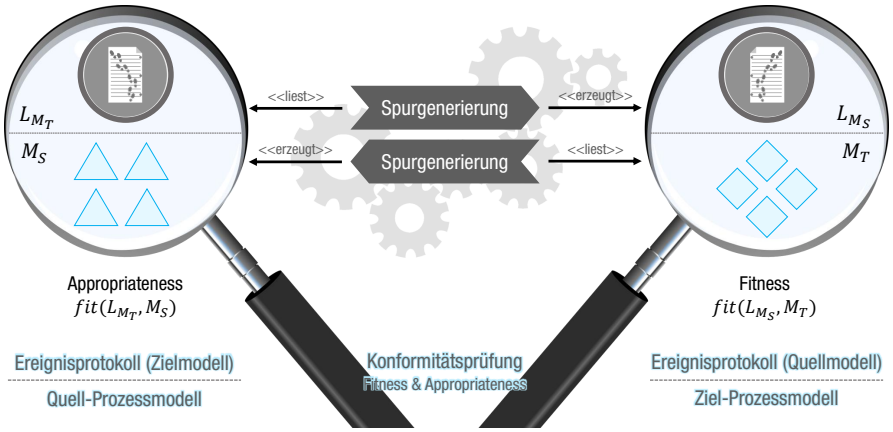



Abbildung 8.3: Konformität: Fitness und Appropriateness

Dabei wird im Gegensatz zur regulären Berechnung der Fitness-Metriken das Alignment des künstlichen Zielmodell-Ereignisprotokolls ($\mathrm{L}_{\mathrm{M_t}}$) und dem Quellmodell ($\mathrm{M_s}$) bewertet. Eine präzisere Parametrierung der oben eingeführten Fitness-Berechnungsvorschrift lautet demnach $\mathrm{fit}(\mathrm{L}_{\mathrm{M_s}}, \mathrm{M_t})$.

Die bisher betrachteten Metriken zur Bewertung der Konformität zwischen Ereignisprotokoll und Prozessmodell berücksichtigen lediglich den Kontrollfluss, da die Kostenfunktion $\mathrm{fcost}\,(\mathrm{L}, \mathrm{M})$ lediglich von den Unterschieden der Aktivitätsketten des idealen und des tatsächlichen Alignments abhängig ist. Für die Berücksichtigung weiterer Perspektiven bieten sich hier zwei Möglichkeiten: *(a)* Es wird eine neue Kostenfunktion eingeführt oder *(b)* die alte Kostenfunktion wird beibehalten und für jede weitere Prozessperspektive wird eine eigene Qualitätsmetrik definiert. Für die vorliegende Arbeit ist die Entscheidung auf Variante *(b)* gefallen, da Variante *(a)* einen vollständigen Verlust der verfügbaren Werkzeugunterstützung bedeuten würde. Wie die Messungen im späteren Verlauf dieses Abschnitts zeigen, besteht dafür keine Notwendigkeit.

*Eigene Qualitätsmetrik für jede Perspektive*

Da in den im späteren Verlauf beschriebenen Versuchen, zusätzlich zur Funktionalen und der Verhaltensorientierten Perspektive, lediglich die Organisatorische Perspektive in Form von Aufgaben-Rollen-Assoziationen betrachtet wird, erfolgt die Bewertung letzterer getrennt von den Bewertungen des Kontrollflusses. Für die Korrektheit $\mathrm{Op}$ der Assoziation von Aktivitäten mit Rollen wird folgende Metrik verwendet:

*Korrektheit aus der organisatorischen Perspektive*

$$\mathrm{Op} = \frac{\sum_{\sigma_L \in L} \frac{r_c \text{ in } \sigma_L}{r_a \text{ in } \sigma_L}}{|L|} \tag{32}$$

Dabei ist $\sigma_L$ eine Ausführungsspur aus dem Ereignisprotokoll $\mathrm{L}$. Mit $\mathrm{r_c}$ wird die Anzahl *korrekter* und mit $\mathrm{r_a}$ die Anzahl *aller* Rolle-Aktivität-Assoziationen innerhalb von $\sigma_L$ bezeichnet. Die Werte für $\mathrm{Op}$ sind ebenfalls auf den Wertebereich von 0 bis 1 normiert.

Alle im aktuellen Abschnitt beschriebenen Metriken für die Konformitätsprüfung zwischen Prozessmodellen und Ereignisprotokollen werden nachfolgend benötigt, um die Qualität der Translationsresultate zu bewerten.

### 8.4.2 *Versuchsaufbau*

In diesem Abschnitt wird der Versuchsaufbau zur Prüfung der korrekten Funktionsweise des Translationsansatzes aus Kapitel 5 beschrieben. Das schließt die Wahl der betrachteten Prozessmodellierungssprachen sowie die Beschreibung aller Beispiel-Modelle und des Vorgehensmodells zur Konformitätsprüfung ein.

*Facettenreiche Auswahl von Prozessmodellierungssprachen*

Da es im eingeschränkten Rahmen der vorliegenden Arbeit nicht möglich ist, die Korrektheit der Übersetzung für *alle* möglichen Sprachpaarungen zu untersuchen, wird zunächst eine Auswahl getroffen. Dabei wird berücksichtigt, dass einerseits imperative und deklarative sowie andererseits rein kontrollflussbasierte und multi-perspektivische Prozessmodellierungssprachen vertreten sind. So ist eine facettenreiche Abdeckung durch die sich anschließende Evaluation gewährleistet. Im Bereich der imperativen Prozessmodelle fiel die Entscheidung auf die BPMN, was mit der gegenüber Petri-Netzen kompakteren Darstellung zu begründen ist. Aus der Menge der deklarativen, kontrollflussorientierten Sprachen wird Declare ausgewählt, da diese die größte Verbreitung und Werkzeugunterstützung ihrer Klasse aufweist. Im multi-perspektivischen Sektor werden einerseits erneut die BPMN und andererseits DPIL ausgewählt. BPMN eignet sich vordergründig aufgrund seiner weiten Verbreitung und reichen Werkzeugunterstützung. Die Entscheidung für DPIL beruht auf der nachweislich hohen Ausdrucksmächtigkeit der Sprache. Somit ergeben sich für den Versuchsaufbau die folgenden beiden Paarungen: *(i)* BPMN, Declare und *(ii)* BPMN, DPIL. Für diese ersten Untersuchungen der Korrektheit induktiver Übersetzungen werden lediglich Modelle geprüft, für die eine bezüglich der Ausführungssemantik äquivalente Repräsentation in beiden Sprachen der jeweiligen Paarung ermittelt werden kann.

*BPMN, DPIL und Declare*

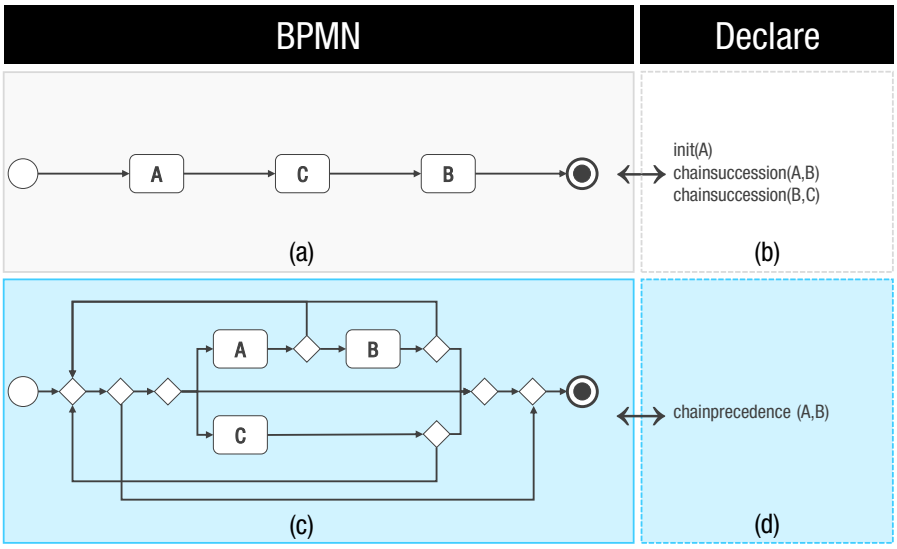



Abbildung 8.4: Beispielmodelle für Versuchsblock B1 mit Äquivalenzen (a)-(b) und (c)-(d)

Die Untersuchungen bezüglich dieser Sprachpaare sind in vier Blöcke aufgeteilt. In jedem wird die Korrektheit der Translation der als Beispiele verwendeten Prozessmodelle mit den vorab genannten Metriken Fitness und Appropriateness quantifiziert. Die Blöcke sind nach Sprachpaarung und Komplexität der verwende-

ten Beispielmodelle klassifiziert und betrachten in allen Fällen die *bidirektionale* Übersetzung der jeweiligen Sprachpaarung. Die einzelnen Versuchsblöcke setzen sich daher wie folgt zusammen:

*Vier Versuchsblöcke*

B1 *BPMN, Declare (künstlich, kompakt):* In diesem ersten Block der Untersuchungen werden für die BPMN und Declare je zwei künstliche, kompakte Prozesse mit klar abzugrenzenden Charakteristika modelliert (Abbildung 8.4). Dabei ist jedes Modell der einen Sprache ein valides Transformationsergebnis eines Modells der anderen Sprache. In der Abbildung sind jeweils die Modelle *(a)* und *(b)* respektive *(c)* und *(d)* verhaltensgleich. Dieser Untersuchungsblock dient vor allem dazu, *manuell nachvollziehbare* Ergebnisse zu produzieren.

B2 *BPMN, Declare (real, komplex):* Im zweiten Untersuchungsblock werden zwei inhaltlich voneinander unabhängige Prozessmodelle (Anhang A, Abbildung A.1 und Abbildung A.2) betrachtet, von denen jeweils eines als BPMN-Modell und eines als Declare-Modell repräsentiert wird. Ziel dieser Untersuchung ist es, zu prüfen, wie sich das Translationsprinzip bei realitätsbezogenen und damit meist komplexeren Prozessmodellen verhält.

B3 *BPMN, DPIL (künstlich, kompakt):* Der dritte Untersuchungsblock ist das multi-perspektivische Pendant zum ersten Block. Dafür werden die BPMN-Modelle des ersten Versuchsblocks um einige Rollendefinitionen erweitert, um die Organisatorische Perspektive zu berücksichtigen. Weiterhin wird für jedes dieser BPMN-Modelle ein ausführungssemantisch äquivalentes DPIL-Modell angefertigt (Abbildung 8.5).

B4 *BPMN, DPIL (real, komplex):* Im vierten und letzten Untersuchungsblock wird das Prozessmodell eines realen Prozesses aus B2 ebenfalls wiederverwendet und um organisatorische Rollen erweitert. Zusätzlich wird ein, im Sinne der Ausführungssemantik äquivalentes DPIL-Modell des Prozesses manuell erzeugt (Anhang A, Abbildung A.3).

Die hier vorgestellten Versuche wurden in der wissenschaftlichen Gemeinschaft diskutiert [13, 14]. Für die Versuchsblöcke B1 und B2 wird die BPMN verwendet, was scheinbar einen Widerspruch dazu darstellt, dass Declare lediglich die Funktionale und die Veraltensorientierte Perspektive unterstützt. Die BPMN diente hier jedoch lediglich der Kompaktheit der Darstellung, welche sich durch die Wahl geeigneter Beispiele auch ausschließlich auf die von Declare unterstützten Perspektiven fokussierte. Somit ist eine ausführungssemantisch vollständig korrekte Translation möglich.

*Versuchsblock B1*

Für Versuchsblock B1 werden vier einfache künstliche Prozessmodelle erzeugt, die sich – abgesehen von der jeweils verwendeten Sprache – bezüglich ihrer Komplexität unterscheiden. Wie Abbildung 8.4 zeigt, ist das imperative Modell *(a)* ein Extrem, da es die Bearbeitung der drei Aufgaben nur in einer fest vorgegebenen Sequenz gestattet. Das zugehörige, bezüglich der Ausführungssemantik äquivalente Declare-Modell *(b)* beinhaltet aufgrund dieser Stringenz drei Regeln. Dagegen enthält das Declare-Modell in *(d)* nur eine Regel. Das ausführungssemantisch äquivalente, imperative Modell *(c)* bietet entsprechend viele Freiheiten

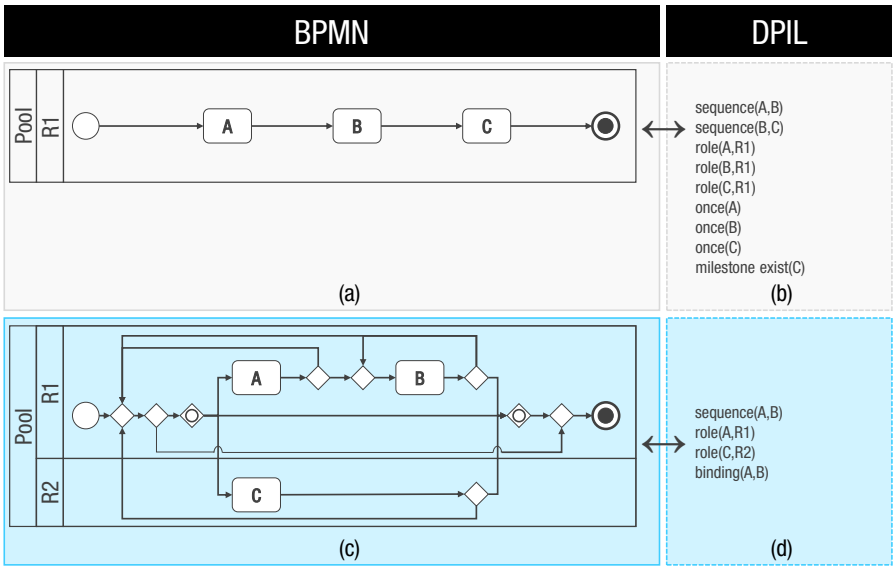



Abbildung 8.5: Beispielmodelle für Versuchsblock B3 mit Äquivalenzen (a)-(b) und (c)-(d)

und dadurch auch die meisten möglichen Ausführungspfade. Somit dient der erste Versuchsblock einerseits dazu, aufgrund der geringen Größe der Modelle manuell nachvollziehbare Ergebnisse zu produzieren und um andererseits in diesem kompakten Rahmen zwei Extreme gegenüberzustellen.

*Versuchsblock B3*

Mit dem dritten Versuchsblock (B3) werden ähnliche Ziele wie im zweiten verfolgt, wobei die exemplarischen Modelle um die Organisatorische Perspektive erweitert werden. Wie auch im ersten Block, stellen die beiden Modellpaarungen *(a)* und *(b)* respektive *(c)* und *(d)* in einem kompakten Rahmen Extreme bezüglich ihrer Komplexität dar. Während in Versuchsblock B1 im oberen Modellpaar die Gegenläufigkeit der Komplexität semantisch äquivalenter imperativer und deklarativer Prozessmodelle kaum zutage tritt, ist dieser Unterschied nach dem Hinzufügen der Organisatorischen Perspektive in Abbildung 8.5 greifbarer.

*Versuchsblock B2*

Für Versuchsblock B2 werden die Modelle zweier verschiedener Prozesse verwendet, die in Abbildung A.1 und Abbildung A.2 von Anhang A dargestellt sind. Ersteres beschreibt den Auslieferungsprozess eines Brotlieferanten und wird auch in [155] verwendet, um die Verbesserung der Verständlichkeit von Prozessmodellen in Kombination mit textuellen Arbeitsanweisungen zu prüfen. Die Verzweigungen des Prozesses gestatten knapp fünfzig verschiedene Ausführungspfade. In einem Teil des zweiten Versuchsblocks wird dieses Modell in die Sprache Declare übersetzt. Im zweiten Teil wird das Declare-Modell aus Abbildung A.2 in die entgegengesetzte Richtung übersetzt. Das Prozessmodell wurde aus den Ereignisprotokollen der *BPI Challenge 2014*[7] mit Hilfe des *Declare Maps Miner* [115] rekonstruiert. Zwar ist so nicht sichergestellt, dass das generierte Modell den Prozess adäquat abbildet – was für den Zweck der Untersuchungen im Rahmen dieser Arbeit jedoch unerheblich ist. Ziel eines Translationsansatzes ist es, ein beliebiges Quellmodell in die gewünschte Zielsprache zu übersetzen. Folglich ist lediglich die Korrekt-

7 http://www.win.tue.nl/bpi/doku.php?id=2014:challenge, zuletzt aufgerufen: 03.06.2017

heit dieser Abbildung, nicht jedoch die der Erfassung der realen Zusammenhänge entscheidend. Dasselbe Modell wird zudem auch bereits in [41] verwendet, um die Qualität der Ergebnisse eines Spurgenerators für Declare-Modelle zu bewerten. Die Protokolle wurden im Rahmen von Interaktionen von Kunden-Service-Desk-Mitarbeitern bezüglich Störungen in Telekommunikationssystemen erzeugt.

*Versuchsblock B4*

Im letzten Teil der Untersuchungen – Versuchsblock B4 – wird das Beispiel eines universitären Dienstreiseprozesses, welches im Laufe der Arbeit mehrfach Anwendung findet, erneut aufgegriffen. Zu diesem wird jeweils ein Modell in BPMN und eines in DPIL angefertigt, was in Abbildung A.3 von Anhang A dargestellt ist. Die Semantik der verwendeten DPIL-Regeltypen kann Codeausschnitt 3.7 bis Codeausschnitt 3.9 von Abschnitt 3.1.5 entnommen werden. Der Prozess ist an einigen Stellen vereinfacht worden, bildet jedoch die wesentlichen Aspekte und Perspektiven des Prozesses ab.

An den gewählten Beispielen ist zu sehen, dass keines der Modelle Informationen über die Datenorientierte oder die Operationale Perspektive beinhaltet. Das hat den Grund, dass für diese zum aktuellen Zeitpunkt keine Process-Mining-Techniken existieren. In diesen ersten Versuchen soll jedoch das prinzipielle Potential einer induktiven Übersetzung von Prozessmodellen geprüft werden. Die Berücksichtigung vom Translationsansatz *nicht unterstützter* Perspektiven würde die Testergebnisse verzerren und somit die Bewertung des Übersetzungserfolgs hinsichtlich der *unterstützten* Perspektiven erschweren. Aus diesem Grund werden die beiden genannten Perspektiven explizit von allen Versuchen ausgeschlossen.

An den einzelnen Versuchsblöcken sind mit der BPMN, Declare und DPIL drei Sprachen beteiligt. Folglich müssen nach dem in Abschnitt 5.2 vorgestellten Übersetzungsprinzip drei Spurgeneratoren sowie drei Process-Mining-Werkzeuge ausgewählt werden. Diese Auswahl ist in Abbildung 8.6 dargestellt. Für die Translation eines Declare-Modells in die BPMN wird folglich der *Synthetic Log Generator* [41] als Spurgenerator und der *BPMN Miner* [43] mit dem Algorithmus des *Flexible Heuristics Miner (FHM)* [191] eingesetzt. Da dieser Mining-Ansatz lediglich die Funktionale und die Verhaltensorientierte Perspektive betrachtet, wird die Erweiterung aus [37] verwendet, welche ein BPMN-Modell durch organisatorische Rollen ergänzt. Für die Gegenrichtung wird eine Kombination aus *PLG2*[8] [34] und Declare Maps Miner verwendet. Der PLG2 findet in Kombination mit dem *DPIL-Miner* [164] eine zweite Anwendung bei der Translation von BPMN-Modellen in die Sprache DPIL. Eine letzte Kombination, bestehend aus dem Spurgenerator *MuDePS* (Abschnitt 4.2) und erneut dem BPMN Miner ermöglicht die entgegengesetzte Translation.

*Auswahl der Komponenten des Translationssystems*

Im Falle der Spurgeneratoren für Declare und DPIL ist die Wahl damit begründet, dass zum aktuellen Zeitpunkt keine Alternativen existieren. Gleiches gilt für die Process-Mining-Technik DPILMiner. Für die Erschließung von Declare-Modellen aus Ereignisprotokollen existieren mehrere Techniken. Die Wahl fiel hier auf den Declare Maps Miner, da einerseits eigene Tests zu zufriedenstellenden Ergebnissen führten. Andererseits zeigen existierende Vergleiche, dass die Qualität der Mining-Resultate beispielsweise gegenüber des alternativen *MINERful*-Ansatzes

*Begründung der Wahl der konkreten Komponenten*

8 In [14] wird ein anderer Spurgenerator verwendet, der jedoch meist quantitativ und in jedem Fall qualitativ identische Ergebnisse lieferte.

überlegen ist [42]. Ein Grund dafür ist, dass MINERful sensibler auf die jeweilige Konfiguration reagiert, die wiederum vom konkreten Ereignisprotokoll abhängig ist. Die Wahl des Flexible Heuristics Miner ist schließlich mit dessen Robustheit zu begründen:

*Robustheit des Flexible Heuristics Miner*

- Er bewältigt Nachteile einfacherer Algorithmen, wie beispielsweise die Schwierigkeiten des Alpha Miners bei der Erkennung von kurzen Schleifen im Kontrollfluss
- Der FHM ist darauf spezialisiert, besonders komplexe Kontrollflussstrukturen abbilden zu können, was bei der Translation deklarativer Prozessmodelle entscheidend sein kann. Ein deklaratives Prozessmodell mit beispielsweise einer größeren Zahl an Prozessschritten, aber wenigen Regeln, lässt unter Umständen sehr komplexe Kontrollflüsse zu.
- Der Algorithmus ist fähig, die Probleme *schlecht-strukturierter Domänen* (engl. *low-structured domains*) zu bewältigen – was auch der Intention bei der Entwicklung der Sprache Declare entspricht.

Die Wahl der einzelnen Komponenten des induktiven Translationsprinzips kann nach Belieben verändert werden. Für die vorliegende Arbeit werden diese Festlegungen getroffen, um den Fokus der Evolution auf eine *grundlegende* Potentialeinschätzung zu legen. Die Ermittlung der besten Kombination aus Spurgenerator und Mining-Technik wäre eine wertvolle Information bezüglich des konkreten Sprachpaares, würde jedoch den Rahmen der Arbeit überschreiten und vom eben genannten eigentlichen Fokus ablenken.

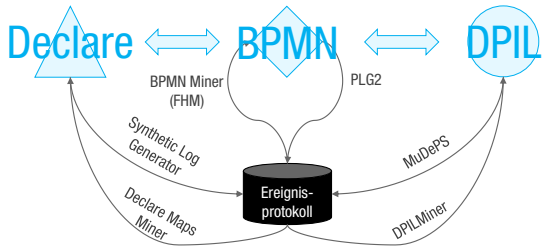



Abbildung 8.6: Wahl der Komponenten für die induktive Translation von Declare-, BPMN- und DPIL-Modellen

Nachdem die Auswahl der einzelnen Übersetzungkomponenten abgeschlossen ist, verbleiben noch deren Parametrierung sowie die Beschreibung des Versuchsaufbaus zu eigentliche Messung der Ergebnisqualität.

*Parametrierung*

Die größte Schwierigkeit bei der Erschließung von Prozessmodellen aus *realen, historischen* Ereignisprotokollen ist der Umgang mit Unvollständigkeit und Rauschen[9]. Diese beiden Probleme erfordern in der Regel eine individuelle Anpassung der Schwellwerte, die in der Regel zur Rauschbehandlung eingesetzt werden. Da die Translation jedoch auf Basis von *künstlichen* Ereignisprotokollen erfolgt, wird per Definition eine Datenbasis *ohne* Rauschen erzeugt und verarbeitet. Folglich ist

*Rauschfreiheit vereinfacht Konfiguration*

9 Aus Gründen der Einheitlichkeit wird an dieser Stelle erneut der Rauschbegriff verwendet. Dieser ist für das Verständnis vieler Parameter der beschriebenen Process-Mining-Technologien zentral.

| Parameter | Wert | Bedeutung |
|---|---|---|
| Flexible Heuristics Miner | | |
| Relative-to-best | 0.0 | Rauschparameter, Toleranzbereich für Abhängigkeit |
| Dependency | 49.0 | Rauschparameter, Schwellwert für Stärke einer Abhängigkeit |
| Length-one/two loop | 0.0 | Rauschparameter, Schwellwert für kurze Schleifen |
| Long distance | 100.0 | Rauschparameter, Schwellwert für transitive Abhängigkeiten |
| All tasks connected | true | Allgemeiner Parameter, Zusammenhang des Graphen ein-/ausschalten |
| Long distance dependenies | true | Allgemeiner Parameter, Ermittlung von Long distance dependencies ein-/ausschalten |
| Ignore loop dependency threshold | false | Allgemeiner Parameter, Schleifenschwelle beachten/missachten |
| Declare Maps Miner | | |
| Ignore Event Types | false | Allgemeiner Parameter, Beachtung des Aktivitätslebenszyklus |
| Minimum Support | 100.0 | Rauschparameter, Schwellwert für Relevanz/Irrelevanz einer Regel |
| Alpha | 0.0 | Rauschparameter, Schwellwert für Korrektheit einer Regel (0.0 bedeutet: keine Toleranz) |
| DPILMiner | | |
| Support | 5.0 | Rauschparameter, Schwellwert für Relevanz/Irrelevanz einer Regel |
| Confidence | 100.0 | Rauschparameter, Schwellwert für Korrektheit einer Regel (100.0 bedeutet: keine Toleranz) |

Tabelle 8.3: Statische Konfiguration der Process-Mining-Komponenten

einerseits eine individuelle Anpassung der Rauschparameter unnötig und andererseits können diese so restriktiv wie möglich gewählt werden. Die Vollständigkeit der künstlichen Protokolle liegt in der Verantwortung des Spurgenerators. Um den Evaluationsprozess zu vereinfachen ist deshalb in Tabelle 8.3 eine *statische* Konfiguration der drei Process-Mining-Techniken angegeben, welche für jeden Versuch verwendet wird.

