Lehrstuhl für
Wirtschaftsinformatik

Information Systems
Management

No. A1/1998

January 1998

Bayreuther Arbeitspapiere zur Wirtschaftsinformatik

Michael Zapf, Armin Heinzl

Ansätze zur Integration von objektorientierten Konzepten und Petri-Netzen

Bayreuth Reports on Information Systems Management

UNIVERSITÄT
BAYREUTH

ISSN 1864-9300

Die Arbeitspapiere des Lehrstuhls für Wirtschaftsinformatik dienen der Darstellung vorläufiger Ergebnisse, die i. d. R. noch für spätere Veröffentlichungen überarbeitet werden. Die Autoren sind deshalb für kritische Hinweise dankbar.

The Bayreuth Reports on Information Systems Management comprise preliminary results which will usually be revised for subsequent publications. Critical comments would be appreciated by the authors.

Authors:

Michael Zapf
Armin Heinzl

Information Systems Management Working Paper Series

Edited by:

Prof. Dr. Armin Heinzl

Managing Assistant and Contact:

Universität Bayreuth
Lehrstuhl für Wirtschaftsinformatik (BWL VII)
Prof. Dr. Torsten Eymann
Universitätsstrasse 30
95447 Bayreuth
Germany

Email: wi@uni-bayreuth.de

ISSN 1864-9300

UNIVERSITÄT
BAYREUTH

Lehrstuhl für
Betriebswirtschaftslehre VII
Wirtschaftsinformatik

Ansätze zur Integration von objektorientierten Konzepten und Petri-Netzen

Michael Zapf, Armin Heinzl

Arbeitspapier 1/1998
(Fassung vom Januar 1998)

Arbeitspapiere Wirtschaftsinformatik

Herausgeber: Prof. Dr. Armin Heinzl

Universität Bayreuth
Lehrstuhl für Betriebswirtschaftslehre VII (Wirtschaftsinformatik)
Universitätsstraße 30, D-95440 Bayreuth
Telefon 0921/55-2807, Telefax 0921/55-2216
E-Mail: wi@uni-bayreuth.de
Internet: http://wi.oec.uni-bayreuth.de

Ansätze zur Integration von objektorientierten Konzepten und Petri-Netzen

Michael Zapf, Armin Heinzl[1]

Abstract

Der Beitrag gibt einen systematischen Überblick über verfügbare Ansätze zur Integration von Objekten und Petri-Netzen. Dabei geht es weniger um die Analyse und Bewertung von Integrationsansätzen im einzelnen, sondern vielmehr darum, grundlegende Integrationsrichtungen und ihre spezifischen Eigenschaften herauszuarbeiten sowie Anhaltspunkte zum daraus resultierenden Nutzenpotential für die Entwicklung betrieblicher Informationssysteme zu gewinnen.

Stichworte

Petri-Netze, Objektorientierte Analyse, Objektorientierter Entwurf, Modellierung von Geschäftsprozessen, Koppelung statischer und dynamischer Systemeigenschaften

[1] Dipl.-Wirtschaftsmath. Michael Zapf, Prof. Dr. Armin Heinzl, Universität Bayreuth, Lehrstuhl BWL VII (Wirtschaftsinformatik), E-Mail: {zapf, heinzl}@uni-bayreuth.de

1 Einleitung

Im Vergleich zu früheren Ansätzen stellt die objektorientierte Software-Entwicklung neuartige Konzepte für die Entwicklung betrieblicher Informationssysteme (IS) zur Verfügung. Aufbauend auf den Erkenntnissen aus der objektorientierten Programmierung wurden unterschiedliche Methoden zur Analyse und zum Entwurf von Informationssystemen entwickelt. Zur Systemdarstellung finden grafische Notationen Verwendung, in deren Mittelpunkt die Beschreibung von Objekten steht. Das Gesamtsystem ergibt sich als eine Menge interagierender Objekte, die nicht zentral gesteuert, sondern mit Hilfe von gegenseitigem Nachrichtenaustausch dezentral koordiniert werden. Gleichartige Objekte, die identische oder ähnliche Eigenschaften und Verhaltensmuster aufweisen, werden in Klassen zusammengefaßt. Die Darstellung und Anordnung von Klassen wird in allen Ansätzen mit ähnlichen Konstrukten unterstützt und stellt ein wesentliches Hilfsmittel zur Abbildung statischer Systemstrukturen dar.