*Konfiguration: Flexible Heuristics Miner*

Die *Dependency*-Schwelle des Flexible Heuristics Miner wird auf den Wert 49 gesetzt. Der Parameter bestimmt die Rauschempfindlichkeit des Algorithmus. Nachdem die künstlichen Ereignisprotokolle unverrauscht sind, ist die Annahme valide, dass ein Pfad, der lediglich einmal protokolliert wird, dennoch im Quellmodell erlaubt ist. Der Dependency-Wert für ein einmaliges Auftreten ist 50, sodass die Schwelle leicht darunter liegen muss. Eine einzelne feste Schwelle hat für die Verarbeitung historischer Ereignisprotokolle den Nachteil, dass bereits eine leichte Unterschreitung der Schwelle zum Verwerfen der Abhängigkeitsannahme zweier Aktivitäten führt. Der *Relative-to-best*-Schwellwert kompensiert diese Problematik durch die Etablierung eines dynamische Toleranzbereichs. Wurden bereits Abhängigkeiten akzeptiert und ist die Differenz der Stärke einer Abhängigkeit und der einer noch nicht akzeptierten geringer als dieser Schwellwert, so wird die bisher nicht akzeptierte Abhängigkeit doch noch als gültig angesehen. Der Parameter *All tasks connected* sorgt für einen zusammenhängenden Graphen, also für die Verbindung aller Aktivitäten in *einem* BPMN-Modell. Mittels des Schwellwertes für Abhängigkeiten der Kategorie *Long distance* wird der Wert auf 100 gesetzt. Einfachere Algorithmen, wie beispielsweise der Alpha Miner, scheitern daran, Abhängigkeiten zwischen nicht unmittelbar aufeinander folgenden Aktivitäten zu erkennen. Der FHM bietet hierfür eine eigene Schwelle an. Indem diese auf 100 gesetzt wird, wird festgelegt, dass diese Abhängigkeit in jeder Prozessausführungsspur gültig sein muss. Zudem müssen die beiden fraglichen Aktivitäten in der gleichen Häufigkeit auftreten. Mittels des Parameters *Long distance dependencies* wird die Funktion zur Erkennung dieser Kategorie von Abhängigkeiten aktiviert respektive deaktiviert. Schleifen der Länge eins und zwei[10] bereiten Process-Mining-Techniken häufig Schwierigkeiten, weswegen der

10 Beispiele: $< ...AA... >$ respektive $< ...ABA... >$

FHM hierfür zwei eigene Parameter anbietet. Werden diese auf 0 gesetzt, dann bedeutet das, dass im Zielmodell eine Schleife erzeugt wird, sobald sich in einer Prozessausführungsspur diese Schleife mindestens einmal manifestiert. Damit der Algorithmus diesen Parameter beachtet, muss *Ignore loop dependency thresholds* auf *false* gesetzt werden. Damit ist die Konfiguration des Flexible Heuristics Miners, der im BPMN Miner gekapselt ist, abgeschlossen.

*Konfiguration: Declare Maps Miner*

Weit weniger Konfigurationsmöglichkeiten weist der deklarative Process-Mining-Ansatz Declare Maps Miner auf. Mit der Festlegung, dass *Ignore Event Types* auf *false* zu setzen ist, wird das Mining-Werkzeug angewiesen, Ereignistypen zu berücksichtigen. Das ermöglicht die differenziertere Erkennung von Regeln zwischen beispielsweise dem Start und dem Ende von Aktivitätsdurchführungen, anstatt letztere als atomar anzusehen. Der Parameter *Minimum Support* beschreibt einen Schwellwert zur Rauschbehandlung. Dieser wird in Form des Prozentsatzes der Ausführungsspuren, in denen der jeweilige Regelkandidat erfüllt ist, angegeben. Aufgrund der Rauschfreiheit der künstlichen Ereignisprotokolle kann dieser Wert so restriktiv wie möglich gewählt werden. Mittels des Parameters *Alpha* werden Regelkandidaten verworfen, welche ausschließlich *trivial erfüllt sind*. Betrachtet man beispielsweise die Declare-Regel $\mathtt{response}(\mathrm{A}, \mathrm{B})$, welche aussagt, dass nach Ausführung von Aktivität $\mathrm{A}$ später auch Aktivität $\mathrm{B}$ ausgeführt werden muss, dann ist diese Regel trivial erfüllt, wenn Aktivität $\mathrm{A}$ niemals ausgeführt wird. Dies schließt die Konfiguration des Declare Maps Miner ab.

*Konfiguration: DPILMiner*

Noch weniger Parameter weißt der DPILMiner auf, welcher auf dem *Support-Confidence*-Rahmenwerk basiert. Es ähnelt der Parameterkombination Minimum Support und Alpha des Declare Maps Miners. Allerdings bezeichnet der Confidence-Parameter hier den Prozentsatz der Ausführungsspuren, in denen ein Regelkandidat erfüllt ist. Aufgrund der Rauschfreiheit der generierten Ereignisprotokolle ist dieser Wert maximal restriktiv gewählt worden. Der Wert für Minimum Support hat im Wesentlichen die Semantik des Alpha-Parameters des Declare Maps Miner. Allerdings handelt es sich bei ersterem um einen Schwellwert, welcher den Algorithmus Regelkandidaten mit einem Support von $\geqslant 5\%$ akzeptiert, während der Alpha-Wert lediglich Kandidaten mit einem Wert von $> \mathrm{x}$ akzeptiert. Damit ist auch die Konfiguration des DPILMiner abgeschlossen.

Die Simulationskomponenten tragen die Verantwortung für die Vollständigkeit der Datengrundlage. Teil des Experimentes ist es daher, herauszufinden, welche Anzahl von Prozessausführungsspuren welcher maximalen Länge zufriedenstellende Translationsresultate erzielen. Die jeweiligen Konfigurationen und die resultierende Ergebnisqualität sind im nachfolgenden Abschnitt erläutert.

Um die Ergebnisqualität zu messen, werden die im vorherigen Abschnitt beschriebenen Metriken auf jedes Translationsergebnis angewendet und in Abhängigkeit von der gewählten Parametrierung der Spurgeneratoren dargestellt.

### 8.4.3 *Ergebnisse*

Die Ergebnisse der einzelnen Versuchsblöcke (B1 bis B4) werden in Abhängigkeit der maximalen Anzahl ($N$) und Länge ($L$) der durch Spurgeneratoren erzeugten Prozessausführungsspuren dargestellt. Zur Bewertung dienen die in Abschnitt 8.4.1

definierten Metriken Fitness, Appropriateness und OP. Dabei wird zur Messung der Appropriateness die ebenfalls im referenzierten Abschnitt beschriebenen Methode der Reverse Fitness verwendet. Die Messungen zu jedem Versuch werden insgesamt zehnmal durchgeführt. Die Tabellen zeigen den daraus abgeleiteten Mittelwert. Zur Durchführung der Konformitätsprüfungen werden neu erzeugte Ereignisprotokolle verwendet, um einen möglichen Effekt der Überanpassung an die zur Übersetzung verwendeten Protokolle abzuschwächen. Der Umfang dieser neuen Testprotokolle variiert je nach Quellmodell und Quellsprache. In den Versuchsblöcken B3 und B4 werden die Spurgeneratoren so konfiguriert, dass jeder organisatorischen Rolle zwei Akteure zugeordnet werden. Dies ist notwendig, um unterscheiden zu können, ob eine Aktivität an eine bestimmte Rolle oder einen bestimmten Akteur gebunden ist.

*Ergebnisse: Versuchsblock B1*

Tabelle 8.4 zeigt eine Übersicht über die Ergebnisse der Konformitätsprüfung aus Versuchsblock B1, welche die Übersetzung jedes der vier Prozessmodelle aus Abbildung 8.4 von der Sprache BPMN in die Sprache Declare oder umgekehrt beinhaltet. Mit *(a→De)* ist folglich die Übersetzung des Modells *(a)* in ein semantisch äquivalents Modell der Sprache Declare (De) bezeichnet. Eine Möglichkeit hierfür ist Modell *(b)* derselben Abbildung. Es existieren jedoch potentiell mehrere Lösungen. Der Umfang des für die Konformitätsprüfung erzeugten Ereignisprotokolls beträgt hierbei 10000 Spuren einer maximalen Länge von 12.

| N | L | Fitness | | | | Appropriateness | | | |
|---|---|---|---|---|---|---|---|---|---|
| | | (a→De) | (b→B) | (c→De) | (d→B) | (a→De) | (b→B) | (c→De) | (d→B) |
| 10 | 3 | 1.0 | 1.0 | 0.7110 | 0.4932 | 1.0 | 1.0 | 0.9917 | 1.0 |
| 100 | 3 | 1.0 | 1.0 | 0.8911 | 0.6295 | 1.0 | 1.0 | 1.0 | 1.0 |
| 1000 | 3 | 1.0 | 1.0 | 1.0 | 0.7286 | 1.0 | 1.0 | 1.0 | 1.0 |
| 10000 | 3 | 1.0 | 1.0 | 1.0 | 0.7286 | 1.0 | 1.0 | 1.0 | 1.0 |
| 10 | 6 | 1.0 | 1.0 | 0.713 | 0.6111 | 1.0 | 1.0 | 0.9929 | 1.0 |
| 100 | 6 | 1.0 | 1.0 | 0.9874 | 0.7257 | 1.0 | 1.0 | 1.0 | 1.0 |
| 1000 | 6 | 1.0 | 1.0 | 1.0 | 0.9975 | 1.0 | 1.0 | 1.0 | 1.0 |
| 10000 | 6 | 1.0 | 1.0 | 1.0 | 1.0 | 1.0 | 1.0 | 1.0 | 1.0 |
| 10 | 9 | 1.0 | 1.0 | 0.713 | 0.6420 | 1.0 | 1.0 | 0.9929 | 1.0 |
| 100 | 9 | 1.0 | 1.0 | 1.0 | 0.7844 | 1.0 | 1.0 | 1.0 | 1.0 |
| 1000 | 9 | 1.0 | 1.0 | 1.0 | 1.0 | 1.0 | 1.0 | 1.0 | 1.0 |
| 10000 | 9 | 1.0 | 1.0 | 1.0 | 1.0 | 1.0 | 1.0 | 1.0 | 1.0 |

Tabelle 8.4: Konformität: Versuchsblock B1 (BPMN ↔ Declare – künstlich, kompakt)

*Erfolgreicher Test mit künstlichen, kompakten Modellen*

Auch wenn die Beispiele aus Abbildung 8.4 die Übersetzungsproblematik simplifizieren, können aus den Ergebnissen dennoch einige erste Erkenntnisse abgeleitet werden. Die Übersetzung der beiden einfachen Modelle *(a)* und *(b)* gelingt bereits mit einer sehr geringen Anzahl an Prozessausführungsspuren einer ebenfalls geringen Maximallänge. Da die beiden Modelle nur genau einen, dreischrittigen Ausführungspfad durch das Modell gestattet, war dieses Ergebnis zu erwarten. Eine erste wesentliche Erkenntnis ist, dass ein idealer Wert für die Appropriateness bereits mit sehr wenigen Prozessausführungsspuren erzielt werden kann. Die Erklärung hierfür ist, dass die meisten Process-Mining-Techniken – und so auch

die hier verwendeten – die zur Verfügung gestellten Prozessausführungsspuren als das einzig erlaubte Prozessverhalten interpretieren. Folglich führen fehlende Ausführungsspuren *nicht* zu einer Verschlechterung der Appropriateness und es kann bereits mit einer einzigen Ausführungsspur ein perfekter Wert erzielt werden. Umgekehrt verhält es sich folglich in Bezug auf die Fitness-Metrik. Damit ein Prozessmodell alle gewünschten Prozessausführungspfade unterstützt, müssen alle dafür notwendigen Übergangsmöglichkeiten von einer Aktivität zur nächsten im Protokoll enthalten sein. Aus diesem Grund entspricht es ebenfalls den Erwartungen, dass für die Modelle *(c)* und *(d)* eines äußerst flexiblen Prozesses eine größere Zahl an Prozessausführungsspuren benötigt wird. Beispielsweise sorgen die enthaltenen Schleifen dafür, dass die Anzahl der verschiedenen Ausführungspfade theoretisch unendlich ist. Betrachtet man die in der Tabelle dargestellten Ergebnisse, so zeigt sich, dass eine ausführungssemantisch vollständig korrekte Übersetzung auch dann zu erzielen ist, wenn die Menge der verschiedenen möglichen Prozessausführungsspuren nicht vollständig ist. Eine weitere Erkenntnis ist, dass, trotz der in den Modellen *(c)* und *(d)* enthaltenen Schleifen eine maximale Spurlänge von 3 bereits ausreicht, um eine korrekte Übersetzung zu erzielen. Betrachtet man das imperative der beiden Modelle, dann lässt sich ableiten, dass es möglich ist, mit den Ausführungsspuren *AA*, *AB*, *AC*, *CA* und *CC* sowie den Dreierketten *ABA* und *ABC* alle paarweisen Aktivitätsübergänge zu beschreiben. Für viele fortgeschrittenere Process-Mining-Techniken genügt dies, um entsprechende Kontrollflussverzweigungen im Zielmodell zu rekonstruieren. Somit ist möglich, das imperative Prozessmodell trotz enthaltener Schleifen mit einer maximalen Spurlänge von 3 in ein semantisch äquivalentes Prozessmodell zu überführen.

| N | L | Fitness | App. |
|---|---|---|---|
| 100 | 24 | 0.6371 | 1.0 |
| 1000 | 24 | 0.8181 | 1.0 |
| 10000 | 24 | 0.9992 | 1.0 |
| 100000 | 24 | 1.0 | 1.0 |
| 100 | 36 | 0.6554 | 1.0 |
| 1000 | 36 | 0.8988 | 1.0 |
| 10000 | 36 | 0.9998 | 1.0 |
| 100000 | 36 | 1.0 | 1.0 |

| N | L | Fitness | App. |
|---|---|---|---|
| 10 | 15 | 0.5 | 1.0 |
| 100 | 15 | 0.6253 | 1.0 |
| 1000 | 15 | 0.7462 | 1.0 |
| 10000 | 15 | 0.8784 | 1.0 |
| 100000 | 36 | 0.9335 | 1.0 |

Tabelle 8.5: Konformität: Versuchsblock B2 (l.: BPMN → Declare, r.: Declare → BPMN)

*Ergebnisse: Versuchsblock B2*

In Versuchsblock B2 wird das induktive Translationsprinzip auf zwei Modelle für reale Prozesse angewendet (Abbildung A.1 und Abbildung A.2 in Anhang A). Tabelle 8.5 zeigt, dass die Übersetzungen beider Modelle weitestgehend korrekt ist. Aufgrund der größeren Menge möglicher Ausführungspfade ist auch eine größere Menge an Prozessausführungsspuren nötig, um die Fitness des erzeugten Prozessmodells zu optimieren. Zudem bestätigen die Untersuchungen die Erkenntnisse des vorangegangenen Versuchsblocks in Bezug auf die Entwicklung der beiden Metriken. Während die Appropriateness in allen Fällen – und unabhängig von Spuranzahl und -länge – ideal ist, wächst die Fitness mit der steigenden

*Erfolgreicher Test mit realen, komplexen Modellen*

Anzahl zur Verfügung stehender Ausführungsspuren. Im Falle der Übersetzung des Declare-Modells in die BPMN wird mit der gewählten maximalen Anzahl an Prozessausführungsspuren kein ideales Ergebnis erzielt. Zur Überprüfung wird hier je ein neu generiertes Ereignisprotokoll mit 200000 Spuren einer maximalen Länge von 48 verwendet. Die Belegungen für Spurlänge sind hier nicht willkürlich. Die kürzestmögliche Ausführungsspur hat in Abbildung A.1 eine Länge von 12. Dagegen setzt sich das Prozessmodell in Abbildung A.2 aus insgesamt 12 Aktivitäten zusammen. Aus diesem Grund wird die Abhängigkeit von der Spurlänge jeweils mit einem ganzzahligen Vielfachen von 12 analysiert.

| N | L | Fitness | | | | Appropriateness | | | | OP Precision | | | |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
| | | (a→Dp) | (b→B) | (c→Dp) | (d→B) | (a→Dp) | (b→B) | (c→Dp) | (d→B) | (a→Dp) | (b→B) | (c→Dp) | (d→B) |
| 10 | 3 | 1.0 | 1.0 | 0.773 | 0.625 | 1.0 | 1.0 | 0.530 | 0.666 | 1.0 | 1.0 | 1.0 | 1.0 |
| 100 | 3 | 1.0 | 1.0 | 0.818 | 0.655 | 1.0 | 1.0 | 0.616 | 0.685 | 1.0 | 1.0 | 1.0 | 1.0 |
| 1000 | 3 | 1.0 | 1.0 | 0.884 | 0.777 | 1.0 | 1.0 | 0.788 | 0.757 | 1.0 | 1.0 | 1.0 | 1.0 |
| 10 | 6 | 1.0 | 1.0 | 0.802 | 0.749 | 1.0 | 1.0 | 0.669 | 0.940 | 1.0 | 1.0 | 1.0 | 1.0 |
| 100 | 6 | 1.0 | 1.0 | 0.931 | 0.876 | 1.0 | 1.0 | 0.868 | 0.999 | 1.0 | 1.0 | 1.0 | 1.0 |
| 1000 | 6 | 1.0 | 1.0 | 1.0 | 1.0 | 1.0 | 1.0 | 1.0 | 1.0 | 1.0 | 1.0 | 1.0 | 1.0 |

Tabelle 8.6: Konformität: Versuchsblock B3 (BPMN ↔ DPIL – künstlich, kompakt)

*Ergebnisse: Versuchsblock B3*

In Versuchsblock B3 wird zusätzlich zur Funktionalen und Verhaltensorientierten die Organisatorische Perspektive berücksichtigt. Für eine erste Einschätzung der Auswirkungen der neu hinzugekommenen Perspektive werden ebenfalls simplifizierende Prozessmodelle – dargestellt in Abbildung 8.5 – als Beispiele herangezogen. Tabelle 8.6 zeigt die Bewertung der Übersetzungsqualität von BPMN (B) in die Sprache DPIL (Dp) und umgekehrt. Im Unterschied zum ersten Versuchsblock zeigt sich hier, dass sich die Werte für die Appropriateness für das Modellpaar *(c)* und *(d)* schwerfälliger verbessern als bei dem ähnlichen Modellpaar aus Abbildung 8.4. Dieser Effekt ist auf die neu hinzugekommene Schleife, welche eine direkte Wiederholung von Aufgabe *B* gestattet, zurückzuführen. Eine wesentliche Erkenntnis ist, dass die Erschließung der organisatorischen Rollen in beide Translationsrichtungen in allen Konfigurationen ideal ist.

*Erfolgreicher Test mit künstlichen, kompakten Modellen*

*Ergebnisse: Versuchsblock B4*

Im letzten Versuchsblock – B4 – werden je ein DPIL- und ein BPMN-Modell eines universitären Dienstreiseprozesses in die jeweils andere Sprache übersetzt. Die Modelle sind insofern strikt, als keine der möglichen Ausführungsspuren mehr als zwölf Schritte beinhaltet. Aus diesem Grund wird auch die Analyse auf diese Spurlänge beschränkt, sodass nur die Anzahl der Spuren variiert. Flexibilität in den beiden Prozessmodellen entsteht vor allem bei der Wahl der Beförderungsmittel, deren Kombination beliebig ist. Wie Tabelle 8.7 zeigt, wird bezüglich der Ausführungssemantik in beiden Übersetzungsrichtungen eine ideale Übersetzungs-

*Erfolgreicher Test mit realen, komplexen Modellen*

| N | L | Fitness | App. | OP | N | L | Fitness | App. | OP |
|---|---|---|---|---|---|---|---|---|---|
| 10 | 12 | 0.856 | 0.696 | 1.0 | 10 | 12 | 0.811 | 0.717 | 1.0 |
| 100 | 12 | 1.0 | 0.772 | 1.0 | 100 | 12 | 0.925 | 0.798 | 1.0 |
| 1000 | 12 | 1.0 | 0.898 | 1.0 | 1000 | 12 | 0.966 | 0.941 | 1.0 |
| 10000 | 12 | 1.0 | 1.0 | 1.0 | 10000 | 12 | 1.0 | 1.0 | 1.0 |

Tabelle 8.7: Konformität: Versuchsblock B4 (BPMN ↔ DPIL – real, komplex)

qualität erzielt. Auffällig ist, dass hier die Appropriateness deutlich schlechter bewertet wird, als im Realprozessbeispiel von Untersuchungsblock B2. Die Zuweisung organisatorischer Rollen ist erneut in allen Konfigurationen ideal. Das ist vor allem auf die limitierte Ausdrucksmächtigkeit der Sprache BPMN zurückzuführen, da diese beispielsweise keine dynamische Rollenzuordnung gestattet.

Im aktuellen Abschnitt werden die vier in Abschnitt 8.4.2 beschriebenen Versuchsblöcke durchgeführt, um zu prüfen, ob und unter welchen Bedingungen das generische Übersetzungsprinzip korrekte Ergebnisse liefern kann. Die Blöcke unterscheiden sich hinsichtlich der verwendeten Beispielmodelle und -Sprachen. Diese sind entweder einzel- oder multi-perspektivisch und künstlich-kompakt oder real-komplex. Die Qualität der Übersetzungsergebnisse werden mittels der Metriken Fitness, Appropriateness und OP bewertet. Eine wesentliche Erkenntnis ist, dass zum Erreichen eines annähernd idealen Fitness-Wertes eine tendenziell große Anzahl an Ausführungsspuren benötigt werden. Dagegen stellt sich dieser ideale Wert bei der Appropriateness in der Mehrzahl der Fälle früher ein. Die Translation der organisatorischen Perspektive ist von Anzahl und Länge der Spuren weitestgehend unabhängig und führt in allen Versuchen zu idealen Ergebnissen. Allerdings ist die Ausdrucksmächtigkeit von BPMN hinsichtlich der Organisatorischen Perspektive gegenüber Sprachen wie DPIL limitiert. Die vorgestellten Ergebnisse gestatten lediglich eine erste Potentialeinschätzung des induktiven Translationsansatzes. Für eine allgemeingültige Einschätzung der Fehlerrate bezüglich der gewählten Sprachpaare oder gar unabhängig von diesen ist der Untersuchungsumfang deutlich zu gering. Aus diesem Grund ist die allgemeine Anwendbarkeit des induktiven Translationsansatzes zwar nicht widerlegt aber auch nicht *be*legt worden. In sich anschließenden Arbeiten sollten einerseits weitere Sprachpaarungen analysiert werden. Für jede Sprachpaarung ist es andererseits sinnvoll, die Übersetzungsergebnisse eine größeren Anzahl an Prozessmodellen zu bewerten. Auch eine Variation der Process-Mining-Techniken sollte Teil dieser fortführenden Untersuchungen sein.

## 8.5 QUALITATIVE UMFRAGEEVALUATION ZUR VERBESSERUNG DER VERSTÄNDLICHKEIT VON PROZESSMODELLEN

Während die Nützlichkeit der im Rahmen dieser Arbeit entwickelten Verfahren für Interoperabilität, Modellvergleich und Modellevolution größtenteils von deren Korrektheit abhängen, ist die Beurteilung der Nützlichkeit für die Verbesserung des Verständnisses von Prozessmodellen nicht zu pauschalisieren. Der wesentliche Grund dafür ist, dass diese Beurteilung nur subjektiv sein kann. Beispielsweise ist die Verständlichkeit eines Prozessmodells grundsätzlich von dem Grad der Vertrautheit mit der jeweiligen Prozessmodellierungssprache abhängig. Ein weiterer Faktor ist, dass mit zunehmender Komplexität des Modells die kognitive Belastung für den Betrachter steigt [93, 151], sodass die Nützlichkeit jeglicher Hilfsmittel auch von dieser Dimension abhängig ist. Aus diesen Gründen wird eine Umfrage durchgeführt, welche erste Erkenntnisse und Indizien liefert, um zu prüfende *Hypothesen* abzuleiten, welche sich auf die Nützlichkeit der drei bereitgestellten Hilfsmittel beziehen. Dies entspricht einem möglichen Ziel einer *qualitativen* Evaluation (Ab-

*Ziel: Hypothesen über die Nützlichkeit*

schnitt 8.5.1). Eine Untersuchung der generierten Hypothesen würde jedoch den Rahmen der vorliegenden Arbeit sprengen. Folglich wird keine absolute Entscheidung für oder gegen die Nützlichkeit der im Rahmen der vorliegenden Arbeit entwickelten Hilfsmittel getroffen. Allerdings sollen die nachfolgenden Abschnitte als Grundlage dienen, um diese Bewertung in sich anschließenden Arbeiten leichter erbringen zu können.

*Fokus: Deklarative Prozessmodellierungssprache*

Eine letzte Einschränkung des Untersuchungsbereichs ist die Fokussierung auf eine deklarative Prozessmodellierungssprache. Das ist damit begründet, dass der im Zuge dieser Arbeit entwickelte Spurgenerator sprachspezifisch ist. Aus diesem Grund orientiert sich die in der Umfrage verwendete Darstellung der Prozessentitäten und -regeln stark an der multi-perspektivischen, deklarativen Prozessmodellierungssprache DPIL.

Nach einer groben Einordnung (Abschnitt 8.5.1) der Untersuchungsmethode werden anschließend die Ziele der Umfrage zusammengefasst (Abschnitt 8.5.2). Darauf aufbauend folgt eine Erläuterung des grundsätzlichen Fragebogenaufbaus. Anschließend wird ein Überblick über Charakteristika und Verteilung der Teilnehmer gegeben. Der größere Teil der Umfrageergebnisse basiert nicht auf subjektiven Einschätzungen seitens der Teilnehmer, sondern auf Fragestellungen, die eine objektive Betrachtung gestattet. Diese Ergebnisse werden in einem weiteren Abschnitt analysiert. Darauf aufbauend und unter Ergänzung der Ergebnisse hinsichtlich subjektiver Einschätzungen der Teilnehmer werden schließlich Hypothesen bezüglich der Nützlichkeit aller drei Hilfsmittel extrahiert. Selbige können für sich anschließende Arbeiten als Grundlage für *spezifischere* Untersuchungen hinsichtlich der Nützlichkeit einzelner Hilfsmittel dienen

### 8.5.1 *Warum eine qualitative Umfrage?*

Für die Überprüfung des Potentials der drei im Rahmen dieser Arbeit entwickelten Techniken für die Verbesserung der Verständlichkeit von Prozessmodellen wird eine *qualitative Umfrage* durchgeführt. Die Begründung für diese Art der Evaluation gliedert sich in die Beantwortung der folgenden zwei Teilfragen: *(i)* Warum qualitativ und *(ii)* warum eine Umfrage? Im aktuellen Abschnitt werden diese Fragen beantwortet sowie sich daraus ergebende Konsequenzen und Anforderungen zusammengefasst.