Die Abbildung dynamischer Systemstrukturen erfolgt dagegen weniger einheitlich. Zwar lassen sich Möglichkeiten zur Beschreibung der Interaktion zwischen Objekten sowie der Darstellung von Objektlebenszyklen konstatieren, jedoch sind diese Techniken entweder zu stark auf einzelne Objekte fixiert oder nicht ausreichend formal fundiert. Sie eignen sich deshalb nur bedingt, um das Verhalten komplexer Systeme exakt darzustellen und zu analysieren. Insofern überrascht es kaum, daß zunehmend Petri-Netze aufgegriffen werden, um sie mit objektorientierten Ansätzen zu verknüpfen. Petri-Netze eignen sich zur grafischen Modellierung, Analyse und Simulation dynamischer Systeme. Sie besitzen ein theoretisches Fundament und werden bereits erfolgreich bei der Modellierung und Ausführung von IS, insbesondere von Workflowsystemen, angewendet [Ober96].

Der vorliegende Beitrag gibt einen systematischen Überblick über verfügbare Ansätze zur Integration von Objekten und Petri-Netzen. Dabei geht es weniger um die Analyse und Bewertung von Integrationsansätzen im einzelnen, sondern vielmehr um das Herausarbeiten grundlegender Integrationsrichtungen, die Darstellung ihrer Eigenschaften und um die Gewinnung von Anhaltspunkten des daraus resultierenden Nutzenpotentials für die Entwicklung betrieblicher Informationssysteme.

Zu diesem Zweck werden zunächst relevante Grundlagen der Entwicklung betrieblicher Informationssysteme behandelt. Anschließend erfolgt eine Darstellung charakteristischer Eigenschaften der einzelnen Integrationsrichtungen. Dabei wird jeweils eine Methode exemplarisch herausgegriffen, an einem einfachen Beispiel veranschaulicht und eine Übersicht über weitere Vertreter der jeweiligen Kategorie gegeben. Am Ende wird ein Überblick über die jeweiligen Anwendungsgebiete der unterschiedlichen Ansätze gegeben, ein kurzes Fazit gezogen und auf zukünftige Untersuchungsschwerpunkte hingewiesen.

2 Entwicklung betrieblicher Informationssysteme

2.1 Systemsichten und Entwicklungsphasen

Die Struktur eines Systems wird als Menge von Elementen bzw. Komponenten verstanden, die miteinander in Beziehung stehen [Patz82, 39]. Auf dieser Definition aufbauend lassen sich für betriebliche IS unterschiedliche Sichten entwickeln [FeSi98, 125]. Wird der Systeminhalt aus der Perspektive sachlicher Zusammenhänge betrachtet, so stehen die *statischen Beziehungen* der Systemelemente im Vordergrund. Als Elemente lassen sich Funktionen (Funktionssicht), Datenstrukturen (Datensicht) und Kommunikationskanäle (Interaktionssicht) konstatieren. Ist dagegen die zeitliche Anordnung von Elementen von Interesse, so steht das *dynamische*

Verhalten von Systemen im Mittelpunkt. In dieser Sicht werden betriebliche Vorgänge oder Prozesse (Vorgangs- bzw. Prozeßsicht) betrachtet. Die Koppelung unterschiedlicher Sichten stellt eine anspruchsvolle, aber wichtige Voraussetzung zur Entwicklung von IS dar. Dabei ist einerseits die Verknüpfung der statischen und dynamischen Sicht und andererseits die Verbindung der Daten-, Funktions- und Interaktionssicht bedeutsam.

Bei der Entwicklung betrieblicher IS lassen sich zudem drei phasenförmig angeordnete Aufgabenkerne identifizieren [Balz96, 92]. In der Analyse- oder Definitionsphase (analysis) werden zunächst die Anforderungen an das zu entwickelnde IS ermittelt, beschrieben und analysiert. Darauf aufbauend erfolgt in der Entwurfsphase (design) die Spezifikation softwaretechnischer Komponenten im Sinne einer Softwarearchitektur. Anhand dieser Vorgaben werden in der Implementierungsphase (implementation) die eigentlichen Programmiertätigkeiten durchgeführt. Da die Ergebnisse vorgelagerter Phasen als Input für nachgelagerte Phasen dienen, tritt ebenfalls eine Verknüpfungsnotwendigkeit auf. Die mit Hilfe einschlägiger Methoden definierten Anforderungen sollen jeweils ohne Reibungsverluste in der Entwurfsphase weiterverarbeitet und die so spezifizierten Entwürfe nahtlos in der Implementierung berücksichtigt werden. Diese Eigenschaft wird auch als „weicher“ Phasenübergang bezeichnet [Sinz91].

2.2 Objektorientierte Ansätze zur Entwicklung betrieblicher Informationssysteme

Konventionelle Methoden zur Entwicklung betrieblicher Informationssysteme, wie z.B. SA/SD [DeMa78] oder ERM [Chen76], gehen von einer Trennung der Funktions- und Datensicht aus und unterstützen lediglich die Analyse- und Entwurfsphase [Fers97, 289]. Objektorientierte Ansätze betrachten dagegen Objekte, in denen ihre Eigenschaften (Attribute), darauf ausführbare Operationen (Methoden) und Nachrichten für die Kommunikation zwischen Objekten gekapselt sind. Objekte mit gleichen Eigenschaften werden zu Klassen zusammengefaßt. Die Definition einer Klasse beinhaltet die Strukturbeschreibung aller zugehörigen Objekte. Zwischen den Klassen können hierarchische Anordnungen durch Generalisierung und Aggregation vorgenommen werden. Bei der Generalisierung werden bestimmte Eigenschaften einer Oberklasse an eine Unterklasse vererbt [Balz96, 201].

Die damit verbundenen Vorteile sind eine bessere Nachvollziehbarkeit der definierten statischen Systemanforderungen durch die Integration von Funktions-, Daten- und Interaktionssicht, „weiche“ Übergänge infolge der Nutzung identischer Modellierungs- bzw. Gestaltungselemente in den Entwicklungsphasen und eine höhere Wiederverwendbarkeit von Systemkomponenten aufgrund der Kapselungs- und Vererbungsmöglichkeiten.

Für die Beschreibung dynamischer Systemstrukturen stehen eine Vielzahl von Methoden zur Verfügung, deren wichtigste Vertreter in der Unified Modeling Language (UML) zusammengeführt worden sind [Burk97]. Dort wird mit Hilfe von Kollaborations- und Interaktionsdiagrammen der Nachrichtenfluß zwischen Objekten und die zeitliche Abfolge von Methodenaufrufen spezifiziert. Allerdings können dabei keine weiteren Regeln zur Prozeßsteuerung, wie etwa Entscheidungsoperatoren, aufgenommen werden. Dies führt unter anderem dazu, daß sich nur sequentielle Abläufe und keine alternativen Prozeßverläufe abbilden lassen [NüZi97]. Diese Mängel werden in Zustands- und Aktivitätsdiagrammen behoben, die beide auf dem Prinzip der Statecharts von Harel [Hare87] basieren. Während Zustandsdiagramme ausschließlich zur Modellierung von Lebenszyklen einzelner Objekte eingesetzt werden können, eignen sich Aktivitätsdiagramme auch zur Darstellung von objektübergreifenden Abläufen [Rati97]. Allerdings ist die Semantik dieser Diagrammtypen nicht ausreichend formal definiert, um dynamische Systemeigenschaften exakt zu spezifizieren. Gleichzeitig existieren keine Mechanismen, um Zusammenhänge zwischen unterschiedlichen Diagrammen darzustellen.