*Warum qualitativ?*

Die Untersuchung der Eignung und Einsatzgebiete qualitativer und quantitativer Methoden ist bis heute Forschungsgegenstand – vor allem der Sozialforschung – und löst heftige Debatten aus [30, 72]. Dabei sind qualitative und quantitative Methoden keineswegs Antagonisten, sondern ergänzen sich häufig. Beispielsweise ist es möglich, mittels qualitativer Methoden Hypothesen zu generieren, welche dann in einer quantitativen Untersuchung verifiziert oder falsifiziert werden [30]. Das ist besonders dann hilfreich, wenn der Untersuchungsgegenstand kaum erforscht ist, was auf die in der vorliegenden Arbeit betrachteten drei Techniken zutrifft. Für diese sind keine Untersuchungen der Verständlichkeitsproblematik bekannt. Aus diesem Grund ist es sinnvoll, zunächst Hypothesen für die Einflüsse der Hilfsmittel auf die Verständlichkeit von Prozessmodellen aufzustellen.

*Warum Umfrage?*

Nachdem die Entscheidung für eine *qualitative* Methode begründet ist, bleibt die Diskussion der Festlegung auf eine Umfrage. Verständlichkeit ist kein objektives, sondern ein subjektives Maß. Aus diesem Grund ist es sinnvoll, bei der Evaluation von Hilfsmitteln zur Verbesserung der Verständlichkeit von Prozessmodellen potentielle Nutzer diesbezüglich zu befragen [30, 72]. Auf Basis der Resultate dieser Befragungen können anschließend potentiell zu verallgemeinernde Hypothesen identifiziert werden, was nicht zuletzt auch die Eignung von Umfragen für eine qualitative Evaluation zeigt. Bei diesen stehen häufig die Befragten und ihre subjektiven Meinungen und Eindrücke im Vordergrund [72].

*Konsequenzen*

Die Festlegung auf eine qualitative Befragung zieht mehrere Konsequenzen nach sich. Als vermutlich wichtigster Punkt ist hier die Unterscheidung zwischen den Stichproben quantitativer und dem sogenannten *theoretischen Sampling* qualitativer Verfahren zu diskutieren. Stichproben werden in der Regel zufällig ausgewählt. Theoretisches Sampling bedeutet dagegen eine gezielte Zusammenstellung der betrachteten Fälle. Der für die Befragung verwendete Fragebogen (Abschnitt 8.5.3) wurde aus diesem Grund einer ausgewählten Gruppe an Probanden vorgelegt, die sich in etwa hälftig in Experten und Nicht-Experten für Prozessmodellierung aufteilen. Gleichzeitig handelt es sich bei diesen Probanden um typische Vertreter der potentiellen Nutzergruppe für die in der vorliegenden Arbeit vorgestellten Techniken. Das entspricht den Anforderungen eines theoretischen Samplings für die qualitative Evaluation [72, S. 265].

*Repräsentativität*

Neben der Zusammensetzung ist auch die Größe der Gruppe an Probanden ein wichtiger Faktor. Bei quantitativen Methoden stellt sich die Frage der statistischen *Repräsentativität*. Dies gilt für qualitative Untersuchungen naturgemäß nicht, was eine Folge der gezielten Auswahl der Probanden ist [104, S. 183]. Zudem fokussieren sich qualitative Methoden auf Meinungen und Eindrücke der Befragten. Nur weil die Mehrzahl dieser eine bestimmte Meinung teilen, bedeutet dies nicht zwangsläufig, dass sie automatisch korrekt ist. Umgekehrt kann aber auch eine Einzelmeinung allgemeingültig und korrekt sein. Da durch die Befragung im Rahmen der vorliegenden Arbeit lediglich erste Hypothesen abgeleitet werden, spielt die Größe der Probandengruppe eine eher untergeordnete Rolle. Ziel der Hypothesenbildung ist die Festlegung zukünftiger Untersuchungsrichtungen, nicht aber die Formulierung finaler Gesetzmäßigkeiten.

*Abweichungen vom Standardvorgehen*

Der anhaltende Streit über Eignung und Einsatztauglichkeit qualitativer oder quantitativer Evaluationsmethoden ist vielfältig Anlass zur Kompromissbildung oder zur Vermischung der beiden Typen [46]. Aus diesem Grund ist der für die Umfrage verwendete Fragebogen (Abschnitt 8.5.3) entgegen der üblichen Vorgehensweise bei qualitativen Befragungen nicht offen, sondern gibt für jede Frage ein Antwortspektrum vor. Dies ermöglicht zwar eine spätere Quantifizierung der Umfrageergebnisse, dient aber lediglich der Vereinfachung der Generierung von Hypothesen. Ziel ist demnach die Festlegung auf qualitative Behauptungen, nicht aber eine differenzierte, quantitative Bewertung des Verständlichkeitsgewinns durch die drei betrachteten Techniken. Dies entspricht auch dem wesentlichen Ziel dieser ersten Evaluation, einen vorläufigen und groben Eindruck zu gewinnen, ob die entwickelten Hilfsmittel potentiell zur Verbesserung der Verständlichkeit von Prozessmodellen beitragen können.

### 8.5.2 *Ziele der Umfrage*

Der Begriff der *Nützlichkeit* der entwickelten Hilfsmittel für die Verbesserung des Verständnisses von Prozessmodellen erfasst das Thema der Umfrage, lässt aber die konkrete Zielsetzung derselben offen. Im aktuellen Abschnitt werden deshalb mehrere Ziele benannt, welche mit der Durchführung der Umfrage verfolgt werden. Anschließend wird zur klareren Abgrenzung auch auf Ziele eingegangen, welche *nicht* verfolgt werden.

*Zweck: Potentialeinschätzung für Artefakte*

Da zum aktuellen Zeitpunkt keine früheren Umfrageergebnisse zur selben oder einer ähnlichen Thematik existieren, dient die im Rahmen dieser Arbeit durchgeführte Umfrage der ersten Orientierung und generellen *Potentialeinschätzung* der drei genannten Hilfsmittel. Als Hilfsmittel werden im Rahmen dieser Umfrage die *Artefakte* bezeichnet, welche sich mit den im Konzeptteil (Teil ii) vorgestellten Techniken erzeugen lassen. Die Umfrage fokussiert sich demnach auf die Identifikation von Hypothesen über die Nützlichkeit von Beispielen für valide oder nicht-valide Prozessausführungsspuren, von Prozessmodellen, welche in alternativen Sprachen dargestellt werden und von natürlichsprachlichen Prozessbeschreibungen für die Verbesserung der Verständlichkeit von Prozessmodellen. Da derartige Artefakte nicht nur mit den im Rahmen dieser Arbeit entwickelten Techniken erzeugt werden können, kann die Umfrage keine Aussage über die Nützlichkeit der drei Techniken selbst treffen. Stattdessen wird von einer Idealvorstellung bezüglich der Qualität der erzeugten Artefakte ausgegangen. Sich anschließende Forschungsarbeiten müssen demnach gezielt prüfen, ob die drei Techniken diesen Qualitätsansprüchen genügen können und ob der Zusatzaufwand, der bei ihrer Verwendung entsteht, ihren Nutzen einschränkt oder gar aufhebt. Die vorläufige Fokussierung auf die erzeugten Artefakte ist sinnvoll, da, falls sich bereits bei optimaler Qualität derselben kein Nutzen abzeichnet, das Artefakt den angedachten Zweck nicht erfüllt. Folglich ist es unerheblich, ob eine Technik zur Erzeugung eines somit unzweckmäßigen Artefakts in optimaler Qualität fähig ist.

*Keine Aussage über Nützlichkeit der Techniken*

Zur Untersuchung der Nützlichkeit der drei Hilfsmittel werden die nachfolgenden Ziele definiert:

*Verständlichkeit*

G1 Das erste Ziel ist die Erkennung einer Tendenz, ob bei Prozessmanagementaufgaben – wie beispielsweise Prozessmodellierung und Modellvergleich – hinsichtlich der Verständlichkeit bestimmter deklarativer Prozessmodelle Handlungsbedarf besteht oder nicht.

*Nutzen der Hilfsmittel*

G2 Ein zweites Ziel ist die Erkennung einer Tendenz, ob eines oder mehrerer der drei Hilfsmittel das Potential haben, zu einer Verbesserung dieses Verständnisses beizutragen.

*Eignung für Experten/Nicht-Experten*

G3 Ein weiteres Ziel beinhaltet die Erkennung einer Tendenz, ob eines oder mehrere Hilfsmittel eher für Experten respektive Nicht-Experten geeignet sind.

*Nutzen der Hilfsmittel je nach Aufgabe*

G4 Das letzte Ziel ist die Erkennung einer Tendenz, ob sich die Nützlichkeit der Hilfsmittel je nach Art der Aufgabe unterscheidet.

Das Gros der genannten Ziele kann subjektiv und objektiv betrachtet werden. Eine subjektive Komponente für das Ziel G1 ist eine Selbsteinschätzung hinsichtlich des Verständnisses und der Intuitivität bestimmter deklarativer Prozessmodelle. Eine objektive Komponente ist die Simulation ausgewählter Aufgaben des Prozessmanagements und die Erfassung der Fehlerrate bei deren Bewältigung durch eine Teilnehmergruppe, ohne weitere Hilfsmittel zur Verfügung zu stellen.

Um eine Tendenz für die Nützlichkeit der Hilfsmittel abzuleiten, kann vom Teilnehmer – subjektiv – der Grad der Verbesserung der Verständlichkeit eingeschätzt werden (G2). Eine objektivere Einschätzung kann dadurch ermöglicht werden, dass zwei disjunkten Gruppen von Teilnehmern eine Prozessmanagementaufgabe gestellt wird und nur einer dieser beiden Gruppen die drei Hilfsmittel zur Verfügung gestellt werden. Dadurch wird eine Potentialeinschätzung möglich, die auf dem Vergleich der Fehlerraten beider Gruppen bei der Bewältigung der jeweiligen Prozessmanagementaufgabe basiert. Erfasst man zusätzlich, welche der zur Verfügung gestellten Hilfsmittel jeweils Anwendung finden, kann damit die subjektive Komponente dieses Ziels erweitert werden.

Ziel G3 ist das Erkennen einer Tendenz, ob sich die Nützlichkeit der Hilfsmittel je nach Vertrautheit des jeweiligen Teilnehmers in Bezug auf die Prozessmodellierung unterscheidet. Grundsätzlich deckt sich der Weg zur Erreichung dieses Ziels mit dem für das Ziel G2. Im Gegensatz zu letzterem müssen hier jedoch die Probanden als Experten respektive Nicht-Experten klassifiziert werden.

Im Rahmen der Zielsetzung G4 ist zu prüfen, inwiefern die Nützlichkeit der Hilfsmittel von der jeweiligen Prozessmanagementaufgabe abhängig ist. Beispielsweise kann sich der Nutzen jedes Hilfsmittels für die manuelle Überprüfung eines Prozessmodells vom Nutzen für den Vergleich zweier Prozessmodelle unterscheiden. Folglich ist das Ziel G4 nur dadurch zu erreichen, dass in der Umfrage mindestens zwei verschiedene Prozessmanagementaufgaben zu erfüllen sind. Ein Vergleich der Fehlerrate bei beiden Aufgaben und einer zufälligen Verfügbarkeit der Hilfsmittel für die Probanden ist ein möglicher Weg zur Erreichung des Ziels.

Grundsätzlich ist davon auszugehen, dass jedes der Ziele nur angenähert werden kann, da das Verständnis abstrakter Darstellungen von einer Vielzahl individueller Faktoren abhängig ist – wobei der Teilnehmer selbst als einer der einflussreichsten Faktoren anzusehen ist [93, 151]. Somit sind alle Umfrageergebnisse, welche zur Erreichung der oben genannten Ziele ermittelt werden, mit einem *Bias* behaftet. Eines der wichtigsten relevanten Charakteristika des Teilnehmers ist dessen Vertrautheit mit der Prozessmodellierung im Allgemeinen und der Modellierung mit deklarativen Prozessmodellierungssprachen im Speziellen. Da zu berücksichtigen ist, dass die oben genannten Ziele möglicherweise nicht für die Gesamtheit der Befragten erreicht werden können, ist die Differenzierbarkeit nach wesentlichen Teilnehmercharakteristika obligatorisch. Dies dient außerdem dazu, die Auswirkungen des zu erwartenden Bias gegebenenfalls abschwächen zu können.

*Bias*

*Differenzierung nach Teilnehmercharakteristika*

Abschließend werden, um Missverständnissen vorzubeugen, einige Ziele genannt, welche mit der Durchführung der Umfrage ausdrücklich *nicht* verfolgt werden. So ist vor allem eine endgültige Entscheidung über die Nützlichkeit der drei vorgestellten Hilfsmittel nicht Teil der Zielsetzung. Weiterhin beziehen sich alle subjektiven Einschätzungen der Befragten auf *eigene*, persönliche Eindrücke während

der Beantwortung der einzelnen Fragen. Experten der Prozessmodellierung werden beispielsweise *nicht* befragt, um ihre subjektive Einschätzung der Verständlichkeit bestimmter Prozessmodellkonstrukte für die Allgemeinheit in Erfahrung zu bringen. Es ist weiterhin außerhalb des Rahmens der Betrachtung, ob sich hinsichtlich der Umfrageergebnisse Unterschiede zwischen einer Personengruppe mit akademischen und einer mit industriell-wirtschaftlichem Tätigkeitshintergrund abzeichnen. Aus diesem Grund wird bei der Differenzierung der Ergebnisse eine solche Einteilung der Befragten unterlassen. Die in der Umfrage verwendete multi-perspektivische, deklarative Prozessmodellierungssprache ist an die Sprache DPIL angelehnt. Allerdings sind alle in der Umfrage präsentierten Modelle ausschließlich mit informal beschriebenen Regelschablonen formuliert. Diese unterscheiden sich namentlich von den Schablonen der veröffentlichten Bibliothek für Modellierungsmuster, um routinebedingte Flüchtigkeitsfehler seitens der DPIL-Sprachexperten zu minimieren.

Die oben genannten Ziele gelten für die vorliegende Arbeit als erreicht, wenn sich für diese, auf Basis der Umfrageergebnisse, zu begründende Hypothesen ableiten lassen. Diese Hypothesen können die Grundlage für sich anschließende Forschungsarbeiten bieten.

### 8.5.3 *Fragebogenaufbau*

*Fragebogen*

Der für die Umfrageevaluation verwendete Fragebogen sowie die erhobenen Rohdaten sind unter `http://mps.kppq.de` zum Download verfügbar. Der Fragebogen gliedert sich in vier Teile, deren Inhalte nachfolgend zusammengefasst sind.

*Teil 1*

Im *ersten Teil* der Umfrage wird das Verständnis von Prozessregeln in Isolation betrachtet. Dazu werden nacheinander vier Prozessregeln präsentiert, die sich in Komplexität und Bezug zu den fünf Prozessperspektiven (Abschnitt 1.1.2) unterscheiden. Diese Prozessregeln sind an die von DPIL unterstützten Makro-Definitionen angelehnt, wobei zu jeder Regel eine knappe natürlichsprachliche Erläuterung angegeben ist. Ein Beispiel dafür ist die folgende Prozessregel:

```
order(A,B) // Semantics: If B then A before.
```

Zu dieser Regel werden von jedem Teilnehmer zunächst das Verständnis sowie die damit verbundene Intuitivität erfragt, was zum Erreichung des Ziels G1 beiträgt.

*Hilfsmittel*

Anschließend werden die folgenden drei Hilfsmittel zur Verfügung gestellt:

- *Positiv-/Negativbeispiele* für/gegen valide Prozessausführungen: Es wird der in Kapitel 4 vorgestellte Spurgenerator verwendet, um für die DPIL-Entsprechung der dargestellten Prozessregel eine geringe Anzahl an Beispielen valider und nicht-valider Prozessausführungen zu generieren.
- *Imperatives Prozessmodell*: Diese Prozessmodelle sind hinsichtlich ihrer Ausführungssemantik mit dem regelbasierten Modell deckungsgleich. Es wird zur Darstellung die Standard-Notation BPMN verwendet. Die Prozessmodelle werden mittels der in Kapitel 5 beschriebenen induktiven Translationstechnik aus der DPIL-Entsprechung der Prozessregel erzeugt.

- *Natürlichsprachlicher Beschreibungstext*: Zusätzlich zu der knappen natürlichsprachlichen Regelerläuterung wird ein natürlichsprachlicher Text zur Verfügung gestellt, welcher die Regel anhand kompakt beschriebener Szenarien detaillierter erklärt. Diese Texte werden mit der in Kapitel 6 beschriebenen NLG-Technik erzeugt, sind aber zur Kompensation der technologischen Limitierungen der bisherigen Implementierung leicht vereinfacht und verbessert dargestellt. Beispielsweise unterstützt die aktuelle Umsetzung keine Zerlegung der Aktivitätsbezeichner in Nomen und Verb.

Der Teilnehmer wird aufgefordert, zu jedem dieser Hilfsmittel eine Bewertung abzugeben, die seine Einschätzung hinsichtlich des Grades der Verbesserung der Verständlichkeit von Prozessregeln in Isolation widerspiegeln (Beitrag zu Ziel G2).

*Teil 2*

Im *zweiten Teil* wird die Verständlichkeit mehrerer Prozessregeln in Kombination behandelt. Dazu ist ein kompaktes, regelbasiertes Prozessmodell gegeben, welches die Ausführung des beschriebenen Prozesses bezüglich der Verhaltens- und Datenorientierten Perspektive beschränkt. Anschließend werden dem Teilnehmer mehrere Aussagen präsentiert, welche zum Teil die Semantik des Prozessmodells widerspiegeln. Einige davon sind jedoch unzutreffend. Aufgabe des Teilnehmers war es, zu entscheiden, welche der Aussagen auf das gegebene Prozessmodell zutreffen respektive nicht zutreffen. Einem Teil der Probanden werden dazu erneut die drei im Rahmen dieser Arbeit entwickelten Hilfsmittel zur Verfügung gestellt. Diese Probanden sollten zusätzlich angeben, welche der bereitgestellten Hilfsmittel ihnen die beste Hilfestellung für die Bewertung der Aussagen geliefert haben. Die Ergebnisse dieses Teils sollen einen Beitrag zur Erreichung aller vorab genannten Ziele liefern.

*Teil 3*

*Teil drei* des Fragebogens beschäftigt sich mit dem Vergleich von regelbasierten mit imperativen Prozessmodellen. Dazu werden dem Teilnehmer paarweise je ein regelbasiertes und ein imperatives Prozessmodell – letzteres erneut mittels BPMN dargestellt – präsentiert. Der Teilnehmer war dann dazu aufgefordert, zu entscheiden, ob das jeweilige Modellpaar hinsichtlich seiner Ausführungssemantik äquivalent ist oder nicht. Falls der Teilnehmer einen Unterschied feststellte, wird er anschließend gebeten, eines oder mehrere Beispiele für Ausführungspfade anzugeben, die von einem der beiden Modelle nicht unterstützt werden. Letzteres diente dazu, die Schwierigkeit zu erhöhen, durch bloßes Raten die richtige Antwort auszuwählen. Die verschiedenen Modellpaare unterscheiden sich erneut hinsichtlich Komplexität und Berücksichtigung der einzelnen Prozessperspektiven. Auch dieser Teil solle einen Beitrag zu jedem der vorab definierten Ziele leisten. Ein Vergleich der Fehlerrate bei den Modellvergleichen in diesem Teil und der Fehlerrate bei der manuellen Modellprüfung im vorherigen Teil ist beispielsweise ein Fortschritt zur Erreichung von Ziel G4.

*Teil 4*

Der *vierte* und letzte *Teil* dient einerseits dazu, Informationen über den fachlichen Hintergrund der Teilnehmer zu ermitteln. Andererseits wird jeder Teilnehmer auch gebeten, eine persönliche Einschätzung abzugeben, in welchen Situationen die drei Typen von Hilfsmitteln *tatsächlich* hilfreich sein könnten respektive waren. Dieser Teil hilft einerseits dabei, die Teilnehmer in Experten und Nicht-Experten zu klassifizieren. Andererseits dient die erneute Nachfrage nach der subjektiv

eingeschätzten Nützlichkeit der einzelnen Hilfsmittel der Kontrolle vorheriger subjektiver Einschätzungen.

Die Umfrage wurde im Zeitraum von etwa zwei Monaten (29.03.–28.05.2017) mittels der Online-Plattform *LamaPoll*[11] durchgeführt. Alle Ergebnisse der Umfrage werden nachfolgend vorgestellt und ausgewertet.

### 8.5.4 *Umfrageergebnisse: Teilnehmer und Verteilung*

Insgesamt haben an der Umfrage 36 Personen aus unterschiedlichen Tätigkeitsfeldern teilgenommen. Für eine differenzierte Betrachtung der Umfrageergebnisse wird im aktuellen Abschnitt auch eine Klassifikation der Teilnehmer anhand zweier weiterer Charakteristika vorgenommen. Dazu zählt einerseits die Unterteilung der Teilnehmer in Experten und Nicht-Experten im Bereich der Prozessmodellierung. Das zweite Charakteristikum beschreibt, ob dem Teilnehmer die drei im Rahmen der vorliegenden Arbeit diskutierten Hilfsmittel zur Verfügung gestellt wurden oder nicht.

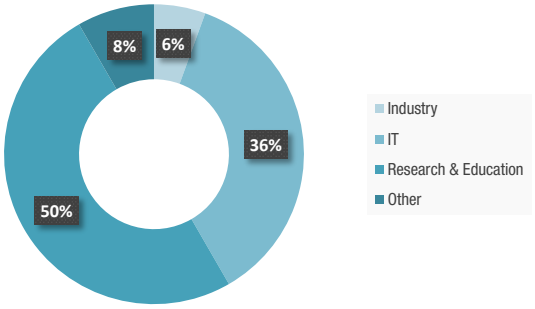



Abbildung 8.7: Verteilung der Teilnehmer über Fachbereiche

*Verteilung: Tätigkeitsfelder*

Wie das in Abbildung 8.7 dargestellte Diagramm zeigt, ist jeweils fast die Hälfte der Befragten entweder im Forschungs- und Bildungssektor oder im Bereich der Informationstechnik tätig. Ein kleinerer Teil der Teilnehmer hat dagegen einen Industriehintergrund.

In Abbildung 8.8 ist eine Übersicht über die Klassifikation der Probanden in Experten und Nicht-Experten der Prozessmodellierung dargestellt.

Die Klassifikation basiert auf einer Selbsteinschätzung der Teilnehmer und ist dementsprechend unscharf. Zudem stellt die Projektion von fachlicher Expertise auf eine diskrete numerische Skala eine starke Vereinfachung der Realität dar. Da jedoch diesbezüglich keine objektive Einschätzung vorliegt, wird nachfolgend von der in Abbildung 8.8 dargestellten subjektiven Klassifikation ausgegangen.

*Verteilung: Experten und Nicht-Experten*

Wie das Diagramm zeigt, ergibt sich somit ein Verhältnis von 58% Experten zu 42% Nicht-Experten. Diese Verteilung stellt durch ihre Ausgewogenheit eine gute Ausgangsbasis für die Analyse der Umfrageergebnisse dar. Somit ist das Potential gegeben, durch die im Rahmen der vorliegenden Arbeit bereitgestellten Hilfsmittel einen Erkenntnisgewinn in Abhängigkeit von der Erfahrenheit der Teilnehmer zu erzielen.

11 https://www.lamapoll.de/, abgerufen am: 24.08.2017

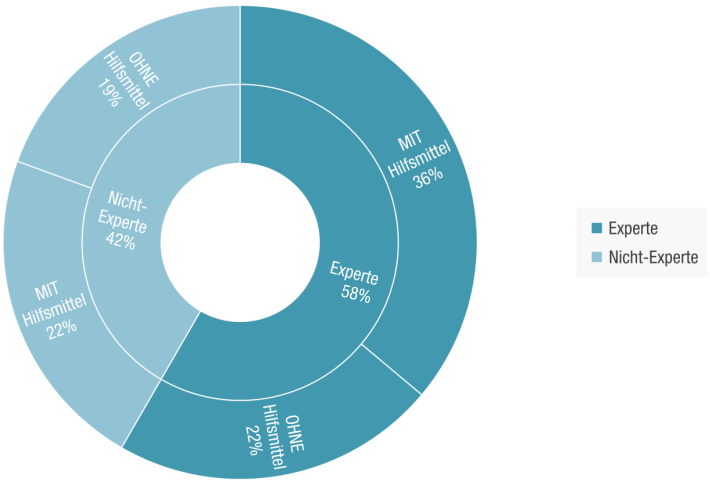



Abbildung 8.8: Verteilung der Teilnehmer: Experte/Nicht-Experte und mit/ohne Verfügbarkeit der Hilfsmittel

*Verteilung: Verfügbarkeit der Hilfsmittel*

Orthogonal zur Expertise jedes Probanden wurde zufällig entschieden, ob die beschriebenen Hilfsmittel für den zweiten und dritten Teil der Umfrage zur Verfügung gestellt werden oder nicht. Zwar wäre eine exakte Gleichverteilung der Probanden wünschenswert gewesen, durch Limitierungen der Umfrageplattform war dies jedoch nicht möglich. In Abbildung 8.8 ist die Verteilung der Probanden hinsichtlich der Verfügbarkeit der drei Hilfsmittel dargestellt.

Wie das Diagramm zeigt, hat sich auch hier eine annähernde Gleichverteilung eingestellt. Dadurch ist eine Gegenüberstellung der zwei Gruppen von Probanden möglich.

Das Gros der Probanden ist entweder im Forschungs- und Bildungs- oder im IT-Sektor tätig ist. Als zweites Kriterium wird die Expertise der Teilnehmer in Bezug auf die Modellierung von Geschäftsprozessen verwendet. Daraus ergibt sich, dass die Zahl der Experten geringfügig die Zahl der Nicht-Experten übersteigt. Gleiches gilt für die zufallsbasierte Entscheidung, ob einem Teilnehmer die drei Hilfsmittel zur Verfügung gestellt wurden oder nicht. Hier überwiegen zu einem geringen Prozentsatz die Fälle, in denen Hilfsmittel zur Verfügung standen.