Somit stehen in objektorientierten Ansätzen bislang nur bedingt geeignete Methoden zur Spezifikation von dynamischen Eigenschaften komplexer Systeme zur Verfügung, insbesondere wenn nebenläufige oder nichtdeterministische Vorgänge zu beschreiben sind. Die Koppelung der statischen mit der dynamischen Sicht wird daher von den objektorientierten Ansätzen nur teilweise als unterstützt angesehen [FeSi98, 125].

2.3 Petri-Netze als Instrument zur Beschreibung dynamischer Systemstrukturen

Petri-Netze stellen gerichtete Graphen dar, die aus Knoten (Stellen) und Kanten (Transitionen) bestehen. Die Stellen bilden Zustände oder Bedingungen ab und die Transitionen repräsentieren Zustandsübergänge oder Ereignisse. Dabei müssen auf Stellen stets Transitionen folgen und umgekehrt. Für die Beschreibung der Systemdynamik werden Marken verwendet, die in Verbindung mit einer Schaltregel durch das Netz wandern. Eine Transition kann dann schalten, wenn jede vorgelagerte Stelle mindestens eine Marke enthält. In diesem Fall wird eine Marke aus jeder Stelle des Vorbereichs der Transition entfernt und jeder Stelle im Nachbereich hinzugefügt [Baum90; Reis85; RoWi91].

Mit Hilfe der zulässigen Anzahl und der Art der Marken lassen sich unterschiedliche Netztypen abgrenzen. Im Rahmen dieser Arbeit sind höhere Netztypen (z.B. Prädikate/Transitionen-Netze [GeLa81; Genr86] oder gefärbte Petri-Netze [Jens96]) von besonderem Interesse, da ihre Marken als Objekte mit konkreten Eigenschaften aufgefaßt und den Transitionen Schaltbedingungen mit Variablen (im Sinne von Verarbeitungsregeln) zugeordnet werden können.

Als wesentliche Vorteile der Petri-Netze lassen sich ihr theoretisches Fundament, der begrenzte Symbolvorrat und ihre grafische Darstellungsmöglichkeit nennen. Diese Eigenschaften ermöglichen eine Visualisierung aller ausführbaren Systemzustände, die formale Analyse und Simulation modellierter Prozesse und die Generierung von Programm-Code. Bei höheren Netztypen wird kritisiert, daß sie schwer zu erstellen bzw. zu analysieren sind. Sie bleiben einschlägig spezialisierten und geübten Modellierern vorbehalten. Zudem bemängeln einige Autoren, daß Petri-Netze ein isoliertes Konzept darstellen, das bis dato mit anderen Methoden nur sporadisch kombiniert werden kann [Balz96, 319; Zele95, 72].

3 Ansätze zur Integration von Objekten und Petri-Netzen

Die nachfolgend behandelten Ansätze integrieren objektorientierte Konzepte und Petri-Netze. Sie lassen sich nach der Integrationsrichtung in folgende Kategorien einordnen [Bast95]:

- Ansätze zur Einbettung von Objekten in Petri-Netze,
- Ansätze zur Einbettung von Petri-Netzen in Objekte und
- Ansätze zur beidseitigen Integration von Objekten und Petri-Netzen.

Aus jeder Kategorie wird ein repräsentativer Vertreter herausgegriffen, kurz vorgestellt und anhand der Modellierung eines einfachen Logistikbeispiels veranschaulicht. Die Auswahl eines repräsentativen Vertreters ist kein leichtes Unterfangen. Im vorliegenden Zusammenhang ist dabei die Anschaulichkeit der Verfahren, die Klarheit des Integrationskonzeptes und der Bekanntheitsgrad der verwendeten Notationen ausschlaggebend. Damit wird weder eine Wertung vorgenommen noch eine Aussage über die Verbreitung oder die sonstigen Eigenschaften der Arbeiten getroffen.