### 8.5.5 *Umfrageergebnisse: Subjektive Messungen*

Inhalt des ersten Umfrageteils war es einerseits, zu ermitteln, wie verständlich und intuitiv – nach *subjektiver* Einschätzung der einzelnen Teilnehmer – einzelne Prozessregeln in Isolation sind. Andererseits sollte in gleicher Manier der Verständnisgewinn durch die Nutzung der drei bereitgestellten Hilfsmittel eingeschätzt werden. Durch die abschließende Ermittlung der Vertrautheit des Nutzers mit verschiedenen Aspekten der Prozessmodellierung, können die Ergebnisse entsprechend differenziert werden. Der aktuelle Abschnitt stellt die diesbezüglich erzielten Ergebnisse vor.

In Abbildung 8.9 ist dargestellt, wie verständlich die Beispielregeln aus Sicht der Umfrageteilnehmer waren. Dabei steht der Wert 1 für unzureichende und der Wert 5 folglich für sehr gute Verständlichkeit.

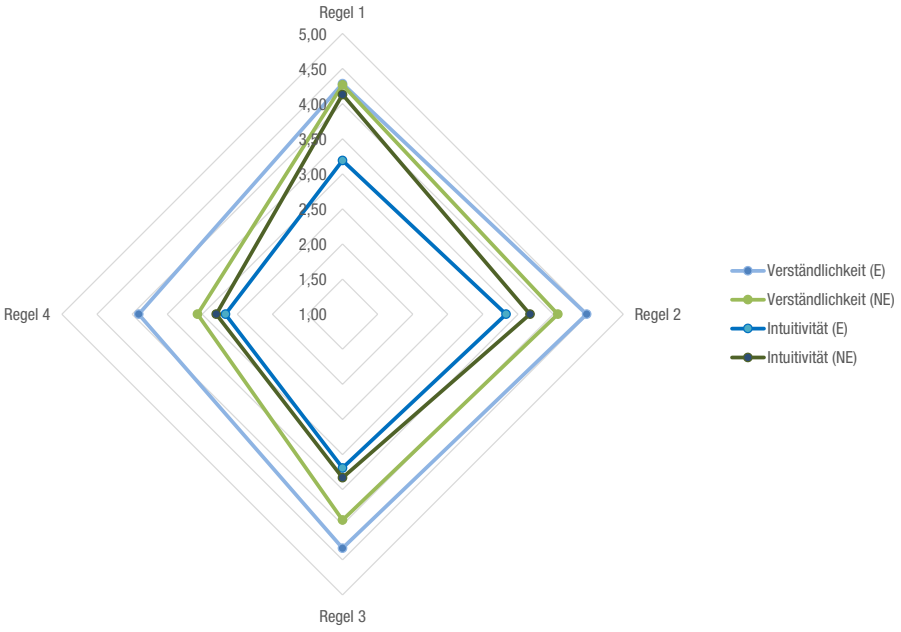



Abbildung 8.9: Verständlichkeit und Intuitivität der Regelbeispiele

*Keine Regel vollkommen verständlich*

Wie das Diagramm zeigt, wird keine der Beispielregeln als vollkommen verständlich angesehen. Die Regeln folgten lediglich einer semi-formalen Notation, deren Semantik jedoch nur in Form eines knappen Kommentars für jede Regel spezifiziert ist. Aufgrund der großen Sprachvielfalt im Bereich der Prozessmodellierung kommen selbst Experten dieses Bereichs häufig mit zwar vertraut anmutenden aber in ihrer Semantik vollständig nur schwer zu erfassenden Sprachen in Kontakt. Durch die sporadisch zur Verfügung gestellten Informationen wurde diese Situation simuliert. Folglich ist es auch nicht verwunderlich, dass Nicht-Experten ihrer eigenen Einschätzung nach ein geringeres Verständnis für die Beispielregeln haben als die Expertengruppe.

Die Intuitivität wird durch Experten und Nicht-Experten ebenfalls subjektiv eingeschätzt. Sind Prozessregeln intuitiv, dann sind zusätzliche Hilfsmittel zur Erklärung derselben überflüssig. Sind sie nicht intuitiv, können dagegen Hilfsmittel zum Einsatz kommen. Dieser Gedanke ist die Motivation der Fragestellung, deren Ergebnisse ebenfalls in Abbildung 8.9 visualisiert sind. Die Skala ist mit der der Visualisierung zur Regelverständlichkeit identisch.

*Schlechtere Ergebnisse für Intuitivität*

Eine Betrachtung des Diagramms zeigt, dass die Intuitivität im Vergleich zur Verständlichkeit schlechter bewertet wird. Im Gegensatz zu letzterer ist die Intuitivität jedoch von den Nicht-Experten insgesamt besser bewertet worden als von den Experten der Prozessmodellierung. Sowohl die Analyse der Regelverständlichkeit als auch die der Intuitivität lassen einen Handlungsbedarf zur Verbesserung des Verständnisses der Prozessregeln erkennen (Ziel G1).

Für jede der präsentierten Beispielregeln sind jeweils Positiv- und Negativbeispiele, ein ausführlicher natürlichsprachlicher Beschreibungstext sowie – falls möglich – ein imperatives Modell mit äquivalenter Ausführungssemantik angegeben. In einer weiteren subjektiven Einschätzung sollten sowohl Experten als auch Nicht-Experten den Gewinn an Verständlichkeit für jede Regel bewerten. Das Er-

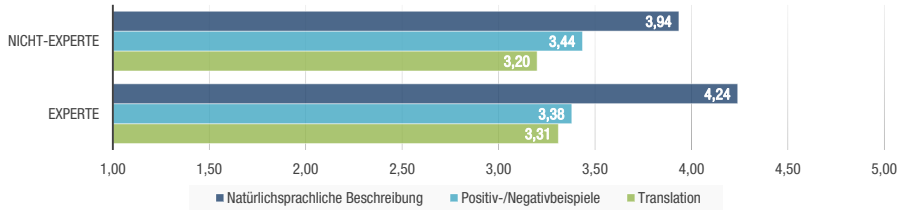



Abbildung 8.10: Verbesserung des Verständnisses von Prozessregeln in Isolation (subjektiv)

gebnis dieses Umfrageabschnitts ist in Abbildung 8.10 dargestellt. Die verwendete Skala für die diesbezügliche Nützlichkeit der Hilfsmittel reicht erneut von 1 bis 5. Der niedrigste Wert steht dabei für einen unzureichenden Nutzen, der höchste entsprechend für einen hohen Nutzen.

*Bewertung für Regeln in Isolation*

Das Diagramm zeigt die Bewertung der Nützlichkeit der Hilfsmittel differenziert nach der Expertise des Befragten. Auffällig ist dabei, dass die sich ergebende Rangfolge der drei Hilfsmittel bei beiden Teilnehmergruppen identisch ist. Natürlichsprachliche Beschreibungen werden als am nützlichsten eingeschätzt, Positiv- und Negativbeispiele sowie eine Translation in ein imperatives Modell folgen mit einigem Abstand. Der größte Unterschied ist, dass die Gruppe der Nicht-Experten das Instrument einer natürlichsprachlichen Beschreibung der Prozessregeln mit noch deutlicherem Abstand als am hilfreichsten einschätzt. Im Hinblick auf die geringere Erfahrung im Bereich der Prozessmodellierung stimmt dies mit den Erwartungen überein.

Die Bewertung der Nützlichkeit der drei Hilfsmittel für Prozessregeln in Kombination ist ebenfalls erfasst. Zur Semantik eines gegebenen regelbasierten Prozessmodells mussten vorgegebene Aussagen manuell auf ihre Richtigkeit geprüft werden. Hierbei wurden nur einem Teil der Probanden die drei Hilfsmittel zur Verfügung gestellt. Bei jeder Aussage wurde der Teilnehmer zusätzlich gebeten, einzuschätzen, welche der drei Werkzeuge für die Bewältigung der Aufgabe subjektiv am hilfreichsten waren. Dabei ist sowohl die Auswahl keines oder mehrerer Werkzeuge gestattet gewesen. In Abbildung 8.11 ist die relative Häufigkeit der Verwendung aller drei Hilfsmittel dargestellt.

*Bewertung für Regeln in Kombination*

Aus der Darstellung wird ersichtlich, dass jedes der Hilfsmittel Anwendung gefunden hat und dass die Rangfolge der am häufigsten verwendeten Hilfsmittel bei beiden Teilnehmergruppen erneut identisch ist. Die Häufigkeit der Anwendung der einzelnen Werkzeuge unterscheidet sich jedoch zum Teil drastisch. Nicht-Experten stuften häufiger alle drei Hilfsmittel als äußerst hilfreich ein als Experten. Bei allen bildeten die bereitgestellten Positiv- und Negativbeispiel für Prozessausführungsspuren die Spitze. In zwei Dritteln aller Fälle wurde dieses Mittel als äußerst hilfreich bewertet. Für Nicht-Experten hatten die natürlichsprachlichen Beschreibungen einen in etwa vergleichbaren Wert. Experten schätzen dieses Werkzeug jedoch nur in einem Drittel der Fälle als äußerst wertvoll ein. Verglichen mit der Bewertung der Hilfsmittel für das Verstehen von Prozessregeln in Isolation haben natürlichsprachliche Beschreibungen und beispielhafte Prozessausführungsspuren folglich den Rang getauscht (Ziel G4).

Da die Probanden durch ihre Teilnahme an der Umfrage selbst Erfahrung mit den bereitgestellten Hilfsmitteln erworben haben, wird im letzten Teil erneut um ei-

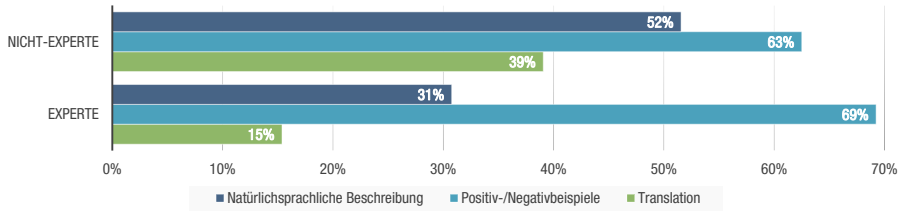



Abbildung 8.11: Verwendungshäufigkeit der Hilfsmittel für Prozessregeln in Kombination

ne subjektive Einschätzung der Nützlichkeit letzterer gebeten. Das Ergebnis ist in Abbildung 8.12 dargestellt. Es wird erneut nach Teilnehmergruppe und zusätzlich nach Anwendung der Werkzeuge auf Regeln in Isolation respektive in Kombination differenziert. Als Skala ist hier der Bereich von 1 bis 3 gewählt, um einerseits eine einfache „Wiederverwendung" des im ersten Umfrageteil angegebenen Wertes zu verhindern. Andererseits ist die Skala gröber gewählt, um eine stärkere Polarisierung zu erreichen.

*Bewertung der Nützlichkeit nach praktischer Anwendung der Hilfsmittel*

Für die Verbesserung des Verständnisses von Regeln in Isolation wird die im ersten Teil beobachtete Einschätzung bestätigt, dass natürlichsprachliche Beschreibungen den größten Beitrag leisten. Im Falle der Nicht-Experten werden nun jedoch Positiv- und Negativbeispiele als weitestgehend gleichwertig angesehen. Entgegen der vorherigen Beobachtung, dass für die manuelle Prüfung eines regelbasierten Prozessmodells Positiv- und Negativbeispiele am hilfreichsten sind, werden von beiden Teilnehmergruppen erneut die natürlichsprachlichen Beschreibungen als das Werkzeug mit dem größten Potential angesehen. Dieser Widerspruch kann mit der zur Verfügung stehenden Datengrundlage nicht zweifelsfrei aufgelöst werden. Allerdings lassen sich mehrere Vermutungen ableiten, die in sich anschließenden Forschungsarbeiten geprüft werden können: *(i)* Die in der Umfrage selbst formulierten Beschreibungstexte der regelbasierten Prozessmodelle beinhalteten nicht alle notwendigen Informationen, *(ii)* die Texte sind zu umfangreich respektive undeutlich formuliert oder *(iii)* natürlichsprachliche Texte sind zwar potentiell informativer aber Negativ- und Positivbeispiele stellen durch Kompaktheit und Vergleichbarkeit mit beliebigen Prozessmodellen ein praktisch nützlicheres Hilfsmittel dar.

*Verbesserung der Verständlichkeit erkennbar*

Aus subjektiver Sicht ist somit eine Tendenz zu erkennen, dass alle bereitgestellten Hilfsmittel einen mindestens durchschnittlichen Beitrag zur Verbesserung der Verständlichkeit von Prozessregeln leisten können. Die diesbezüglich größten Erfolge haben die natürlichsprachlichen Beschreibungen für Prozessregeln in Isolation sowie mit Positiv- und Negativbeispielen für Regeln in Kombination erzielt (Ziel G2).

*Bewertung differenziert nach Experten und Nicht-Experten*

Eine Tendenz für Unterschiede zwischen Experten und Nicht-Experten bezüglich der Bevorzugung der Hilfsmittel ist nicht oder höchstens schwach zu erkennen. So war die Rangfolge der bewerteten Hilfsmittel bei beiden Gruppen immer identisch. Allerdings zeigte sich bei der Prüfung von Aussagen bezüglich des gegebenen regelbasierten Prozessmodells, dass Nicht-Experten zwar ebenfalls Positiv- und Negativbeispiele favorisierten, jedoch öfter auch die beiden anderen Hilfsmittel verwendeten. Das war bei der Expertengruppe nicht der Fall (Ziel G3).

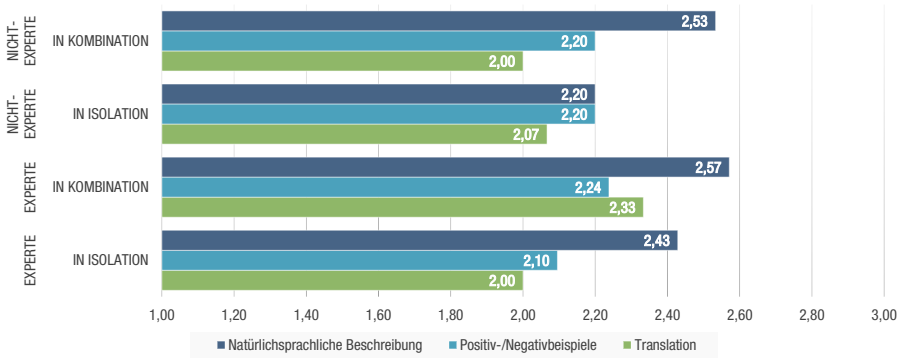



Abbildung 8.12: Verbesserung des Verständnisses von Prozessregeln in Isolation respektive Kombination

Auch wenn das grundsätzliche Verständnis von Prozesszusammenhängen in allen Fällen als durchschnittlich bis gut bewertet wird, zeigt unter anderem die schlechter bewertete Intuitivität der Regeln, dass tendenziell dennoch Handlungsbedarf zur Verbesserung der Verständlichkeit besteht. Aus Sicht der Teilnehmer bergen natürlichsprachliche Beschreibungen der Regeln und Positiv- sowie Negativbeispiele für Prozessausführungsspuren das diesbezüglich größte Potential.

### 8.5.6 *Umfrageergebnisse: Objektive Messungen*

Um die Nützlichkeit der drei Hilfsmittel zu prüfen reicht eine subjektive Einschätzung (Abschnitt 8.5.5) nicht aus. Daher wurden im zweiten und dritten Umfrageteil jeweils eine Prozessmanagementaufgabe gestellt. Für die Bewältigung dieser Aufgaben ist ein detailliertes Verständnis der Modellsemantik vonnöten. Um die Verbesserung des Verständnisses der Modellsemantik durch die drei Hilfsmittel objektiv zu evaluieren, wurden die Probanden zunächst zufallsgesteuert in zwei Gruppen aufgeteilt. Einer dieser beiden Gruppen wurden erneut die drei Hilfsmittel, also ein natürlichsprachlicher Text, Positiv- und Gegenbeispiele für oder gegen valide Prozessausführungen sowie ein hinsichtlich der Ausführungssemantik äquivalentes imperatives Prozessmodell zur Verfügung gestellt. Der Vergleich der Fehlerraten bei der Bewältigung dieser Aufgaben mit und ohne Hilfsmittel dient hier als objektive Komponente der Potentialeinschätzung für letztere.

*Messungen bei Unterstützung praktischer Prozessmanagementaufgaben*

Inhalt der beiden Aufgabenstellungen des zweiten und dritten Umfrageteils ist einerseits die Interpretation eines deklarativen Prozessmodells und andererseits ein Vergleich mehrerer solcher Modelle mit einem imperativen Prozessmodell. Wie im Verlauf von Abschnitt 8.5.3 dargelegt, sind die zu erwartenden Ergebnisse insofern objektiver Natur, als die Korrektheit jeder Eingabe des Teilnehmers zweifelsfrei geprüft werden kann.

Im zweiten Teil der Umfrage sind neben einem regelbasierten Prozessmodell auch acht Aussagen gegeben, die nur zum Teil auf das Modell zutreffen. Aufgabe war es, zu entscheiden, welche der Aussagen über das präsentierte Prozessmodell wahr respektive falsch sind. Dabei hat sich insgesamt das in Abbildung 8.13 dargestellte Resultat eingestellt. Im Diagramm sind die Ergebnisse einerseits nach der Klassifikation des Teilnehmers als Experte oder Nicht-Experte und andererseits

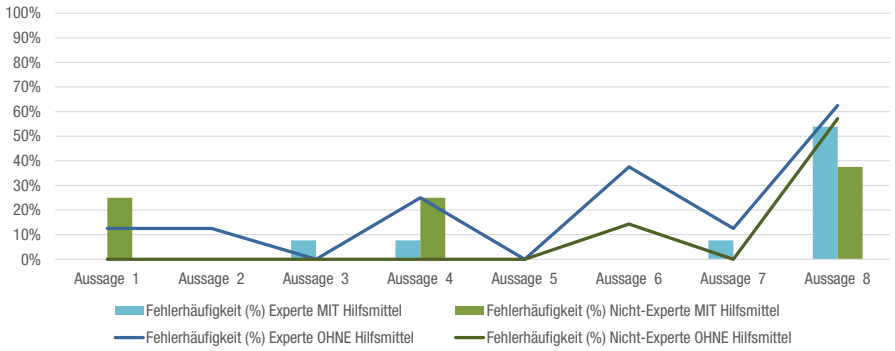



Abbildung 8.13: Fehlerhäufigkeit bei manueller Prüfung der Modellsemantik

nach der Verfügbarkeit der Hilfsmittel differenziert. Die Höhe der Säulen und der Linien des Diagramms gibt die relative Häufigkeit dafür an, dass die Bewertung der jeweiligen Aussage durch den Teilnehmer *fehlerhaft* ist.

*Ergebnisse: Fehlerhäufigkeit bei Modellanalyse*

Um den Unterschied in der Fehlerhäufigkeit mit und ohne Verwendung der Hilfsmittel zu visualisieren, wird sowohl für die Experten- als auch die Nicht-Experten-Gruppe eine Balkendarstellung mit einer Liniendarstellung kombiniert. Entgegen den Erwartungen ist zwischen Experten und Nicht-Experten kein deutlicher Unterschied bezüglich der Fehlerhäufigkeit zu erkennen. Allerdings ist bei der Expertengruppe bei sechs von acht Aussagen häufiger korrekt entschieden worden, wenn die Hilfsmittel zur Verfügung standen. Bei Aussage 5 traten weder mit noch ohne Hilfsmittel Fehler auf, sodass sich lediglich bei Aufgabe 3 die Fehlerrate geringfügig erhöht hat. In Bezug auf die Nicht-Experten-Gruppe stellt sich hier ein anderes Bild dar. Lediglich in zwei von acht Fällen ist die Fehlerhäufigkeit mit Verwendung der Hilfsmittel geringer als ohne selbige. In vier weiteren Fällen hatte die Verfügbarkeit der Hilfsmittel keinen Einfluss, was aus der identischen Fehlerhäufigkeit abgeleitet werden kann. Folglich suggerieren diese Ergebnisse, dass die Nützlichkeit der Hilfsmittel für die Expertengruppe größer war (G3).

In Abbildung 8.14 ist die Fehlerhäufigkeit beim Vergleich deklarativer mit imperativen Prozessmodellen hinsichtlich der Äquivalenz ihrer Ausführungssemantik visualisiert. Jeder Proband wurde gebeten, diese Äquivalenz zu prüfen und im Falle eines festgestellten Unterschieds ein Beispiel für diesen anzugeben. Dieses Vorgehen dient der Reduktion der Wahrscheinlichkeit, durch bloßes Raten die korrekte Antwort auszuwählen. Erneut werden die Ergebnisse in Abhängigkeit der Expertise der Teilnehmer sowie der Zuteilung der drei Hilfsmittel differenziert.

*Ergebnisse: Fehlerhäufigkeit bei Modellvergleich*

Für diese Aufgabe zeichnet sich zwischen Experten und Nicht-Experten ein drastischer Unterschied in der Fehlerhäufigkeit ab. Bei jedem der drei Vergleichsaufgaben war selbige bei den Nicht-Experten mehr als doppelt so hoch wie bei den Experten. Ein einheitliches Bild bietet sich dagegen hinsichtlich des Einflusses der drei Hilfsmittel. Sowohl Nicht-Experten als auch Experten erzielten für zwei der drei Vergleiche eine geringere Fehlerrate *mit* verfügbaren Hilfsmitteln. Das bestätigt die Beobachtungen aus der vorherigen Aufgabe und es ist folglich auf dieser Basis kein Unterschied der Nützlichkeit der Hilfsmittel in Abhängigkeit vom jeweiligen Aufgabentyp zu erkennen (G4).

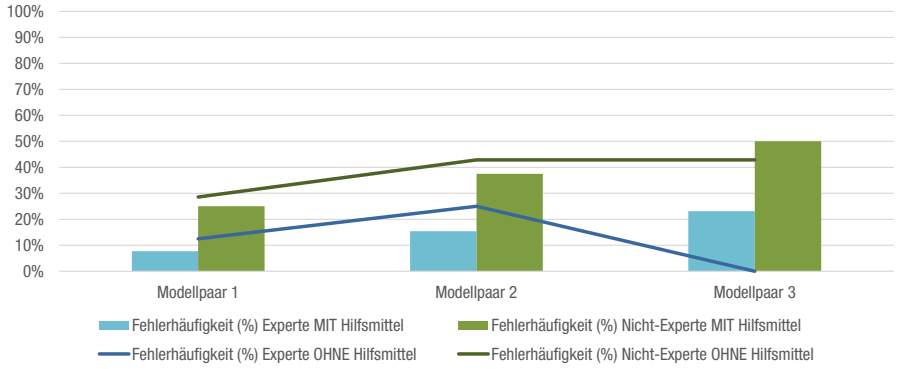



Abbildung 8.14: Fehlerhäufigkeit bei manuellen Modellvergleichen

Beide objektiven Messungen bestätigen die subjektiven Einschätzungen der Teilnehmer, dass einerseits Handlungsbedarf zur Verbesserung der Verständlichkeit regelbasierter Prozessmodelle besteht und die drei Hilfsmittel andererseits einen Beitrag zu dieser Verbesserung leisten können. Diese Erkenntnisse dienen als Grundlage für einige der anschließend abgeleiteten Hypothesen (Abschnitt 8.5.7).

Ziel dieses Abschnitts ist es, den Nutzen der drei Hilfsmittel anhand objektiver Kriterien zu bewerten. Das Ergebnis ist, dass bei der Bewältigung beider Aufgaben tendenziell weniger Fehler auftraten, wenn die drei Hilfsmittel verfügbar waren. Eine Abhängigkeit der Nützlichkeit der Hilfsmittel vom Aufgabentyp konnte nicht beobachtet werden. Dagegen ist eine Tendenz erkennbar, dass die Hilfsmittel für Experten einen größeren Nutzen bei der Bewältigung der beiden Aufgaben haben. Eine differenziertere Betrachtung der Erkenntnisse

### 8.5.7 *Umfrageergebnisse: Extraktion von Hypothesen für die Verwendung der bereitgestellten Mittel*

Auf Basis der vorab diskutierten Untersuchungen werden im aktuellen Abschnitt mehrere Hypothesen abgeleitet.

Alle abgeleiteten Vermutungen beziehen sich auf die Nützlichkeit der drei Hilfsmittel für die Verbesserung des Verständnisses regelbasierter Prozessmodelle. Die dafür abgeleiteten Hypothesen sind: *Hypothesen*

H1 Hinsichtlich des Verständnisses von Prozessregeln in Isolation sowie in Kombination zur Bewältigung von Aufgaben der manuellen Modellkontrolle und des Modellvergleichs besteht Handlungsbedarf. *Handlungsbedarf*

H2 Für die Verbesserung des Verständnisses von Prozessregeln in Isolation ist eine detaillierte, natürlichsprachliche Beschreibung besser geeignet als Positiv- und Negativbeispiele oder eine Übersetzung in eine imperative Form. *Natürlichsprachliche Beschreibungen bei Regeln in Isolation*

H3 Für die Verbesserung des Verständnisses von zu Prozessmodellen kombinierten Prozessregeln sind Positiv- und Negativbeispiele aus praktischer Sicht das attraktivste der drei Werkzeuge. Zu empfehlen ist außerdem die Kombination mit natürlichsprachlichen Beschreibungen. *Positiv- und Negativbeispiele bei Regelkombinationen*

*Translation leistet geringsten Beitrag*

H4 Die Translation eines regelbasierten Prozessmodells in eine imperative Form trägt am wenigsten zu einem besseren Verständnis der Modellsemantik bei.

*Gleiche Nützlichkeit für Modellanalyse und -vergleich*

H5 Die Hilfsmittel sind für eine manuelle Analyse der Semantik eines regelbasierten Prozessmodells und eines Vergleichs desselben mit einem imperativen Modell gleich nützlich.

*Expertenwerkzeuge*

H6 Experten der Prozessmodellierung profitieren von den drei Hilfsmitteln stärker als Nicht-Experten.

Wie bereits eingangs erläutert, dient die durchgeführte Umfrage nicht dazu, die Nützlichkeit der drei im Rahmen der vorliegenden Arbeit entwickelten Hilfsmittel final festzustellen oder zu widerlegen. Aufgrund des Mangels an vergleichbaren Untersuchungen ist das Ziel lediglich eine erste Potentialeinschätzung und das Aufstellen diesbezüglicher Hypothesen. Dies ist konform zur Vorgehensweise explorativer, qualitativer Evaluationen, welche oft zur Schaffung einer Grundlage für spezifischere Untersuchungen durchgeführt werden [30]. Beispielsweise bauen quantitative Methoden häufig auf den Erkenntnissen einer qualitativen Voruntersuchung auf. Die eingeschränkte Zielsetzung der vorliegenden Arbeit auf eine rein qualitative Betrachtung ist weiterhin dadurch begründet, dass bisher keine Erfahrungen bezüglich adäquater Darstellungsformen für Prozessausführungsspuren oder hinsichtlich Inhalt und Struktur natürlichsprachlicher Beschreibungen existieren. Folglich ist ein eventueller, starker Einfluss der Darstellungsform aller Hilfsmittel nicht auszuschließen.

## 8.6 ALLGEMEINE UND SZENARIOBASIERTE ANWENDUNGSEMPFEHLUNGEN

Die bisherigen Untersuchungen geben Aufschluss über Eigenschaften (Abschnitte 8.1 und 8.2) und Korrektheit (Abschnitte 8.3 und 8.4) der drei entwickelten Techniken sowie erste Einblicke in die Nützlichkeit der damit zu erzeugenden Artefakte (Abschnitt 8.5). Dieser Abschnitt greift die grundlegenden Problembereiche (Abschnitt 1.2) auf, welche die vorliegende Arbeit hauptsächlich motivieren und fasst die diesbezüglich diskutierten Anwendungsszenarien sowie zugehörige Anwendungsvoraussetzungen zusammen. Letztere beziehen sich hierbei ausschließlich auf konzeptionelle Charakteristika der Ansätze oder das Szenario selbst und nicht auf technische Einschränkungen. Grund für diese Filterung ist, dass der aktuelle Abschnitt das Ziel verfolgt, eine Entscheidungsgrundlage über die Sinnhaftigkeit einer Anwendung der drei grundlegenden Konzepte zu liefern. In ausführlicherer Form werden alle allgemeinen Anwendungsvoraussetzungen bereits im jeweiligen Konzeptkapitel (Abschnitte 4.2.10, 5.2.5 und 6.2.5) und im Rahmen der Beschreibung der Problembereiche selbst (Abschnitt 1.2) diskutiert. Alle Szenarien basieren auf Aufgaben im Prozesslebenszyklus (Abschnitt 1.1.1), weswegen dieser nachfolgend als Strukturierungshilfe dient. Die für das jeweilige Szenario spezifischen Anwendungsvoraussetzungen werden in den Abschnitten 8.6.2 bis 8.6.4 erläutert.

### 8.6.1 *Allgemeine Anwendungsvoraussetzungen*

Alle drei Ansätze weisen einige Charakteristika auf, die ihre generelle Anwendbarkeit einschränken. Jedes der in den Folgeabschnitten betrachteten Szenarien birgt zusätzlich spezifische Voraussetzungen, ist aber auch von der Erfüllung der im aktuellen Abschnitt skizzierten allgemeinen Bedingungen abhängig. Letztere werden hier lediglich zusammengefasst, da sie bereits an anderer Stelle detaillierter betrachtet werden.

Für die allgemeine Anwendbarkeit des Spurgenerators (Abschnitt 4.2) für multiperspektivische, deklarative Prozessmodelle sind die folgenden Voraussetzungen zu nennen (Abschnitt 4.2.10):

- Es existieren keine historischen Ereignisprotokolle oder es bestehen Zweifel an der Qualität derselben. *Keine historischen Informationen*
- Das Wissen über eine zweckmäßige Konfiguration von Anzahl und Länge der Prozessausführungsspuren ist verfügbar oder kann erworben werden. *Kenntnis einer zweckmäßigen Konfiguration*

Der Ansatz zur Translation von Prozessmodellen (Abschnitt 5.2) ist im Gegensatz zu den beiden anderen Techniken generisch, weswegen Umsetzung und Leistungsfähigkeit vom jeweiligen Sprachpaar abhängig ist. Allgemeine Anwendungsvoraussetzungen für dieses Prinzip lauten wie folgt (Abschnitt 5.2.5):

- Ist ein regelbasiertes Übersetzungssystem verfügbar und ist die Sicherstellung einer ad-hoc korrekten Abbildung des Quellmodells auf ein Zielmodell von essentieller Bedeutung, dann ist von der Verwendung eines induktiven Translationssystems abzusehen. Letzteres ist durch die Nutzung statistischer Verfahren naturgemäß fehlerträchtig und kann dem Anspruch der garantierten Korrektheit der Übersetzung nicht genügen. *Korrektheit: Regelbasierter Übersetzer vor induktivem*
- Ein durch den induktiven Charakter des Prinzips hervorgerufener inhaltlicher Fehler oder ein Informationsverlust im Zielmodell ist entweder akzeptabel oder – was wahrscheinlicher ist – kann durch einen Experten der Prozessmodellierung ausgebessert werden. Ein Informationsverlust, welcher die Folge eines Gefälles in der Ausdrucksmächtigkeit zwischen Quell- und Zielsprache ist, muss ebenfalls akzeptiert werden können. Dies gilt allerdings unabhängig vom konkreten Übersetzungsansatz. *Irrelevanz eines Informationsverlusts*
- Im Falle dessen, dass eine Translation nur transitiv – auf einem Umweg über andere Prozessmodellierungssprachen – möglich ist, sollte keine der Zwischensprachen eine geringere Ausdrucksmächtigkeit aufweisen als die Zielsprache. Ansonsten muss auch hier ein eventueller Informationsverlust berücksichtigt werden. *Transitive Übersetzung: Ausdrucksmächtigkeit einheitlich*
- Die Übersetzung darf ganzheitlich – nicht-inkrementell – erfolgen und somit existierende Übersetzungsergebnisse ignorieren. *Nicht-inkrementelle Übersetzung*
- Abhängigkeiten des Prozessverlaufs von konkreten Datenwerten und der Erfüllung von Bedingungen an definierten Entscheidungspunkten müssen nicht rekonstruiert werden. *Datenwerte und Bedingungen problematisch*

Für die Anwendung der Technik zur Generierung natürlichsprachlicher Prozessbeschreibungen auf Basis von multi-perspektivischen, deklarativen Prozessmodellen müssen die folgenden Voraussetzungen gelten:

*Verlust der Eindeutigkeit*

- Ein Verlust der Eindeutigkeit formaler Sprachen ist akzeptabel. So muss beispielsweise die Identität von Objekten anhand linguistischer Referenzen eigenständig nachvollzogen werden (Abschnitt 6.2.5).

*Geringe Bedeutung sprachlicher Flexibilität*

- Die Vollständigkeit der natürlichsprachlichen Prozessbeschreibung ist wichtiger als sprachliche Variabilität und Vielfalt (Abschnitt 6.2.5).

*Modellgröße und -komplexität*

- Das Prozessmodell überschreitet eine individuell zu bestimmende Größe respektive Komplexität oder die manuelle Anfertigung einer textuellen Beschreibung ist aus einem anderen Grund unerwünscht (Abschnitt 3.6.1).

Der Liste dieser Anwendungsvoraussetzungen aller drei Techniken sind für eine praktische Anwendung auch technische Bedingungen hinzuzufügen. Diese werden in den zugehörigen Konzeptkapiteln (Abschnitt 4.2.10, Abschnitt 5.2.5 und Abschnitt 6.2.5) betrachtet.

### 8.6.2 *Szenario 1 (Design, Implementierung, Ausführung): Verstehen eines Prozessmodells*

Sowohl in der Design- als auch in der Implementierungs- und der Ausführungsphase des Prozesslebenszyklus werden Prozessmodelle manuell interpretiert. Die Verständlichkeit desselben ist daher von essentieller Bedeutung (Abschnitt 1.2.1). Das erste Szenario ist daher, dass ein menschlicher Nutzer die Inhalte eines Prozessmodells erfassen muss, ihm aber entweder die verwendete Modellierungssprache nicht geläufig ist oder das Modell in selbiger nicht verständlich genug dargestellt werden kann. Prinzipiell können hier alle drei Ansätze zur Verbesserung des Modellverständnisses beitragen (Tabelle 1.1). Im Folgenden wird dies anhand von Anwendungsvoraussetzungen differenzierter betrachtet.

Der Spurgenerator für deklarative, multi-perspektivische Prozessmodelle (Abschnitt 4.2) kann zur Erzeugung von Positiv- und Gegenbeispielen für respektive gegen valide Prozessausführungsspuren verwendet werden. Dies ist unter folgenden Voraussetzungen sinnvoll:

*Konkrete Fragestellung*

- Es ist zu empfehlen, dass der Spurgenerator zur Beantwortung einer konkreten Fragestellung verwendet wird. Die dafür notwendige Informationsgrundlage muss im Modell kodiert sein. Grund für diese Empfehlung ist, dass Ereignisprotokolle selbst bei geringer Modellgröße schnell sehr umfangreich und unübersichtlich werden können. Eine konkrete Fragestellung schränkt diese Anzahl präventiv ein.

*Konfiguration kann bestimmt werden*

- Basierend auf dem eben genannten Kriterium muss der Nutzer zudem in der Lage sein, eine geeignete Konfiguration für Anzahl und Länge der Prozessausführungsspuren zu bestimmen, bei der sich die Datengrundlage für die Beantwortung der Fragestellung manifestiert.

Das im Rahmen dieser Arbeit entwickelte Translationsprinzip (Abschnitt 5.2) ist in der Lage, Prozessmodelle in eine alternative Prozessmodellierungssprache zu übersetzen. Die Nutzung dieser Technik zur Verbesserung des Modellverständnisses unterliegt einigen Bedingungen:

*Nachvollziehbarkeit der Translation unnötig*

- Zum besseren Verständnis des Quellmodells wird die Nachvollziehbarkeit der Translation durch Korrespondenzen zwischen Quell- und Zielmodell-Elementen nicht benötigt. Durch die Verwendung von Ereignisprotokollen als Transfermedium können keine solchen direkten Korrespondenzen erzeugt werden.

*Deklarative Zielsprache*

- Die Übersetzung in eine deklarative Modellierungssprache ist dann zu empfehlen [61, 62], wenn
  a. der Nutzer mit der deklarativen Zielsprache vertraut ist,
  b. das Quellmodell zahlreiche alternative Ausführungspfade gestattet und
  c. situationsbezogene Informationen hervorgehoben und gleichzeitig konkrete Ausführungspfade ausgeblendet werden sollen.

*Imperative Zielsprache*

- Umgekehrt ist die Übersetzung in eine imperative Sprache zu empfehlen [61, 62], wenn
  a. der Nutzer mit der imperativen Zielsprache vertraut ist,
  b. das Quellmodell wenige alternative Ausführungspfade gestattet und
  c. ablaufzentrierte Informationen hervorgehoben werden sollen.

*Didaktischer Nutzen einer niedrigeren Ausdrucksmächtigkeit*

- Bei einem Gefälle hinsichtlich der Ausdrucksmächtigkeit zwischen Quell- und Zielsprache kann ein Informationsverlust entstehen. E in Modell in der Zielsprache kann aber immer erzeugt werden. Diese Eigenschaft des Prinzips kann didaktisch wertvoll sein. Beispielsweise lassen sich so die Auswirkungen einer Beziehung zwischen Datenproduzent und -konsument auf den Kontrollfluss verdeutlichen.

Prozessmodelle werden mittels formaler Notationen repräsentiert, was zu einer Sprachbarriere führen kann. Eine natürlichsprachliche Prozessbeschreibung hebt diese Sprachbarriere auf. Eine Anwendung des ebenfalls in dieser Arbeit entwickelte NLG-System (Abschnitt 6.2) für multi-perspektivische, deklarative Prozessmodell ist unter den nachfolgenden Bedingungen sinnvoll:

*Aussagekräftige Bezeichner für Aktivitäten*

- Für die Aktivitäten im Prozessmodell werden aussagekräftige Bezeichner verwendet, welche eine Tätigkeit beschreiben. Beispielsweise Akronyme können die Lesbarkeit eines Textes sonst stark negativ beeinflussen.

*Keine Variation des Abstraktionsniveaus notwendig*

- Gegenüber einer manuell angefertigten natürlichsprachlichen Beschreibung ist es mit der NLG-Technik nicht möglich, das Abstraktionsniveau der Erklärungen zu variieren. Aus diesem Grund muss das angestrebte Niveau der Beschreibungen die exakte Erläuterung aller Modellinhalte sein.

In diesem Abschnitt sind lediglich Grundvoraussetzungen für eine Anwendung der drei Techniken aufgeführt, ohne Informationen über eine kognitive Komponente zu berücksichtigen. Die im Rahmen dieser Arbeit durchgeführte Umfrage (Abschnitt 8.5) bildet lediglich den Auftakt für eine extensive Untersuchung der Nützlichkeit der von den drei Techniken produzierten Artefakte. Besonders zwei der im Rahmen dieser Umfrage generierten Hypothesen sind an dieser Stelle interessant. So zeigt die Auswertung der Umfrage eine Tendenz, dass zur Verbesserung des Verständnisses einzelner Regeln deklarativer Prozessmodelle eine detaillierte natürlichsprachliche Beschreibung von den Nutzern präferiert wird. Im Gegensatz dazu bieten Beispiele für valide respektive nicht-valide Prozessausführungsspuren tendenziell die beste Unterstützung für die Verbesserung des Verständnisses von deklarativen Prozessregeln in Kombination. Diese Hypothesen bilden allerdings primär einen Ausgangspunkt für sich anschließende Forschungsarbeiten und können in Konsequenz dazu nicht als finale Empfehlungen angesehen werden.

### 8.6.3 *Szenario 2 (Design): Vergleich verschiedensprachiger Prozessmodelle*

Besonders im Bereich von Prozessrepositorien, deren Inhalte als Modellierungsvorlage dienen können, spiegelt sich die Notwendigkeit von Vergleichsmethoden für Prozessmodelle wider. Die meisten existierenden Methoden basieren allerdings auf der Annahme, dass die zu vergleichenden Prozessmodelle in derselben Modellierungssprache vorliegen. Dies ist jedoch nicht immer der Fall (Abschnitt 1.2.4). Dementsprechend ist das zweite Szenario ein Vergleich zweier verschiedensprachiger Prozessmodelle durch einen Nutzer. Auch hier können unter nachfolgend beschriebenen Voraussetzungen prinzipiell alle drei in dieser Arbeit vorgestellten Techniken zum Einsatz kommen (Tabelle 1.1).

Der Spurgenerator (Abschnitt 4.2 ist in der Lage, eine exemplarische Vergleichsbasis in Form von Positiv- und Gegenbeispielen für respektive gegen valide Prozessausführungsspuren zu schaffen. Dies ist unter den folgenden Voraussetzungen sinnvoll:

*Nur Vergleich der Ausführungssemantik*

- Im Fokus steht der Vergleich der Ausführungssemantik der jeweiligen Modelle. Die Charakterisierung derselben durch Existenz und zeitlicher Reihenfolge von Aktivitäten sowie deren Assoziation mit anderen Prozessentitäten ist zur Beschreibung derselben ausreichend. Strukturelle Unterschiede der Prozessmodelle sind nicht von Interesse.

*Vollständige Ereignisprotokolle*

- Der Spurgenerator ist in der Lage, für eine konfigurierte Anzahl und Länge der Prozessausführungsspuren die Vollständigkeit der erzeugten Ereignisprotokolle zu garantieren[12]. Gilt dies für den Spurgenerator des zu vergleichenden Modells nicht, ist die Vollständigkeit der damit erzeugten Ereignisprotokolle nachträglich sicherzustellen.

*Kein Vergleich von Datenwerten*

- Ein Vergleich möglicher Wertebelegungen für Prozessvariablen wird nicht angestrebt. Da sich diese im kontinuierlichen Spektrum bewegen können, ist ein Vergleich auf einer diskreten, exemplarischen Basis nicht möglich.

12 Eine Ausnahme bildet die Belegung modellierter Prozessvariablen (Abschnitt 8.6.1).

Mittels des generischen Translationsprinzips (Abschnitt 5.2) können verschiedensprachige Prozessmodelle in eine gemeinsame Modellierungssprache übersetzt werden, was den Vergleich der Modelle erleichtern kann. Hierfür müssen die folgenden Voraussetzungen erfüllt sein:

*Ignorieren syntaktischer Unterschiede*

- Vertreter der Process-Mining-Komponente des Verfahrens neigen oft dazu, Prozessmodelle von unnötig großer Komplexität zu erzeugen. Daher muss das zu verwendende Vergleichsverfahren in der Lage sein, äquivalente syntaktische Konstruktionen im Modell zu erkennen. Beispielsweise können Inklusiv-Oder-Konstruktionen in BPMN durch ein OR-Gateway oder durch eine Kombination aus XOR- und AND-Gateways ausgedrückt werden. Der Vergleich muss sich folglich auf die Ausführungssemantik der Modelle konzentrieren.

*Vergleich mit perspektivischer Einschränkung*

- Ein Vergleich von Modellen, die in Sprachen unterschiedlicher Ausdrucksmächtigkeit repräsentiert sind, ist immer problematisch. Möchte man dennoch solche Modelle vergleichen, dann sollte die Zielsprache der Translation von allen verwendeten Sprachen die geringste Ausdrucksmächtigkeit aufweisen. Dies mag zunächst paradox erscheinen. Allerdings ist das induktive Translationsprinzip beispielsweise dazu in der Lage, Beziehungen zwischen Datenproduzenten und -konsumenten auf reine Kontrollflusskonstrukte abzubilden. Zwar zieht dies den Verlust der inhaltlichen Rechtfertigung für die feste Reihenfolge der jeweiligen Aktivitäten nach sich, aber der ausführungssemantische Effekt bleibt erhalten.

*NLG-Technik bei inhaltlichem Vergleich der Modellsemantik*

Abseits eines automatischen Vergleichs von Prozessmodellen kann die in dieser Arbeit vorgestellte NLG-Technik die Grundlage liefern, um einen solchen Vergleich manuell durchzuführen. Dies ist besonders dann sinnvoll, wenn die zur Verfügung stehenden Vergleichsmittel unzureichend oder nicht existent sind. Besonders bei der Interpretation der Bezeichner aller Entitäten eines Prozesses scheitern viele automatisierte Vergleichsverfahren. Ein manueller Vergleich auf Basis einer natürlichsprachlichen Prozessbeschreibung kann hier Abhilfe schaffen.

### 8.6.4 *Szenario 3 (Implementierung): Beseitigung einer technischen Sprachbarriere zwischen konzeptuellem Prozessmodell und Prozessausführungssystem*

Aus verschiedenen Gründen kann es in der Implementierungsphase dazu kommen, dass ein Prozessmodell und das Zielsystem für die Prozessausführung sprachlich inkompatibel sind. Diese Situation tritt beispielsweise dann auf, wenn ein Wechsel der konzeptuellen Prozessmodellierungssprache stattgefunden hat oder ein operatives Prozessmodell auf eine höhere Sprachversion migriert wurde (Abschnitt 1.2.2). Das Szenario ist demnach, dass ein konzeptuelles Prozessmodell existiert, welches auf einem spezifischen Ausführungssystem implementiert werden soll, dieses jedoch die verwendete Prozessmodellierungssprache nicht unterstützt. Von den drei in dieser Arbeit vorgestellten Ansätzen findet hier nur das generische Translationsprinzip direkt Anwendung (Tabelle 1.1).

Im aktuellen Kontext gelten für das Translationsprinzip folgende Anwendungsbedingungen:

*Nachbearbeitung: Datenabhängigkeiten, Entscheidungspunkte*

- Modellierte Beziehungen, an denen konkrete Datenwerte beteiligt sind und auch Bedingungen an definierten Entscheidungspunkten müssen manuell nachgearbeitet werden.

*Nur syntaktische Migration*

- Im Falle der Verwendung des Translationsprinzips zur Migration eines Prozessmodells auf eine andere Sprachversion mit dem Ziel der Wiederherstellung der Kompatibilität mit einem Prozessausführungssystem dürfen die zu migrierenden Änderungen ausschließlich syntaktischer Natur sein.

Die ebenfalls in dieser Arbeit vorgestellten Techniken zur Spurgenerierung und zur Erzeugung natürlichsprachlicher Prozessbeschreibungen können indirekt ebenfalls genutzt werden. Dies ist vor allem dann der Fall, wenn keine automatische Technik zur Translation eines Prozessmodells in eine vom Ausführungssystem unterstützte Sprache zur Verfügung steht. In diesem Fall muss die Translation manuell vorgenommen werden. Mittels des Spurgenerators erzeugte valide und nicht-valide Prozessausführungsspuren können dann verwendet werden, um die Konformität des so erzeugten Modells mit dem Ausführungsverhalten des Quellmodells automatisiert zu prüfen. Eine solche Prüfung kann auch manuell erfolgen, wobei eine natürlichsprachliche Beschreibung des Prozesses Unterstützung bieten kann. Gleichzeitig kann ein solcher Text auch als Vorlage für die manuelle Übersetzung selbst verwendet werden. Ist die vom Ausführungssystem unterstützte Sprache imperativ, dann können auch valide Prozessausführungsspuren als Modellierungsvorlage dienen.

# 9 ZUSAMMENFASSUNG UND AUSBLICK

Dieses letzte Kapitel greift die eingangs dieser Arbeit identifizierten Problembereiche auf (Abschnitt 1.2), welche durch die große Vielfalt von teilweise sehr unterschiedlichen Prozessmodellierungssprachen entstehen. Darauf aufbauend werden die für die vorliegende Arbeit grundlegenden Teilprobleme sowie die zugehörigen Lösungsansätze zusammengefasst.

## 9.1 ZUSAMMENFASSUNG

*Problemstellung*

Die Notwendigkeit der Dokumentation und Steuerung von Geschäftsprozessen in größeren Unternehmen hat eine Vielzahl unterschiedlicher Modellierungssprachen hervorgebracht. Davon erfreuen sich einige einer größeren Verbreitung, wohingegen andere nur in Ausnahmefällen Anwendung finden. Allerdings zeigt das kürzliche Aufkommen deklarativer Prozessmodellierungssprachen deutlich, dass selbst etablierte Sprachen nicht für jeden Prozesstyp geeignet sind. Besonders im Bereich der unternehmensübergreifenden Prozesse sind Probleme im Bereich der Sprach- und Werkzeuginteroperabilität die Folge. Auch für menschliche Beteiligte stellt es eine Herausforderung dar, jede relevante Sprache zu erlernen. Es existieren also zusätzlich Verständlichkeitsprobleme. Beide Probleme werden noch verstärkt, da sich Prozessmodellierungssprachen auch weiterentwickeln. Das erfordert die Migration existierender Modelle und ein neuerliches Erlernen der möglicherweise veränderten Syntax. Orthogonal dazu stellt sich auch die Frage der Vergleichbarkeit von Prozessmodellen, die im Bereich von Prozessrepositorien und der Fusion von Unternehmen von entscheidender Bedeutung ist. Es besteht die Gefahr, dass die Eignung einer Prozessmodellierungssprache anhand der eben genannten Probleme bewertet wird – und nicht anhand des Modellierungsproblems selbst.

*Vier Anforderungen*

Ziel der vorliegenden Arbeit ist es daher, Mittel zu identifizieren, um die Wahlfreiheit und die dafür notwendige Akzeptanz von Prozessmodellierungssprachen zu steigern. Dazu werden vier Anforderungen identifiziert, welche die Untersuchungsgrundlage der vorliegenden Arbeit darstellt. Diese beinhalten die Schaffung flexibler Möglichkeiten der Übersetzung von Prozessmodellen in andere Sprachen oder in einen natürlichsprachlichen Beschreibungstext. Eine weitere Anforderung ist die Möglichkeit, auf Wunsch gültige und nicht-gültige Beispielinstanzen aus einem gegebenen Prozessmodell abzuleiten. Eine letzte Anforderung ist die Bereitstellung abwärtskompatibler, automatisierter Migrationsmechanismen.

*Status Quo: Translation von Prozessmodellen*

Existierende Translationstechniken sind meist sprachspezifisch und fokussieren sich primär auf den Bereich der imperativen Prozessmodellierungssprachen. Übersetzungen zwischen imperativen und deklarativen Sprachen werden somit nur unzureichend unterstützt. Um die diesbezügliche Abdeckung von Translationssystemen zu vergrößern, müssten nach dem gängigen Modell-zu-Modell-Transformationsprinzip für jedes Sprachpaar eine Vielzahl an explizit formulierten Abbildungsregeln

definiert werden. Dies stellt nicht nur einen signifikanten technischen Aufwand dar, sondern birgt auch große Herausforderungen, da sich besonders imperative und deklarative Prozessmodellierungssprachen hinsichtlich der Modellinhalte drastisch unterscheiden. In der vorliegenden Arbeit wird aus diesem Grund ein *induktives* Translationsprinzip vorgestellt, welches existierende Techniken und Technologien aus dem Bereich der Prozesssimulation und des Process Mining wiederverwendet. Von Simulatoren künstlich erzeugte Prozessausführungsprotokolle dienen dabei als sprachneutrales Transfermedium. Process-Mining-Techniken rekonstruieren daraus ein Modell in der jeweiligen Zielsprache. Der Weg eines induktiven Verfahrens annulliert sowohl den technischen Aufwand der Definition von Übersetzungsregeln als auch die konzeptionelle Kluft zwischen imperativen und deklarativen Prozessmodellen.

*Beitrag: Induktives Translationsprinzip*

Aktuell existiert lediglich ein Ansatz, welcher aus Prozessmodellen automatisiert natürlichsprachliche Prozessbeschreibungen ableitet. Dieser ist jedoch auf blockstrukturierte BPMN-Modelle beschränkt und ist somit paradigmen- und sogar sprachspezifisch. Im Rahmen der vorliegenden Arbeit wird daher ein Ansatz zur Generierung natürlichsprachlicher Beschreibungen für *deklarative* Prozessmodelle untersucht. Dieser macht sich computerlinguistische Standard-Prinzipien zur Strukturierung von Sätzen und Dokumenten zunutze. Im Gegensatz zu imperativen Modellen weisen deklarative Modelle keine native Leserichtung auf, weswegen die Hauptschwierigkeit in der Ableitung einer adäquaten Textstruktur von zentraler Bedeutung ist. Weiterhin sollte die Generierung natürlichsprachlicher Prozessbeschreibungen nicht ohne Kontextbezug betrachtet werden. Aus diesem Grund wird auch die Unterscheidung zwischen der Generierung zum Modellierungs- und der zum Ausführungszeitpunkt des Prozesses unterschieden. Gleichzeitig werden auf Basis eines Nutzermodells Rückschlüsse auf das Verständnis verbessernde Modifikationen des Zieltextes gezogen. Diese Flexibilität begründet sich auf dem linguistischen Fundament des Textgenerierungsansatzes, welches die konkrete Formulierung eines Sachverhalts von ihrer semantischen Repräsentation trennt.

*Status Quo: NLG*

*Beitrag: Textgenerierung für deklarative Prozessmodelle*

Spezielle Formen von Simulationstechniken – sogenannte Spurgeneratoren – sind in der Lage, für ein gegebenes Prozessmodell valide, künstliche Beispielinstanzen abzuleiten, die Ausführungsspuren genannt werden. Nur wenige bieten zusätzlich die Möglichkeit, Negativbeispiele, also nicht-valide Beispielinstanzen, zu generieren. Zudem sind, mit einer Ausnahme, alle Spurgeneratoren auf den imperativen Sprachraum begrenzt. Der einzige, existierende Spurgenerator für deklarative Prozessmodelle basiert auf der kontrollflussorientierten Sprache Declare. Informationen über beispielsweise die Assoziation organisatorischer Ressourcen sowie die Manifestation von Datenzugriffen und -werten sind daher nicht Teil der erzeugten Beispielinstanzen. Die im Rahmen der vorliegenden Arbeit entwickelte Spurgenerator-Technik basiert auf der *multi-perspektivischen*, deklarativen Prozessmodellierungssprache DPIL und ist in der Lage, eben diese Aspekte bei der Simulation zu berücksichtigen und anschließend Informationen darüber bereitzustellen. Da das Fundament dieses Ansatzes die Logiksprache Alloy ist, entspricht das Simulationsvorhaben einem Erfüllbarkeitsproblem, welches Alloy durch eine Instanzsuche in einem beschränkten Lösungsraum bewältigt. Dies gestattet zudem auch eine weitreichende Kontrolle über die Fokussierung der Simulationsdurch-

*Status Quo: Spurgenerierung*

*Beitrag: Multi-perspektivische, deklarative Spurgenerierung*

führung. Beispielsweise können so Vorgaben für partielle Ausführungsspuren berücksichtigt werden, zu denen alle übrigen Fortsetzungen ermittelt werden sollen.

*Status Quo: Sprachevolution*

Modellmigration aufgrund der evolutionären Weiterentwicklung von Prozessmodellierungssprachen kann als Spezialisierung des Problems der Translation aufgefasst werden. Während im Bereich imperativer Prozessmodellierungssprachen immerhin sporadisch Ansätze zur Migration von Prozessmodellen vorhanden sind, fehlen diese im Bereich deklarativer Sprachen gänzlich. Betrachtet man die zwei fraglichen Sprachversionen als verschiedene Sprachen, dann kann ein generisches Übersetzungsprinzip auch hier Anwendung finden. Die Migration von Prozessmodellen wird in der vorliegenden Arbeit nicht explizit untersucht. Der generische Charakter des vorgestellten Translationsansatzes erübrigt jedoch die Entwicklung von Migrationstechniken genau dann, wenn für jede Sprachversion jeweils ein Spurgenerator und eine Process-Mining-Technik zur Verfügung stehen.

*Beitrag: Wiederverwendung des generischen Translationsprinzips*

*Implementierung*

Alle vorab genannten Konzepte sind prototypisch implementiert. Das generische Translationsprinzip basiert dabei auf Erweiterungen der Data-Mining-Plattform RapidMiner, welche unter anderem Schnittstellen für die Interaktion zwischen Spurgenerator und Process-Mining-Werkzeugen bereitstellt. Auch die Vor- und Weiterverarbeitung von Quell- und Zielmodell sowie den künstlich erzeugten Ereignisprotokollen wird durch RapidMiner durch eine flexible Infrastruktur unterstützt. Da der im Rahmen der vorliegenden Arbeit entwickelte Spurgenerator ebenfalls als Komponente der Translationstechnik zum Einsatz kommen sollte, ist er ebenfalls als RapidMiner-Erweiterung realisiert. Die Technik zur Generierung natürlichsprachlicher Beschreibungen basiert dagegen auf der Acceleo-Implementierung des OMG-Standards Mof2Text.

Ein Teil der Überprüfung der Konzepte ist mit der Bereitstellung eines Softwareprototypen erbracht worden. Zusätzlich werden statische Eigenschaften der einzelnen Techniken analysiert. Daran schlossen sich Untersuchungen an, welche die korrekte Arbeitsweise der Technologien sicherstellen soll. Den Abschluss des Evaluationsteils bildete die Durchführung einer Umfrage zur Nützlichkeit der durch die einzelnen Techniken erzeugten Artefakte für die Verbesserung des Verständnisses deklarativer Prozessmodelle.

*Evaluation: Eigenschaften des Translationsprinzips*

Die untersuchten Eigenschaften des vorgestellten Translationsprinzips enthält eine Analyse der Verbesserung der Abdeckung von Übersetzungsmöglichkeiten durch die Verfügbarkeit von Spurgeneratoren und Process-Mining-Techniken. Für lediglich 10% der möglichen Paarungen der gängigsten imperativen und sowie der derzeit verfügbaren deklarativen Prozessmodellierungssprachen existierten direkte Übersetzungsmöglichkeiten. Berücksichtigt man auch alle sich daraus ergebenden transitiven Übersetzungssysteme, verdoppelt sich dieser Prozentsatz. Es zeigte sich zudem, dass das Gros dieser Übersetzungsmöglichkeiten lediglich imperativ-imperative Sprachpaarungen abdecken. Ergänzt man alle Übersetzungsmöglichkeiten, die durch das induktive Translationsprinzip hinzukommen, erhöht sich der Anteil direkter Übersetzungsmöglichkeiten auf 25%. Inklusive transitiver Möglichkeiten steigt die Abdeckung sogar auf 43%. Dabei ist besonders die Abdeckung von Sprachpaarungen mit Beteiligung deklarativer Prozessmodellierungssprachen gestiegen.

Jedes induktive Verfahren – und so auch das vorgestellte Translationsverfahren – hängt von der Vollständigkeit der verfügbaren Beispiele ab. Durch die Verwendung eigens generierter, künstlicher Ereignisprotokolle als Transfermedium kann auf diesen Aspekt Einfluss genommen werden. Da sowohl bei der Simulation als auch beim Process-Mining statistische Verfahren zur Anwendung kommen, ist eine korrekte Übersetzung nicht zu garantieren. Aus diesem Grund werden mehrere Versuche anhand künstlicher und aus der Praxis bezogener Beispiele durchgeführt. Diese lassen eine Tendenz erkennen, dass mit hinreichend großer Datengrundlage ein annähernd oder vollständig korrektes Ergebnis erzielt werden kann. Im Umkehrschluss bestätigt dies aber die Erwartung, dass keine Garantie für die Korrektheit abgegeben werden kann. Zudem sind die Untersuchungen auf wenige Sprachpaarungen beschränkt, da eine Untersuchung aller möglichen Paarungen die Grenzen des vertretbaren Umfangs der vorliegenden Arbeit deutlich überschreiten würde. Folglich ist das vorgestellte Verfahren nur in Kombination mit einer sorgfältigen Prüfung des Translationserfolgs zu verwenden. Das schließt im allgemeinen Fall den Einsatz in vollautomatisierten Arbeitsprozessen aus.

*Evaluation: Korrektheit des Translationsprinzips*

Zur Überprüfung der korrekten Arbeitsweise des im Rahmen der vorliegenden Arbeit entwickelte Spurgenerators werden mittels der Logiksprache Alloy Annahmen modelliert, die als Testspezifikationen dienen. Diese beinhalten im Wesentlichen Prüfbedingungen, die bei einer korrekten Überführung der Semantik einer DPIL-Regel in die Sprache Alloy erfüllt sein müssen. Durch die Formulierung der Testspezifikationen in Alloy, kann der gleiche Mechanismus zur Testdurchführung verwendet werden, der auch für die Konstruktion von Negativbeispielen valider Prozessausführungen verwendet wird. Ist dieser Mechanismus nicht in der Lage, für eine gewählte Größe des Lösungsraumes Negativbeispiele für die modellierten Annahmen zu identifizieren, dann ist damit zwar die Korrektheit der Transformation nicht bewiesen, aber positiv getestet worden. Mit allen derzeit unterstützten DPIL-Regelschablonen wird ein solcher Test durchgeführt – mit ausschließlich positivem Ergebnis. Die Korrektheit der Transformation allein ist keine Garantie für die korrekte Funktionsweise der Spurgenerierung selbst. Da der Spurgenerator jedoch auf dem Alloy-Rahmenwerk basiert und Ausführungsspuren mittels Instanzsuche ermittelt, ist dies eine Frage der Korrektheit des Alloy-internen Lösungsalgorithmus für Erfüllbarkeitsprobleme. Für die vorliegende Arbeit wird von der Korrektheit dieses Algorithmus ausgegangen; praktische Tests bestätigten diese Annahme.

*Evaluation: Korrektheit des Spurgenerators*

Das letzte Artefakt, welches im Zuge dieser Arbeit entstanden ist, ist das System zur Generierung natürlichsprachlicher Beschreibungstexte für Prozessmodelle. Da es sich bei diesem System um einen exemplarischen Durchstich handelt, war und ist eine umfangreiche Evaluation der Korrektheit nicht möglich. Im Rahmen der vorliegenden Arbeit werden hier verschiedene, leicht abgewandelte Versionen des Beispiel-DPIL-Modells aus Codeausschnitt 6.1 verwendet, um die Abbildung auf einen syntaktisch, orthografisch und semantisch kohärenten Text exemplarisch zu prüfen. Für den sehr begrenzten Variantenreichtum der besagte DPIL-Modelle können in diesem Sinne korrekte natürlichsprachliche Prozessbeschreibungen abgeleitet werden.

*Evaluation: Eingeschränkte Prüfung der Korrektheit des NLG-Ansatzes*

Für eine erste Einschätzung der Nützlichkeit aller drei Artefakte hinsichtlich ihres Potentials zur Verbesserung des Verständnisses regelbasierter Prozessmodel-

*Evaluation: Qualitative Umfrage*

le dient eine qualitative Umfrageevaluation mit Probanden aus unterschiedlichen Tätigkeitsbereichen. Selbige identifizierte zunächst einen tendenziellen Handlungsbedarf; die Verständlichkeit regelbasierter Prozessdarstellungen ist sogar für Experten in vielen Fällen nicht optimal. Auf Basis subjektiver Einschätzungen sollten die Teilnehmer dann den Verständnisgewinn durch jedes der drei Hilfsmittel bewerten. Dabei stechen besonders die natürlichsprachlichen Beschreibungen der Prozessregeln sowie Positiv- und Negativ-Beispiele für Prozessausführungsspuren positiv hervor. In einem Vergleich der Fehlerhäufigkeiten unter Verwendung der respektive unter Verzicht auf die besagten Hilfsmittel wird ebenfalls ein tendenzieller Nutzen festgestellt. Dieser ist nach Experten und Nicht-Experten der Prozessmodellierung differenziert in mehreren Hypothesen formuliert.

Abschließend soll die in Abschnitt 2.2 aufgestellte Forschungsfrage beantwortet werden. Diese beschäftigt sich mit der Problematik, welche Hilfsmittel zu einer Verbesserung der Flexibilität bei der Wahl geeigneter Prozessmodellierungssprachen beitragen können. Diese Frage beinhaltet einerseits die Teilfrage, inwiefern solche Hilfsmittel zur Verfügung stehen und andererseits, zu welchem Grad diese zur Lösung des Flexibilitätsproblems beitragen. Der Fokus der vorliegenden Arbeit liegt hier klar auf der erstgenannten Teilfrage. Die Beantwortung dieser Frage wird mit einer Verfügbarkeitsanalyse für Translations- und Spurgenerierungstechniken sowie Techniken zur Generierung natürlichsprachlicher Beschreibungstexte für Prozessmodelle begonnen. Geeignete konzeptionelle Erweiterungen in den drei Bereichen und deren Umsetzungen bilden die Kernbeiträge der vorliegenden Arbeit. Mit einer ersten Evaluation der drei entwickelten Konzepte kann nun auch der erste Teil der Forschungsfrage mit „ja“ beantwortet werden, wobei die oben genannten Evaluationsergebnisse eine differenziertere Betrachtung erfordern. Die Teilfrage nach der Größe des Beitrags der einzelnen Hilfsmittel kann im Rahmen der vorliegenden Arbeit nicht vollständig beantwortet werden. Der Grund ist, besonders im Hinblick auf das Translationssystem, die schiere Vielfalt verfügbarer Prozessmodellierungssprachen. Durch die durchgeführte Umfrage ist jedoch eine Tendenz erkennbar, dass die durch die drei Techniken erzeugten Artefakttypen primär für Experten der Prozessmodellierung aber auch für Nicht-Experten bei der manuellen Interpretation von Modellen zu Kontroll- und Vergleichszwecken tatsächlich eine Hilfe darstellen.

*Beantwortung der Forschungsfrage*

## 9.2 ZUKÜNFTIGE FORSCHUNGSTHEMEN

In der vorliegenden Arbeit werden vier grundlegende Problemfelder identifiziert, aus denen die Notwendigkeit der Entwicklung eines Spurgenerators, eines flexiblen Translationssystems sowie einer Technik zur Generierung natürlichsprachlicher Texte für Prozessmodelle abgeleitet wird. Im Rahmen dieser Entwicklung und der ersten Erfahrungen in der Verwendung der Techniken ergaben sich einige mögliche Anknüpfungspunkte für sich anschließende Untersuchungen. Der aktuelle Abschnitt bietet daher eine Übersicht über einige zukünftige Forschungsthemen.

*Forschung: Spurgenerator*

In Abschnitt 4.2 wird eine Technik zur Generierung künstlicher Prozessausführungsspuren für multi-perspektivische, deklarative Prozessmodelle vorgestellt. Aus

der Arbeit an und mit dieser Technik ergeben sich folgende Themen für fortführende Forschungsarbeiten:

*Flexible Abbildung auf Alloy*

- Zum aktuellen Zeitpunkt ist eine Grundvoraussetzung für die Verwendbarkeit des im Rahmen dieser Arbeit entwickelten Spurgenerators, dass das Eingabe-DPIL-Modell ausschließlich mittels einer fest vorgegebenen Menge an Regelschablonen definiert ist. Da sowohl DPIL als auch die verwendete Simulationssprache Alloy auf der Prädikatenlogik erster Stufe basiert, ist es naheliegend, zu untersuchen, ob atomare DPIL-Ausdrücke auf Alloy-Konstrukte abgebildet werden können.

*Spurgenerator als Ausführungskomponente*

- Der Spurgenerator wird unter anderem hinsichtlich seiner Befähigung zur Erzeugung *künstlicher* Prozessausführungsspuren auf Basis gegebener partieller Spuren untersucht. Auf den daraus gewonnenen Erkenntnissen aufbauend können Untersuchungen der Eignung des Spurgenerators als Ausführungskomponente eines Workflow-Management-Systems durchgeführt werden. Ein wesentlicher Vorteil des vorgestellten Spurgenerators gegenüber konventionellen Ausführungskomponenten ist die Fähigkeit, Auswirkungen aktueller Ereignisse auf zukünftige Ereignisse berechnen zu können. Das wird üblicherweise als Vorausschau-Technik (engl. *look-ahead technique*) bezeichnet.

*Darstellung valider/invalider Prozessausführungsspuren*

- Eine Motivation für die Entwicklung eines Spurgenerators ist die Verwendung desselben als Hilfsmittel für die Verbesserung des Verständnisses multi-perspektivischer, deklarativer Prozessmodelle. In dieser Funktion wird eine geeignete Darstellung der generierten Beispiele und Gegenbeispiele für oder gegen valide Prozessausführungssuren benötigt. Die Suche nach einer solchen adäquaten Darstellung sollte iterativ gestaltet werden, indem nach einem ersten Entwurf das Verständnis der Darstellung selbst sowie ihre Verwendung für die Verbesserung des Verständnisses eines Prozessmodells geprüft wird.

*Forschung: Translation*

Im zweiten Konzeptabschnitt (Abschnitt 5.2) wird ein induktives Prinzip zur Übersetzung von Prozessmodellen in alternative Prozessmodellierungssprachen diskutiert. Dieses basiert auf der losen Kopplung von Spurgeneratoren und Process-Mining-Techniken, deren Zusammenarbeit auf der Grundlage künstlicher Ereignisprotokolle stattfindet. Auch im Bereich dieses Prinzips lassen sich einige potentielle Anschlussthemen ableiten:

*Untersuchung weiterer Sprachpaarungen*

- Durch Kombinationen verschiedener Spurgeneratoren mit verschiedenen Process-Mining-Werkzeugen kann die Abdeckung von Translationsmöglichkeiten für verschiedene Paarungen an Prozessmodellierungssprachen signifikant vergrößert werden. Die Untersuchungen, für welche dieser Paarungen das Translationsprinzip adäquate Ergebnisse liefert, sollten weiter ausgedehnt werden, um potentiellen Anwendern die Einschätzung zu ermöglichen, ob die zu erwartende Ergebnisqualität ausreichend ist oder nicht.

- Process-Mining-Techniken erzeugen potentiell äußerst komplexe Prozessmodelle. Dadurch könnte das Ziel der Verbesserung des Verständnisses

von Prozessmodellen durch das induktive Translationsprinzip gefährdet werden. Da die Komplexität zu großen Teilen auf redundante Beschreibungen zurückzuführen ist, sollten Ansätze zur automatischen Reduktion der Modellkomplexität untersucht und bei zufriedenstellenden Ergebnissen in die Werkzeugkette aufgenommen werden.

*Reduktion der Modellkomplexität*

- Zum aktuellen Zeitpunkt der Untersuchungen werden lediglich von Spurgeneratoren künstlich erzeugte Ereignisprotokolle als Translationsmedium verwendet. Diese weisen vor allem den Vorteil der Rauschfreiheit auf, haben aber gleichzeitig den Nachteil, keine Informationen über das *prototypische* Prozessverhalten zu beinhalten. Aus diesem Grund ist ein weiteres sich anschließendes Forschungsthema die Erschließung der Vorteile bei der Kombination künstlicher und real-historischer Ereignisprotokolle bei der Übersetzung selbst und bei der Validierung des Übersetzungsergebnisses. Ein zu erwartendes Ergebnis ist, dass so geprüft werden kann, inwiefern das erzeugte Prozessmodell typische Verläufe des Prozesses unterstützt.

*Kombination künstlicher und real-historischer Ereignisprotokolle*

Der dritte und letzte konzeptionelle Beitrag der vorliegenden Arbeit ist die Entwicklung einer Technik zur Generierung natürlichsprachlicher Texte für multiperspektivische, deklarative Prozessmodelle, welche in Abschnitt 6.2 vorgestellt wird. Diese wird lediglich exemplarisch untersucht und erfordert daher einerseits eine detailliertere Ausarbeitung der verwendeten Konzepte, lässt aber andererseits bereits potentielle Folgethemen erkennen:

*Forschung: NLG*

- Der aktuelle Forschungsstand der Thematik spart eine umfassende Evaluation der Korrektheit der Ergebnisse der Textgenerierung aus. Ein wichtiges Ziel für eine detailliertere Ausarbeitung muss daher die Identifikation problematischer Regelkonstellationen im Quell-Prozessmodell sein, welche die Qualität des erzeugten Textes negativ beeinflussen. Weiterhin sind bisher keine Erkenntnisse über die Einflussnahme des Modellumfangs vorhanden. Auch eine Erweiterung der Mittel zur Strukturierung des Zieltextes auf Basis rhetorischer Relationen ist für die praktische Anwendung des Systems obligatorisch.

*Umfangreiche Evaluation, Erweiterung der Mittel zur Textstrukturierung*

- Eine weitere Qualitätsdimension ist die *Verständlichkeit* der generierten natürlichsprachlichen Texte. Diese Dimension sollte zukünftig unter Einbeziehung von Forschungsgruppen der Kognitionswissenschaften mit Spezialisierung auf Textverständnis untersucht werden. Eine mögliche Alternative bieten Metriken, wie beispielsweise *BLEU* und *NIST* [28] zur Bewertung der Ähnlichkeit von Texten in Kombination mit manuell erzeugten Referenztexten. Es hat sich allerdings herausgestellt, dass die meisten Metriken mit einem Bias behaftet sind, da sie auf dem Abgleich von *N-Grammen*, also Tupeln aus N Wörtern zwischen generiertem Text und Referenztext basieren. Aus diesem Grund wird eine Kombination aus einer Prüfung durch menschliche Experten und aus einer automatisierten Prüfung mittels der genannten Metriken empfohlen.

- Wie auch beim Spurgenerator ist die Transformation von DPIL-Regeln auf Basis statischer Muster definiert. Dies hat den großen Nachteil, dass aus-

schließlich solche Prozessmodelle ganzheitlich in natürlichsprachlichen Text überführt werden können, die ausschließlich mittels einer festen Menge an Regelschablonen definiert worden sind. Auch in der zugehörigen Publikation ([16]) wird bereits ein auf einer Grammatik basierender Ansatz zur Transformation der Regeln auf *atomarer* Ebene skizziert. Daher ist es ein vielversprechendes und essentielles Forschungsziel, die Transformation beliebiger deklarativer Prozessmodelle auf Basis der atomaren Bestandteile letzterer zu untersuchen.

*Textgenerierung auf Basis atomarer Modellbestandteile*

- Im Sinne einer stärkeren Automatisierung der Textgenerierungstechnik ist die Anwendung von *regelbasierten* Mitteln der Textstrukturierung hinderlich. Beispielsweise in [96] wird ein evolutionärer Algorithmus diskutiert, welcher mit stochastischen Mitteln einen Text auf Basis der Optimierung der Entitäten-Kohärenz strukturiert. Es ist zu untersuchen, ob dieser oder vergleichbare Algorithmen in der Lage sind, die regelbasierte Instanziierung rhetorischer Relationen durch stochastische Mittel zu ersetzen.

*Stochastische Textstrukturierung*

Für alle drei Kernbeiträge der vorliegenden Arbeit ist es zudem notwendig, die Hypothesen aus der durchgeführten qualitativen Umfrage empirisch auf ihre statistische Signifikanz zu prüfen. Die aufgestellten Hypothesen haben bisher lediglich den Charakter begründeter Vermutungen. Sie bilden aber beispielsweise eine geeignete Grundlage für sich anschließende Signifikanztests [189], womit überprüft werden kann, ob die gezielt ausgewählten Vertreter aus der Gruppe von Prozessmodellierern und ihre Einschätzungen repräsentativ sind oder nicht.

*Prüfung der Umfragehypothesen*

Die im Verlauf dieses Abschnitts aufgelisteten Forschungsthemen sind keinesfalls erschöpfend. Sie fokussieren aber auf wesentliche Problemstellungen, welche entweder eine Verbesserung der in dieser Arbeit präsentierten Techniken erfordern oder auf eine umfassendere Ausschöpfung des Potentials derselben abzielt. Bei allen Vorschlägen für zukünftige Forschungsarbeiten handelt es sich um konzeptionelle Erweiterungen, um weitere Verknüpfungen mit Forschungsergebnissen anderer Domänen oder auch um detailliertere Anwendungsstudien.

# Teil IV

# ANHÄNGE

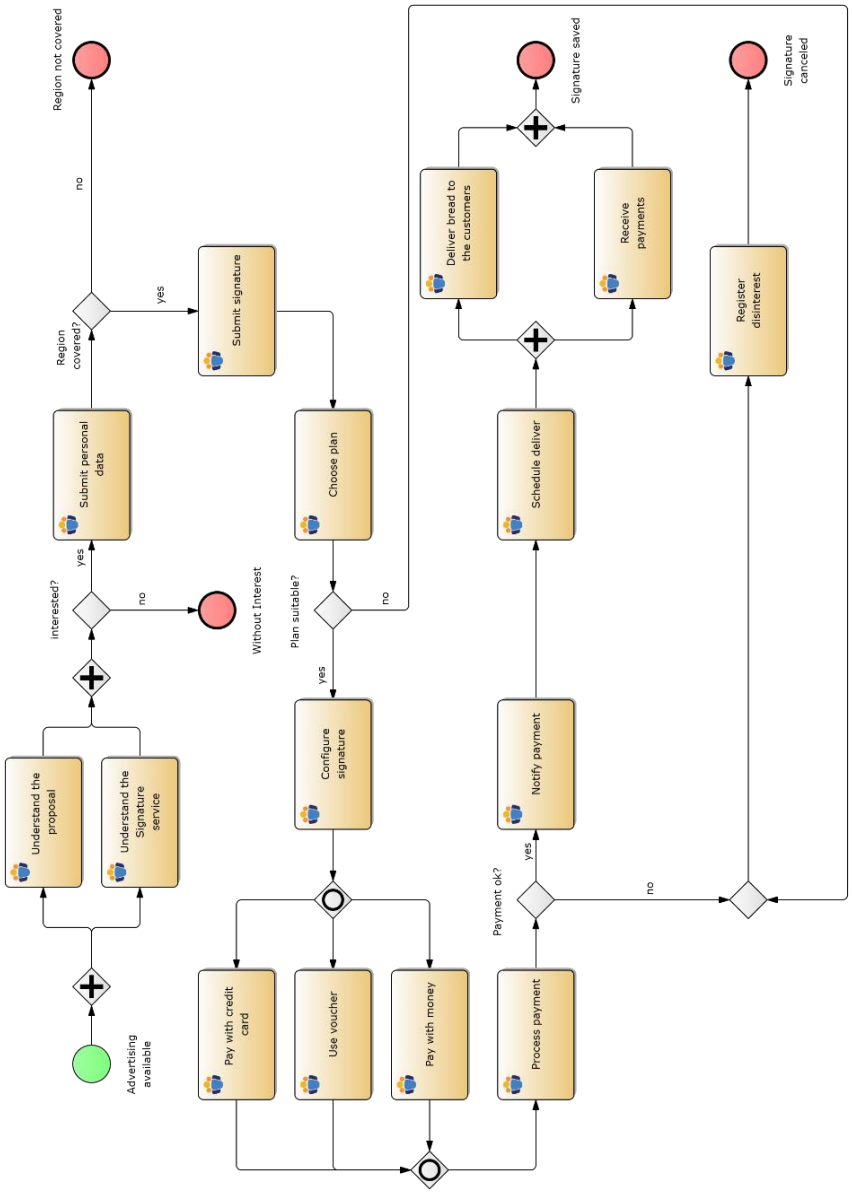



Abbildung A.1: Beispielmodell eines Prozesses zur Auslieferung von Brot (Quelle: [155])

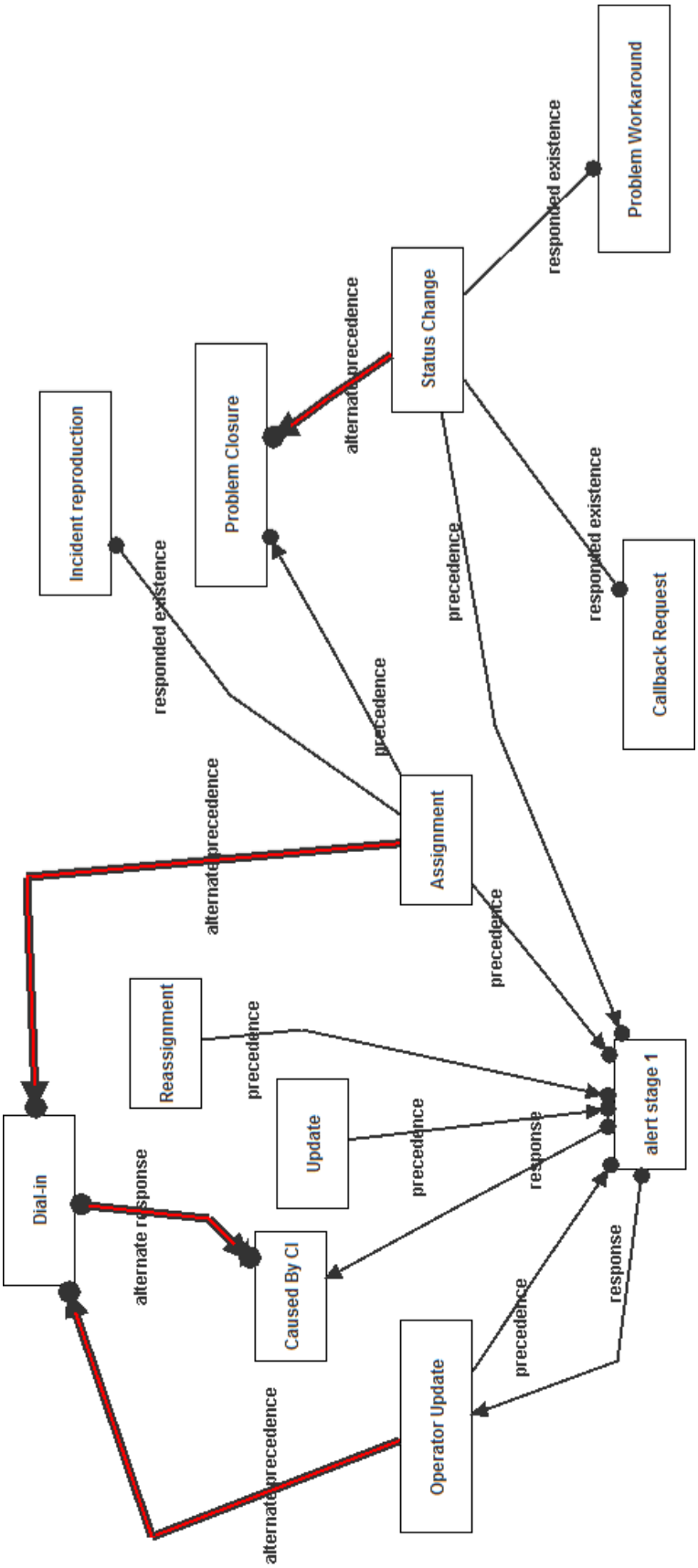



Abbildung A.2: Beispielmodell des Service-Desk-Prozesses eines Telekommunikationsunternehmens (Quelle: [41])

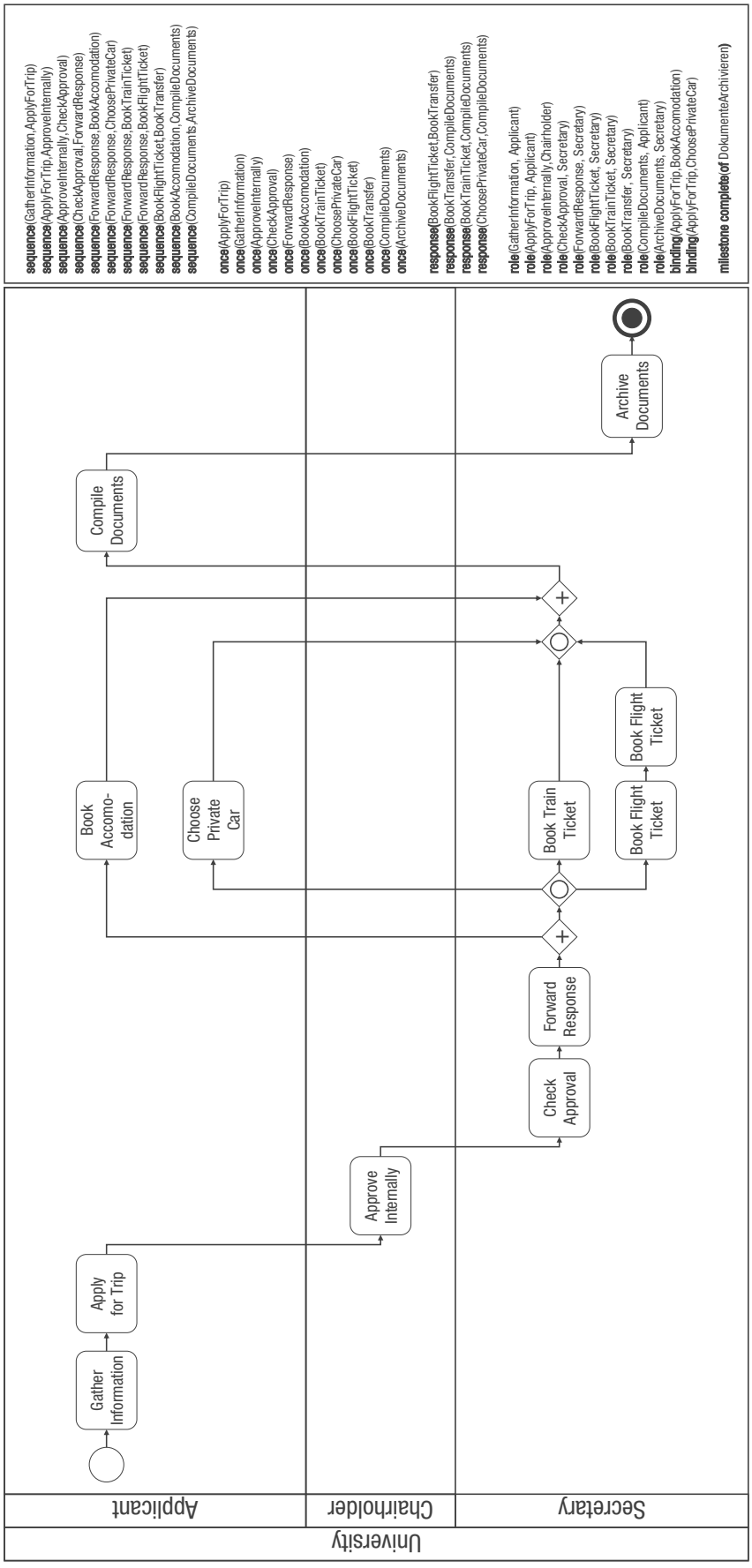



Abbildung A.3: Beispielmodell eines universitären Organisationsprozesses für Dienstreisen

[1] W. M. P. van der Aalst und M. Pesic. „DecSerFlow: Towards a Truly Declarative Service Flow Language“. In: *International Workshop on Web Services and Formal Methods (WS-FM)*. Springer, 2006, S. 1–23.

[2] Wil M. P. van der Aalst. „Business process simulation survival guide“. In: *Handbook on Business Process Management 1: Introduction, Methods, and Information Systems*. Springer, 2015, S. 337–370.

[3] Wil MP van der Aalst und PJS Berens. „Beyond workflow management: product-driven case handling“. In: *International ACM SIGGROUP Conference on Supporting Group Work*. ACM. 2001, S. 42–51.

[4] Wil MP van der Aalst, HT De Beer und Boudewijn F van Dongen. „Process mining and verification of properties: An approach based on temporal logic“. In: *OTM Confederated International Conferences On the Move to Meaningful Internet Systems*. Springer. 2005, S. 130–147.

[5] Wil MP van der Aalst, Mathias Weske und Dolf Grünbauer. „Case handling: a new paradigm for business process support“. In: *Data & Knowledge Engineering* 53.2 (2005), S. 129–162.

[6] Wil van der Aalst. *Process Mining: Data Science in Action*. Springer, 2016.

[7] Wil van der Aalst, Arya Adriansyah und Boudewijn van Dongen. „Replaying history on process models for conformance checking and performance analysis“. In: *Wiley Interdisciplinary Reviews: Data Mining and Knowledge Discovery* 2.2 (2012), S. 182–192.

[8] Wil van der Aalst, Ton Weijters und Laura Maruster. „Workflow mining: Discovering process models from event logs“. In: *IEEE Transactions on Knowledge and Data Engineering* 16.9 (2004), S. 1128–1142.

[17] Arya Adriansyah, Boudewijn F van Dongen und Wil MP van der Aalst. „Conformance checking using cost-based fitness analysis“. In: *IEEE International Enterprise Distributed Object Computing Conference (EDOC)*. IEEE. 2011, S. 55–64.

[18] AeroSpace und Defence Industries Association of Europe. *Simplified Technical English: Specification ASD-STE100*. 2010. URL: http://www.asd-ste100.org.

[19] A Alves, A Arkin, S Askary, C Barreto, B Bloch, F Curbera, M Ford, Y Goland, A Guzar, N Kartha u. a. *Web services business process execution language version 2.0*. 2007.

[20] Robert K. Atkinson, Sharon J. Derry, Alexander Renkl und Donald Wortham. „Learning from examples: Instructional principles from the worked examples research“. In: *Review of Educational Research* 70.2 (2000), S. 181–214.

[21] European Association of BPM (EABPM). *Common Body of Knowledge for BPM - Leitfaden für das Prozessmanagement*. 1. Verlag Dr. Götz Schmidt, 2009.

[22] Jerry Banks. *Discrete event system simulation*. 5. Aufl. Pearson, 2010.

[23] Michael Heinrich Baumann, Michaela Baumann, Stefan Schönig und Stefan Jablonski. „Towards multi-perspective process model similarity matching". In: *Workshop on Enterprise and Organizational Modeling and Simulation*. Bd. 191. Springer. 2014, S. 21–37.

[25] HS Beattie und Robert A. Rahenkamp. „IBM typewriter innovation". In: *IBM journal of research and development* 25.5 (1981), S. 729–740.

[26] Jörg Becker, Michael Rosemann und Christoph Von Uthmann. „Guidelines of Business Process Modeling". In: *Business Process Management* (2000), S. 30–49.

[27] Michael Becker und Ralf Laue. „A comparative survey of business process similarity measures". In: *Computers in Industry* 63.2 (2012), S. 148–167.

[28] Anja Belz und Ehud Reiter. „Comparing Automatic and Human Evaluation of NLG Systems". In: *European Chapter of the ACL (EACL)*. 2006.

[29] Ann L. Brown und Mary Jo Kane. „Preschool children can learn to transfer: Learning to learn and learning from example". In: *Cognitive Psychology* 20.4 (1988), S. 493–523.

[30] Alan Bryman. „The Debate about Quantitative and Qualitative Research: A Question of Method or Epistemology?" In: *The British Journal of Sociology* 35.1 (1984), S. 75–92.

[31] JCAM Buijs. „Flexible evolutionary algorithms for mining structured process models". Diss. Eindhoven University of Technology, 2014.

[32] Joos CAM Buijs, Boudewijn F van Dongen und Wil MP van der Aalst. „A genetic algorithm for discovering process trees". In: *Evolutionary Computation (CEC), 2012 IEEE Congress on*. IEEE. 2012, S. 1–8.

[33] Andrea Burattin. „PLG2: Multiperspective Processes Randomization and Simulation for Online and Offline Settings". In: *CoRR* abs/1506.08415 (2015).

[34] Andrea Burattin. „PLG2: Multiperspective Process Randomization with Online and Offline Simulations". In: *Online Proceedings of the BPM Demo Track*. CEUR-WS.org, 2016.

[35] Andrea Burattin, Fabrizio Maria Maggi und Alessandro Sperduti. „Conformance checking based on multi-perspective declarative process models". In: *Expert Systems with Applications* 65 (2016).

[36] Andrea Burattin und Alessandro Sperduti. „PLG: A framework for the generation of business process models and their execution logs". In: *Business Process Management Workshops*. Springer, 2011, S. 214–219.

[37] Andrea Burattin, Alessandro Sperduti und Marco Veluscek. „Business models enhancement through discovery of roles". In: *IEEE Symposium on Computational Intelligence and Data Mining (CIDM)*. IEEE. 2013, S. 103–110.

[38] Christoph Bussler. *Organisationsverwaltung in Workflow-Management-Systemen*. Deutscher Universitätsverlag, 1998.

[39] Daniel Calegari und Nora Szasz. „Verification of model transformations: A survey of the state-of-the-art". In: *Electronic Notes in Theoretical Computer Science* 292 (2013), S. 5–25.

[40] María Agustina Cibran. „Translating BPMN models into UML activities". In: *International Conference on Business Process Management*. Springer. 2008, S. 236–247.

[41] Claudio Di Ciccio und Mario Luca Bernardi. „Generating Event Logs through the Simulation of Declare Models". In: *Workshop on Enterprise and Organizational Modeling and Simulation (EOMAS)*. Springer, 2015, S. 20–36.

[42] Claudio Di Ciccio und Massimo Mecella. „On the discovery of declarative control flows for artful processes". In: *ACM Transactions on Management Information Systems (TMIS)* 5.4 (2015), S. 24.

[43] Raffaele Conforti, Marlon Dumas, Luciano García-Bañuelos und Marcello La Rosa. „BPMN miner: automated discovery of BPMN process models with hierarchical structure". In: *Information Systems* 56 (2016), S. 284–303.

[44] Jonathan E Cook und Alexander L Wolf. „Discovering models of software processes from event-based data". In: *ACM Transactions on Software Engineering and Methodology (TOSEM)* 7.3 (1998), S. 215–249.

[45] Edward Corwin und Antonette Logar. „Sorting in linear time-variations on the bucket sort". In: *Journal of computing sciences in colleges* 20.1 (2004), S. 197–202.

[46] J.W. Creswell und V.L.P. Clark. *Designing and Conducting Mixed Methods Research*. SAGE Publications, 2011.

[47] Bill Curtis, Marc I Kellner und Jim Over. „Process modeling". In: *Communications of the ACM* 35.9 (1992), S. 75–90.

[48] Krzysztof Czarnecki und Simon Helsen. „Feature-based survey of model transformation approaches". In: *IBM Systems Journal* 45.3 (2006), S. 621–645.

[49] Giuseppe De Giacomo, Marlon Dumas, Fabrizio Maria Maggi und Marco Montali. „Declarative process modeling in BPMN". In: *International Conference on Advanced Information Systems Engineering*. Springer. 2015, S. 84–100.

[50] Giuseppe De Giacomo, Marlon Dumas, Fabrizio Maria Maggi und Marco Montali. „Declarative process modeling in BPMN". In: *International Conference on Advanced Information Systems Engineering*. Springer. 2015, S. 84–100.

[51] Juan De Lara, Hans Vangheluwe und Manuel Alfonseca. „Meta-modelling and graph grammars for multi-paradigm modelling in AToM3". In: *Software & Systems Modeling* 3.3 (2004), S. 194–209.

[52] Jochen De Weerdt, Seppe KLM vanden Broucke und Filip Caron. „Bidimensional process discovery for mining BPMN models". In: *Business Process Management*. Springer. 2014, S. 529–540.

[53] Gero Decker, Remco Dijkman, Marlon Dumas und Luciano García-Bañuelos. „Transforming BPMN diagrams into YAWL nets". In: *International Conference on Business Process Management*. Springer. 2008, S. 386–389.

[54] Claudia Di Ciccio und Massimo Mecella. *MINERful, a mining algorithm for declarative process constraints in MailOfMine*. Techn. Ber. Universität La Sapienza, 2012.

[55] Claudio Di Ciccio, Andrea Marrella und Alessandro Russo. „Knowledge-intensive processes: characteristics, requirements and analysis of contemporary approaches". In: *Journal on Data Semantics* 4.1 (2015), S. 29–57.

[56] Remco M Dijkman, Marlon Dumas und Chun Ouyang. „Semantics and analysis of business process models in BPMN". In: *Information and Software technology* 50.12 (2008), S. 1281–1294.

[57] Klaus R Dittrich, Stella Gatziu und Andreas Geppert. „The active database management system manifesto: A rulebase of ADBMS features". In: *International Workshop on Rules in Database Systems*. Springer. 1995, S. 1–17.

[58] Boudewijn F van Dongen und Wil MP van der Aalst. „A Meta Model for Process Mining Data". In: *CAiSE Workshops*. Bd. 160. FEUP E., 2005, S. 30.

[59] Richard O Duda, Peter E Hart und David G Stork. *Pattern classification*. John Wiley & Sons, 2012.

[60] Marlon Dumas, Wil M van der Aalst und Arthur H Ter Hofstede. *Process-aware information systems: bridging people and software through process technology*. John Wiley & Sons, 2005.

[61] Dirk Fahland, Daniel Lübke, Jan Mendling, Hajo Reijers, Barbara Weber, Matthias Weidlich und Stefan Zugal. „Declarative versus imperative process modeling languages: The issue of understandability". In: *Enterprise, Business-Process and Information Systems Modeling*. Springer, 2009, S. 353–366.

[62] Dirk Fahland, Jan Mendling, Hajo A Reijers, Barbara Weber, Matthias Weidlich und Stefan Zugal. „Declarative versus imperative process modeling languages: the issue of maintainability". In: *Business Process Management Workshops*. Springer. 2009, S. 477–488.

[63] Cédric Favre, Dirk Fahland und Hagen Völzer. „The relationship between workflow graphs and free-choice workflow nets". In: *Information Systems* 47 (2015), S. 197–219.

[64] Usama Fayyad, Gregory Piatetsky-Shapiro und Padhraic Smyth. „From data mining to knowledge discovery in databases". In: *AI magazine* 17.3 (1996), S. 37–54.

[65] Martin Fowler. *Domain-specific languages*. Pearson Education, 2010.

[66] Marcela Genero, Geert Poels und Mario Piattini. „Defining and validating metrics for assessing the understandability of entity–relationship diagrams". In: *Data & Knowledge Engineering* 64.3 (2008), S. 534 –557.

[67] E Mark Gold. „Language identification in the limit". In: *Information and Control* 10.5 (1967), S. 447 –474.

[68] Richard C Gronback. *Eclipse modeling project: a domain-specific language (DSL) toolkit*. Pearson Education, 2009.

[69] Gruschko Gruschko, Gruschko Gruschko, Dimitrios Kolovos, Dimitrios Kolovos, Richard Paige und Richard Paige. „Towards Synchronizing Models with Evolving Metamodels". In: *Workshop on Model-Driven Software Evolution (MODSE), 11th European Conference on Software Maintenance and Reengineering*. IEEE. 2007.

[70] Christian W Günther und HMW Verbeek. *XES Standard Definition*. 2014. URL: http://www.xes-standard.org/ (besucht am 02. 12. 2016).

[71] Holger Günzel und Andreas Bauer. *Data-Warehouse-Systeme: Architektur, Entwicklung, Anwendung*. dpunkt.verlag, 2013.

[72] Michael Häder. *Erhebungsmethoden*. Springer, 2015, S. 189–343.

[73] Cornelia Haisjackl, Irene Barba, Stefan Zugal, Pnina Soffer, Irit Hadar, Manfred Reichert, Jakob Pinggera und Barbara Weber. „Understanding Declare models: strategies, pitfalls, empirical results". In: *Software and Systems Modeling* 15.2 (2016), S. 325–352.

[74] Wolfgang Hesse und Heinrich C Mayr. „Modellierung in der Softwaretechnik: eine Bestandsaufnahme". In: *Informatik-Spektrum* 31.5 (2008), S. 377–393.

[75] Arthur HM ter Hofstede, Massimo Mecella, Sebastian Sardina und Andrea Marrella. „Knowledge-intensive Business Processes". In: *International Workshop on Knowledge-intensive Business Processes*. Bd. 861. CEUR-WS.org, 2012.

[76] David Hollingsworth. „The Workflow Reference Model". In: *Workflow Management Coalition* 19.1.1 (1995), S. 1–55.

[77] Anatol W Holt, HR Ramsey und JD Grimes. „Coordination system technology as the basis for a programming environment". In: *Electrical Communication* 57.4 (1983), S. 307–314.

[78] John E Hopcroft, Jeffrey D Ullman und Rajeev Motwani. *Einführung in die Automatentheorie, formale Sprachen und Komplexitätstheorie*. Bd. 2. Pearson Studium Deutschland, München, 2002.

[79] Eduard H Hovy. „Automated discourse generation using discourse structure relations". In: *Artificial intelligence* 63.1 (1993), S. 341–385.

[80] *IEEE Approved Draft Standard for System, Software and Hardware Verification and Validation*. 2016.

[81] *ISO9001:2008: Quality management systems – Requirements*. 2008.

[82] Michael Igler. „ESProNa-Eine Constraintsprache zur multimodalen Prozessmodellierung und navigationsgestützten Ausführung". Diss. Universität Bayreuth, 2012.

[83] Marta Indulska, Jan Recker, Michael Rosemann und Peter Green. „Business Process Modeling: Current Issues and Future Challenges". In: *International Conference on Advanced Information Systems Engineering (CAiSE)*. Springer, 2009, S. 501–514.

[84] Stefan Jablonski. *Workflow-Management: Entwicklung von Anwendungen und Systemen; Facetten einer neuen Technologie*. dpunkt.verlag, 1997.

[85] Stefan Jablonski und C Bussler. „MOBILE: A modular workflow model and architecture". In: *International Working Conference on Dynamic Modelling and Information Systems*. 1994.

[86] Stefan Jablonski und Christoph Bussler. *Workflow management: modeling concepts, architecture and implementation*. International Thomson Computer Press, 1996.

[87] Daniel Jackson. „Automating First-order Relational Logic". In: *ACM SIGSOFT International Symposium on Foundations of Software Engineering: Twenty-first Century Applications*. Bd. 25. 6. ACM, 2000, S. 130–139.

[88] Daniel Jackson. *Software Abstractions: logic, language, and analysis*. MIT press, 2012.

[89] Kurt Jensen. *Coloured Petri nets: basic concepts, analysis methods and practical use*. Bd. 1. Springer Science & Business Media, 2013.

[90] Kurt Jensen, Lars Michael Kristensen und Lisa Wells. „Coloured Petri Nets and CPN Tools for modelling and validation of concurrent systems". In: *International Journal on Software Tools for Technology Transfer* 9.3 (2007), S. 213–254.

[91] Jens Bæk Jørgensen, Kristian Bisgaard Lassen und Wil MP van der Aalst. „From task descriptions via colored Petri nets towards an implementation of a new electronic patient record workflow system". In: *International Journal on Software Tools for Technology Transfer (STTT)* 10.1 (2008), S. 15–28.

[92] Toon Jouck. „PTandLogGenerator: A Generator for Artificial Event Data". In: (2016).

[93] Marcel A Just und Patricia A Carpenter. „A capacity theory of comprehension: individual differences in working memory." In: *Psychological review* 99.1 (1992), S. 122.

[94] Sylvain Kahane. „The Meaning-Text Theory". In: *Dependency and Valency, An International Handbooks of Contemporary Research* 25.1 (2003), S. 546–569.

[95] Anna A Kalenkova, Wil MP van der Aalst, Irina A Lomazova und Vladimir A Rubin. „Process mining using BPMN: relating event logs and process models". In: *ACM/IEEE International Conference on Model Driven Engineering Languages and Systems*. ACM. 2016, S. 123–123.

[96] Nikiforos Karamanis und Hisar Maruli Manurung. „Stochastic text structuring using the principle of continuity". In: *International Conference on Natural Language Generation (INLG)*. Bd. 2. 2002, S. 81–88.

[97] Valeriia Kataeva. „Applying graph grammars for the generation of process models and their logs". In: *Spring/Summer Young Researchers' Colloquium on Software Engineering (SYRCoSE)*. 2014, S. 83–87.

[98] Dan Klein und Christopher D. Manning. „Fast Exact Inference with a Factored Model for Natural Language Parsing". In: *International Conference on Neural Information Processing Systems*. MIT Press, 2002, S. 3–10.

[99] Alexander Königs. „Model transformation with triple graph grammars". In: *Model Transformations in Practice (Satellite Workshop of MODELS)*. 2005, S. 166.

[100] Oliver Kopp, Daniel Martin, Daniel Wutke und Frank Leymann. „On the Choice Between Graph-Based and Block-Structured Business Process Modeling Languages". In: *Modellierung betrieblicher Informationssysteme (MobIS)*. Bd. P-141. Gesellschaft für Informatik e.V. (GI), 2008, S. 59–72.

[101] Harald Kühn, Marion Murzek und Franz Bayer. „Horizontal Business Process Model Interoperability using Model Transformation". In: *INTEREST Workshop der ECOOP*. 2004.

[102] Richard Lakin, Nick Capon und Neil Botten. „BPR enabling software for the financial services industry". In: *Management services* 40.3 (1996), S. 18–20.

[103] George Lakoff. *Women, fire, and dangerous things: What categories reveal about the mind*. Cambridge University Press, 1990.

[104] Siegfried Lamnek. *Qualitative Sozialforschung: Lehrbuch*. Beltz, 2005.

[105] Sebastian Land und Simon Fischer. *RapidMiner 5 - RapidMiner in academic use*. 2012.

[106] Sander JJ Leemans, Dirk Fahland und Wil MP van der Aalst. „Discovering block-structured process models from event logs-a constructive approach". In: *International Conference on Applications and Theory of Petri Nets and Concurrency*. Springer. 2013, S. 311–329.

[107] Henrik Leopold, Jan Mendling und Artem Polyvyanyy. „Generating natural language texts from business process models". In: *International Conference on Advanced Information Systems Engineering (CAiSE)*. Springer, 2012, S. 64–79.

[108] Henrik Leopold, Jan Mendling und Artem Polyvyanyy. „Supporting process model validation through natural language generation". In: *IEEE Transactions on Software Engineering* 40.8 (2014), S. 818–840.

[109] Henrik Leopold, Sergey Smirnov und Jan Mendling. „Recognising activity labeling styles in business process models". In: *Enterprise Modelling and Information Systems Architectures* 6.1 (2011), S. 16–29.

[110] Tihamer Levendovszky, Daniel Balasubramanian, Anantha Narayanan, Feng Shi und Chris Van Buskirk. „A semi-formal description of migrating domain-specific models with evolving domains". In: *Software & Systems Modeling* 13.2 (2014), S. 807–823.

[111] John W Lloyd. *Declarative programming in Escher*. Techn. Ber. Department of Computer Science, University of Bristol, 1995.

[112] Wenhong Luo und Alex Tung. „A framework for selecting business process modeling methods". In: *Industrial Management & Data Systems* 99.7 (1999), S. 312–319.

[113] Sandra Lusk, Staci Paley und Andrew Spanyi. „The evolution of business process management as a professional discipline". In: *BPTrends* (2005).

[114] Ondrej Macek und Karel Richta. „The BPM to UML activity diagram transformation using XSLT". In: *Dateso*. Bd. 9. CEUR-WS.org, 2009, S. 119–129.

[115] Fabrizio M Maggi, RP Jagadeesh Chandra Bose und Wil MP van der Aalst. „Efficient discovery of understandable declarative process models from event logs". In: *International Conference on Advanced Information Systems Engineering (CAiSE)*. Springer. 2012, S. 270–285.

[116] Fabrizio M Maggi, Arjan J Mooij und Wil MP van der Aalst. „User-guided discovery of declarative process models". In: *IEEE Symposium on Computational Intelligence and Data Mining (CIDM)*. IEEE. 2011, S. 192–199.

[117] Fabrizio Maria Maggi, Marlon Dumas, Luciano García-Bañuelos und Marco Montali. „Discovering data-aware declarative process models from event logs". In: *Business Process Management*. Springer, 2013, S. 81–96.

[118] Saleem Malik und Imran Sarwar Bajwa. „Business Process Management Workshops". In: *Lecture Notes in Business Information Processing*. Bd. 132. 2012, S. 611–622.

[119] William C Mann und Sandra A Thompson. „Rhetorical structure theory: Toward a functional theory of text organization". In: *Text-Interdisciplinary Journal for the Study of Discourse* 8.3 (1988), S. 243–281.

[120] Ronny Mans, Wil MP van der Aalst und HMW (Eric) Verbeek. „Supporting Process Mining Workflows with RapidProM". In: *Online Proceedings of the BPM Demo Track*. CEUR-WS.org, 2014, S. 56.

[121] Ronny Mans, Wil MP van der Aalst und HMW (Eric) Verbeek. „Supporting Process Mining Workflows with RapidProM". In: *Online Proceedings of the BPM Demo Track*. CEUR-WS.org, 2014, S. 56.

[122] Morten Marquard, Muhammad Shahzad und Tijs Slaats. „Web-Based Modelling and Collaborative Simulation of Declarative Processes". In: *Business Process Management*. Springer, 2015, S. 209–225.

[123] Kathleen R McKeown. „Discourse strategies for generating natural-language text". In: *Artificial Intelligence* 27.1 (1985), S. 1–41.

[124] Ana Karla Alves de Medeiros und Christian W. Günther. „Process Mining: Using CPN Tools to Create Test Logs for Mining Algorithms". In: *Workshop and Tutorial on Practical Use of Coloured Petri Nets and the CPN Tools* (2005), S. 177–190.

[125] Igor A Mel'čuk und Alain Polguere. „A formal lexicon in the meaning-text theory:(or how to do lexica with words)". In: *Computational linguistics* 13.3-4 (1987), S. 261–275.

[126] J. Mendling, H.A. Reijers und W.M.P. van der Aalst. „Seven process modeling guidelines (7PMG)". In: *Information and Software Technology* 52.2 (2010), S. 127–136.

[127] Jan Mendling, Michael Moser und Gustaf Neumann. „Transformation of yEPC business process models to YAWL". In: *ACM symposium on Applied computing*. ACM. 2006, S. 1262–1266.

[128] Tom Mens und Pieter Van Gorp. „A taxonomy of model transformation". In: *Electronic Notes in Theoretical Computer Science* 152 (2006), S. 125–142.

[129] Bart Meyers und Hans Vangheluwe. „A framework for evolution of modelling languages". In: *Science of Computer Programming* 76.12 (2011), S. 1223–1246.

[130] George A Miller. „WordNet: a lexical database for English". In: *Communications of the ACM* 38.11 (1995), S. 39–41.

[131] Lee Momtahan. „Towards a Small Model Theorem for Data Independent Systems in Alloy". In: *Electronic Notes in Theoretical Computer Science* 128.6 (2005), S. 37 –52.

[132] James Donald Monk. *Mathematical logic*. Bd. 37. Springer, 2012.

[133] Marco Montali. *Specification and verification of declarative open interaction models: a logic-based approach*. Bd. 56. Springer Science & Business Media, 2010.

[134] Marion Murzek und Gerhard Kramler. „The model morphing approach–horizontal transformations between business process models". In: *International Conference on Perspectives in Business Information Research (BIR)*. University of Tampere, Finnland. 2007, S. 88–103.

[135] Marion Murzek, Gerhard Kramler und Elke Michlmayr. „Structural patterns for the transformation of business process models". In: *Enterprise Distributed Object Computing Conference Workshops, 2006. EDOCW'06. 10th IEEE International*. IEEE. 2006, S. 18–18.

[136] Manuel Neurauter, Jakob Pinggera, Markus Martini, Andrea Burattin, Marco Furtner, Pierre Sachse und Barbara Weber. „The Influence of Cognitive Abilities and Cognitive Load on Business Process Models and Their Creation". In: (2015). Hrsg. von Fred D. Davis, René Riedl, Jan vom Brocke, Pierre-Majorique Léger und Adriane B. Randolph, S. 107–115.

[137] Björn Niehaves. „The reflective designer: designing IT-consulting process". Diss. Wirtschaftswissenschaftliche Fakultät der Westfälischen Wilhelms-Universität Münster, 2006.

[138] Wolfgang Niemeier. *Ausgleichungsrechnung: Statistische Auswertemethoden*. Walter de Gruyter, 2008.

[139] Markus Nüttgens, Thomas Feld und Volker Zimmermann. „Business Process Modeling with EPC and UML: transformation or integration?" In: *The Unified Modeling Language*. Springer, 1998, S. 250–261.

[140] OMG. *MOF Model to Text Transformation Language, v1.0*. 2008. URL: http://www.omg.org/spec/MOFM2T/1.0/.

[141] OMG. *Business Process Model and Notation (BPMN), Version 2.0*. 2011. URL: http://www.omg.org/spec/BPMN/2.0.

[142] OMG. *Business Process Model and Notation (BPMN), Version 2.0.2*. 2013. URL: http://www.omg.org/spec/BPMN/2.0.2.

[143] Maja Pesic. „Constraint-based workflow management systems: shifting control to users". Diss. Eindhoven University of Technology, 2008.

[144] Maja Pesic und Wil MP van der Aalst. „A declarative approach for flexible business processes management". In: *Business Process Management Workshops*. Springer. 2006, S. 169–180.

[145] Maja Pesic, Helen Schonenberg und Wil M P Van Der Aalst. „DECLARE: Full support for loosely-structured processes". In: *IEEE International Enterprise Distributed Object Computing Workshop, EDOC*. IEEE, 2007, S. 287–298.

[146] Paul Pichler, Barbara Weber, Stefan Zugal, Jakob Pinggera, Jan Mendling und Hajo A Reijers. „Imperative versus declarative process modeling languages: An empirical investigation". In: *International Conference on Business Process Management*. Springer. 2011, S. 383–394.

[147] Johannes Prescher, Claudio Di Ciccio und Jan Mendling. „From Declarative Processes to Imperative Models". In: *International symposium on data-driven process discovery and analysis*. CEUR-WS.org. 2014, S. 162–173.

[148] Corina Raduescu, Hui Min Tan, Malini Jayaganesh, Wasana Bandara, Michael zur Muehlen und Sonia Lippe. „A framework of issues in large process modeling projects". In: *ECIS*. 2006. 2006, S. 1594–1605.

[149] Ivo Raedts, Marija Petkovic, Yaroslav S Usenko, Jan Martijn EM van der Werf, Jan Friso Groote und Lou J Somers. „Transformation of BPMN Models for Behaviour Analysis". In: *Modelling, Simulation, Verification and Validation of Enterprise Information Systems (MSVVEIS)*. INSTICC PRESS, 2007, S. 126–137.

[150] Mohamed Ramadan, Hicham G Elmongui und Riham Hassan. „BPMN formalisation using coloured Petri Nets". In: *International Conference on Software Engineering & Applications (SEA)*. GSTF Digital Library, 2011.

[151] Keith Rayner, Alexander Pollatsek, Jane Ashby und Charles Clifton Jr. *Psychology of reading*. Psychology Press, 2012.

[152] Jan Recker, Hajo A. Reijers und Sander G. van de Wouw. „Process model comprehension: The effects of cognitive abilities, learning style, and strategy". In: *Communications of the Association for Information Systems* 34.1 (2014), S. 199–222.

[153] Manfred Reichert und Barbara Weber. *Enabling flexibility in process-aware information systems: challenges, methods, technologies*. Springer, 2012.

[154] Ehud Reiter, Robert Dale und Zhiwei Feng. *Building natural language generation systems*. Bd. 33. MIT Press, 2000.

[155] Raphael De A Rodrigues, Márcio De O Barros, Kate Revoredo, Leonardo G Azevedo und Henrik Leopold. „An experiment on process model understandability using textual work instructions and BPMN models". In: *Brazilian Symposium on Software Engineering (SBES)*. IEEE. 2015, S. 41–50.

[156] A Rozinat, M T Wynn, Wil M P van der Aalst, Arthur H M ter Hofstede und C J Fidge. „Workflow Simulation for Operational Decision Support Using Design, Historic and State Information". In: Springer, 2008, S. 196–211.

[157] Anne Rozinat, RS Mans, Minseok Song und W MP van der Aalst. „Discovering colored Petri nets from event logs". In: *International Journal on Software Tools for Technology Transfer (STTT)* 10.1 (2008), S. 57–74.

[158] Anne Rozinat, Moe Thandar Wynn, Wil MP van der Aalst, Arthur HM ter Hofstede und Colin J Fidge. „Workflow simulation for operational decision support". In: *Data & Knowledge Engineering* 68.9 (2009), S. 834–850.

[159] Nicholas Russell, Arthur HM Ter Hofstede und Wil M van der Aalst. „newYAWL: specifying a workflow reference language using coloured Petri nets". In: *Workshop and Tutorial on Practical Use of Coloured Petri Nets and the CPN Tools*. Department of Computer Science, University of Aarhus, Denmark, 2007.

[160] Nick Russell, Arthur H. M. Ter Hofstede und Nataliya Mulyar. *Workflow ControlFlow Patterns: A Revised View*. Techn. Ber. BPM Center, 2006.

[161] Stefan Schönig. „Ein Process Mining-Rahmenwerk für agile, personenbezogene Prozesse". Diss. Universität Bayreuth, 2015.

[163] Stefan Schönig, Michael Zeising und Stefan Jablonski. „Towards location-aware declarative business process management". In: *International Conference on Business Information Systems*. Bd. 183. Springer. 2014, S. 40–51.

[164] Stefan Schönig, Cristina Cabanillas, Stefan Jablonski und Jan Mendling. „Mining the organisational perspective in agile business processes". In: *International Conference on Enterprise, Business-Process and Information Systems Modeling*. Springer. 2015, S. 37–52.

[165] Stefan Schönig, Claudio Di Ciccio, Fabrizio M Maggi und Jan Mendling. „Discovery of Multi-perspective Declarative Process Models". In: *International Conference on Service-Oriented Computing*. Springer. 2016, S. 87–103.

[166] Stefan Schönig, Andreas Rogge-Solti, Cristina Cabanillas, Stefan Jablonski und Jan Mendling. „Efficient and customisable declarative process mining with SQL". In: *International Conference on Advanced Information Systems Engineering (CAiSE)*. Springer. 2016, S. 290–305.

[167] David W Scott. „Box–muller transformation". In: *Wiley Interdisciplinary Reviews: Computational Statistics* 3.2 (2011), S. 177–179.

[168] Ivan Shugurov und Alexey A Mitsyuk. „Generation of a set of event logs with noise". In: *Spring/Summer Young Researchers' Colloquium on Software Engineering (SYRCoSE)*. 2014, S. 88–95.

[169] Tijs Slaats, Raghava Rao Mukkamala, Thomas Hildebrandt und Morten Marquard. „Exformatics Declarative Case Management Workflows as DCR Graphs". In: *Business Process Management*. Springer, 2013, S. 339–354.

[170] Herbert Stachowiak. *Allgemeine Modelltheorie*. Springer, 1973.

[171] Tony Spiteri Staines. „Intuitive mapping of UML 2 activity diagrams into fundamental modeling concept Petri net diagrams and colored Petri nets". In: *IEEE International Conference and Workshop on the Engineering of Computer Based Systems (ECBS)*. IEEE. 2008, S. 191–200.

[172] Dave Steinberg, Frank Budinsky, Ed Merks und Marcelo Paternostro. *EMF: eclipse modeling framework*. 2. Aufl. Pearson Education, 2009.

[173] Thomas Stocker und Rafael Accorsi. „Secsy: Security-aware synthesis of process event logs". In: *Workshop on Enterprise Modelling and Information Systems Architectures*. 2013, S. 71–84.

[174] Michael Strommer, Marion Murzek und Manuel Wimmer. „Applying model transformation by-example on business process modeling languages". In: *International Conference on Conceptual Modeling*. Springer. 2007, S. 116–125.

[175] Keith D Swenson, Nathaniel Palmer u. a. *Mastering the unpredictable: how adaptive case management will revolutionize the way that knowledge workers get things done*. Bd. 1. Meghan-Kiffer Press Tampa, 2010.

[176] Elham Ramezani Taghiabadi, Vladimir Gromov, Dirk Fahland und WilM P van der Aalst. „Compliance checking of data-aware and resource-aware compliance requirements". In: *OTM Confederated International Conferences On the Move to Meaningful Internet Systems"*. Springer. 2014, S. 237–257.

[177] JBoss Drools Team. *JBoss Drools Documentation-Chapter 7: Rule Language Reference*. 2013.

[178] Willi Tscheschner. „Transformation from EPC to BPMN". In: *Business Process Technology* 1.3 (2006), S. 7–21.

[179] Roman Vaculin, Richard Hull, Terry Heath, Craig Cochran, Anil Nigam und Piyawadee Sukaviriya. „Declarative business artifact centric modeling of decision and knowledge intensive business processes". In: *Enterprise Distributed Object Computing Conference (EDOC), 2011 15th IEEE International*. IEEE. 2011, S. 151–160.

[180] Roman Vaculin, Richard Hull, Maja Vukovic, Terry Heath, Nathaniel Mills und Yutian Sun. „Supporting collaborative decision processes". In: *IEEE International Conference on Services Computing (SCC)*. IEEE. 2013, S. 651–658.

[181] Wil M.P. Van Der Aalst und Arthur H. M. Ter Hofstede. „Verification of Workflow Task Structures: A Petri-net-based Approach". In: *Information Systems* 25.1 (2000), S. 43–69.

[182] Wil Van Der Aalst, Arya Adriansyah, Ana Karla Alves De Medeiros, Franco Arcieri, Thomas Baier, Tobias Blickle, Jagadeesh Chandra Bose, Peter van den Brand, Ronald Brandtjen, Joos Buijs u. a. „Process mining manifesto". In: *Business Process Management*. Springer. 2011, S. 169–194.

[183] Dominik Vanderhaeghen, Sven Zang, Anja Hofer und Otmar Adam. „XML-based transformation of business process models–enabler for collaborative business process management". In: *GI-Workshop XML4BPM*. GI, 2005, S. 81–94.

[184] Gergely Varró. „Advanced techniques for the implementation of model transformation systems". Diss. Budapest University of Technology und Economics, 2008.

[185] HMW Verbeek, Joos CAM Buijs, Boudewijn F Van Dongen und Wil MP Van Der Aalst. „Xes, xesame, and prom 6". In: *Forum at the Conference on Advanced Information Systems Engineering (CAiSE)*. Springer. 2010, S. 60–75.

[186] Markus Völter, Sebastian Benz, Christian Dietrich, Birgit Engelmann, Mats Helander, Lennart C. L. Kats, Eelco Visser und Guido Wachsmuth. *DSL Engineering - Designing, Implementing and Using Domain-Specific Languages*. dslbook.org, 2013.

[187] R Hevner Von Alan, Salvatore T March, Jinsoo Park und Sudha Ram. „Design science in information systems research". In: *MIS quarterly* 28.1 (2004), S. 75–105.

[188] Jos B Warmer und Anneke G Kleppe. *The object constraint language: getting your models ready for MDA*. Addison-Wesley Professional, 2003.

[189] Ronald L. Wasserstein und Nicole A. Lazar. „The ASA's Statement on p-Values: Context, Process, and Purpose". In: *The American Statistician* 70.2 (2016), S. 129–133.

[190] AJMM Weijters, Wil MP van Der Aalst und AK Alves De Medeiros. *Process mining with the heuristics miner-algorithm*. Techn. Ber. Eindhoven University of Technology, 2006.

[191] AJMM Weijters und JTS Ribeiro. „Flexible heuristics miner (FHM)". In: *IEEE Symposium on Computational Intelligence and Data Mining (CIDM)*. IEEE. 2011, S. 310–317.

[192] Jan Martijn EM van der Werf, Boudewijn F van Dongen, Cor AJ Hurkens und Alexander Serebrenik. „Process discovery using integer linear programming". In: *International Conference on Applications and Theory of Petri Nets*. Springer. 2008, S. 368–387.

[193] Michael Westergaard. „CPN tools 4: Multi-formalism and extensibility". In: *Application and Theory of Petri Nets and Concurrency*. Springer, 2013, S. 400–409.

[194] Michael Westergaard und Tijs Slaats. „Mixing paradigms for more comprehensible models". In: *Business Process Management*. Springer, 2013, S. 283–290.

[195] Manuel Wimmer und Gerhard Kramler. „Bridging grammarware and modelware". In: *International Conference on Model Driven Engineering Languages and Systems*. Springer. 2005, S. 159–168.

[196] Moe Thandar Wynn, HMW Verbeek, Wil MP van der Aalst, Arthur HM ter Hofstede und David Edmond. „Reduction rules for YAWL workflows with cancellation regions and OR-joins". In: *Information and Software Technology* 51.6 (2009), S. 1010–1020.

[197] JianHong Ye, ShiXin Sun, Lijie Wen und Wen Song. „Transformation of BPMN to YAWL". In: *International Conference on Computer Science and Software Engineering*. Bd. 2. IEEE. 2008, S. 354–359.

[198] Rina Zazkis und Egan J. Chernoff. „What makes a counterexample exemplary?" In: *Educational Studies in Mathematics* 68.3 (2008), S. 195–208.

[199] Michael Zeising. „Plattform zur integrierten IT-gestützten Ausführung von strikten und agilen Prozessen". Diss. Universität Bayreuth, 2015.

[200] Michael Zeising, Stefan Schönig und Stefan Jablonski. „Towards a common platform for the support of routine and agile business processes". In: *International Conference on Collaborative Computing: Networking, Applications and Worksharing (CollaborateCom)*. Bd. 10. IEEE. 2014, S. 94–103.

[9] Lars Ackermann. *E-Mail Semantics: Semantische Suche auf entsprechend der Security Policy freigegebenen E-Mail-Postfächern*. AV Akademikerverlag, 2012.

[10] Lars Ackermann, Jonathan Owens und Lutz Lukas. „The Bayreuth Morphological Segmenter: A work in progress". In: *Lisan Al-Arab* 9 (2017), S. 11.

[11] Lars Ackermann und Stefan Schönig. „MuDePS: Multi-perspective Declarative Process Simulation." In: *Online Proceedings of the BPM Demo Track*. CEUR-WS.org. 2016, S. 12–16.

[12] Lars Ackermann, Stefan Schönig und Stefan Jablonski. „Inter-Paradigm Translation of Process Models using Simulation and Mining". In: *CoRR* (2016).

[201] Lars Ackermann, Stefan Schönig und Stefan Jablonski. „Simulation of multi-perspective declarative process models". In: *Business Process Management Workshops*. Springer. 2016, S. 61–73.

[13] Lars Ackermann, Stefan Schönig und Stefan Jablonski. „Towards Simulation- and Mining-Based Translation of Process Models". In: *Workshop on Enterprise and Organizational Modeling and Simulation (EOMAS)*. Bd. 12. Springer. 2016, S. 3–21.

[14] Lars Ackermann, Stefan Schönig und Stefan Jablonski. „Towards Simulation- and Mining-based Translation of Resource-aware Process Models". In: *Business Process Management Workshops*. Springer. 2016, S. 359–371.

[15] Lars Ackermann und Bernhard Volz. „Model [NL] generation: Natural language model extraction". In: *ACM workshop on Domain-specific modeling*. ACM. 2013, S. 45–50.

[16] Lars Ackermann, Stefan Schönig, Michael Zeising und Stefan Jablonski. „Natural Language Generation for Declarative Process Models". In: *Workshop on Enterprise and Organizational Modeling and Simulation (EOMAS)*. Bd. 11. Springer. 2015, S. 3–19.

[24] Michaela Baumann, Michael Heinrich Baumann, Lars Ackermann, Stefan Schönig und Stefan Jablonski. „Ansätze zum Ähnlichkeitsabgleich von deklarativen Geschäftsprozessmodellen". In: *GI-Jahrestagung*. GI, 2016, S. 733–738.

[162] Stefan Schönig, Lars Ackermann und Stefan Jablonski. „DPIL Navigator 2.0: Multi-Perspective Declarative Process Execution". In: *Online Proceedings of the BPM Demo Track*. CEUR-WS.org. 2017.

Das beste Modell für eine Katze ist eine Katze; möglichst dieselbe Katze.
*— Norbert Wiener (1894 – 1964), Begründer der Kybernetik*

# DANKSAGUNG

Eine Promotion ist nicht nur eine fachliche Herausforderung. Sie stellt auch einen hohen Anspruch an Disziplin sowie die Fähigkeit, Berufliches und Privates gut zu organisieren. Ohne die wohlwollende Unterstützung aus dem eigenen Umfeld ist es schwer, ein solches Vorhaben vor dem Scheitern zu bewahren. So ist es mir eine große Freude, mich bei den Menschen zu bedanken, die zum Gelingen dieser Dissertation beigetragen haben.

Mein erster Dank gilt meinem Doktorvater Prof. Dr.-Ing. Stefan Jablonski, der mir die Chance auf eine Promotion eröffnete und mich von der Entwicklung des Themas über den finalen Federstrich in der Ausarbeitung bis hin zum letzten Wort im Kolloquium betreut und begleitet hat. Seine Anregungen, Hinweise und zuweilen auch rigorose Kritik waren für meinen nun erfolgreichen Abschluss richtungsweisend. Mit Offenheit und Verständnis gelang es ihm, mich auch in den schwierigen Phasen der Promotionsjahre zu motivieren.

Weiterer Dank richtet sich an meine Lehrstuhl-Kollegen. Besonders in der Anfangszeit haben mir Florian Gillitzer, Matthias Jahn, Bastian Roth, Stefan Schönig und Michael Zeising mit Rat und Tat zur Seite gestanden. Stefan sei hierbei besonders gedankt. Aus der Zusammenarbeit mit ihm sind unter anderem die für meine Arbeit wesentlichen Publikationen hervorgegangen. Weiterer Dank geht an Kerstin Haseloff und Bernd Schlesier, die mir nicht zuletzt bei organisatorischen respektive technischen Belangen eine große Hilfe waren. Für die Übernahme des Zweitgutachtens und die wertvollen Anmerkungen während unseres Treffens in Essen möchte ich an dieser Stelle Prof. Dr. Ulrich Frank danken.

Besonderer Dank gebührt Dr. Michael Alvers, der als Erster sein Vertrauen in mich gesetzt hat. Sein eigenes Beispiel in Sachen Zielstrebigkeit und Motivation inspirierten mich bereits während meines Studiums in Dresden.

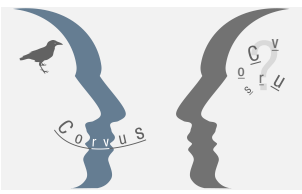
Ein riesengroßes Dankeschön möchte ich meiner Familie, allen voran meinen Eltern Andrea und Falk, meinem Bruder Felix sowie Frank, Steffi und meiner Oma Sieglinde aussprechen. Egal ob es um Entscheidungen oder die Bewältigung der unterschiedlichsten Schwierigkeiten ging, ihre Hilfe war mir stets gewiss. Besonders beim Korrekturlesen dieser Arbeit und deren gestalterischer Prüfung waren meine Mutter, mein Bruder und Frank eine unschätzbare Hilfe. Gleiches gilt auch für meinen langjährigen Freund Kristian, der mir zudem immer wieder mein Ziel vor Augen hielt. Felix Schwägerl möchte ich für wertvolle Denkanstöße, Diskussionen und die kritische Begutachtung zentraler Argumentationsketten danken.

Abschließend möchte ich meiner Freundin Melanie von ganzem Herzen danken. Neben Ihrem grenzenlosen Verständnis für meine nächtlichen Schreibtischaufenthalte, ihrer Unterstützung bei der Bewältigung des unaufschiebbaren Alltags sowie ihren aufbauenden Worten in den unvermeidlichen Motivationstälern hatte sie noch jederzeit ein Lächeln für mich übrig.

Danke euch allen für eure Unterstützung!

KOLOPHON

Diese Dissertation wurde mittels des von André Miede entwickelten typographischen Look-and-Feel `classicthesis` angefertigt. Die Vorlage wurde von Robert Bringhurst's bahnbrechendem Buch über Typographie "*The Elements of Typographic Style*" maßgeblich beeinflusst.




*Sprachzentrierte Ansätze zur Steigerung der Akzeptanz von Geschäftsprozessmodellen*
Lars Ackermann, DISSERTATION, © 2018