Nach der Vorstellung des jeweiligen Beispielansatzes erfolgt eine allgemeine Charakterisierung des betreffenden Integrationskonzeptes und eine kurze Übersicht über andere Vertreter der entsprechenden Kategorie.

3.1 Ein einfaches Logistiksystem als Anwendungsbeispiel

Zur Beschreibung der Ansätze wird ein durchgängiges Anwendungsbeispiel aus der Logistik verwendet, das sich mit dem Transport von Containern beschäftigt [Verk93, 22]:

Ein Transportunternehmen besitzt mehrere LKW mit unterschiedlichen Frachtkapazitäten. Diese Fahrzeuge können von verschiedenen Fahrern gesteuert werden, die jedoch eine spezielle Fahrerlaubnis für den jeweiligen LKW benötigen. Bevor ein Container transportiert werden kann, muß ein entsprechender Transportauftrag vom Kunden eingegangen sein. Anhand dieses Auftrages wird einem LKW mit ausreichender Kapazität ein entsprechender Fahrer zugeordnet. Nach dem Beladen des LKW mit den benötigten Containern wird die Fahrt durchgeführt und der LKW am Bestimmungsort entladen. Abschließend findet anhand der Auftragspapiere eine Kontrolle und Fakturierung des Transportes statt.

3.2 Einbettung von Objekten in Petri-Netze

3.2.1 Der SimCon-Ansatz

Der SimCon-Ansatz (**S**imple **I**ntegrated **M**odel for **C**omplex **O**bject **N**etworks) zeichnet sich durch eine anschauliche graphische Notation und ein klares Konzept zur Einbettung von Objekten in Petri-Netze aus [Verk93]. SimCon wurde speziell für die integrierte Modellierung von Informationssystemen entwickelt. Der dabei verwendete Integrationsbegriff bezieht sich auf zwei verschiedene Modellierungsaspekte: Zum einen wird die Koppelung von statischen und dynamischen Systemeigenschaften durch die Einbettung von Objekten in Petri-Netze angestrebt und zum anderen soll die Verwendung derselben Modellierungstechniken in der Analyse- und der Entwurfsphase zu reibungslosen Phasenübergängen im Softwareentwicklungsprozeß führen.

SimCon ist als Bestandteil des ExSpect-Projektes entstanden (**Ex**ecutable **Spec**ification **T**ool), das Ende der 80er Jahre mit der Zielsetzung ins Leben gerufen wurde, einen formalen Rahmen für den Entwurf von Informationssystemen zu entwickeln [HeSV88; HeSV89; HeSV91a; HeRV93; AaWa91].

Modellierungskomponenten

Die Systemmodellierung erfolgt mit Hilfe von drei Komponenten:

- einem *Objektmodell* zur Definition von Objekteigenschaften und wechselseitigen Beziehungen zwischen unterschiedlichen Objekttypen (SimCon Object Model),
- einem *Prozeßmodell* zur Darstellung von objektinternen und -übergreifenden Abläufen (SimCon Net) und
- einer *Manipulationssprache* zur Formulierung von Veränderungen an Objekten (SimCon Algebra).

Bild 1 zeigt ein *Objektmodell* für das eingangs beschriebene Transportproblem, das sich aus zwei Schichten zusammensetzt:

1. Die erste Schicht besteht aus einfachen Objekttypen, die mit Hilfe von Rechtecken mit abgerundeten Ecken dargestellt werden. Diverse Transportaufträge werden z.B. durch den Objekttyp „Auftrag“ modelliert. Dessen Eigenschaften, wie etwa die Auftragsmenge („Menge“), können durch Kreise visualisiert werden, die über Kanten mit dem zugehörigen Objekttyp verbunden sind. Für Beziehungen zwischen Objekttypen werden gerichtete Kanten (Pfeile) verwendet und entsprechend beschriftet. Unterschiedliche Kardinalitäten von Beziehungen werden dabei mit verschiedenen Pfeilspitzen symbolisiert: Ein „Fahrer“ darf mehrere „LKW“ fahren, ein „Auftrag“ wird dagegen mit genau einem „LKW“ abgewickelt.

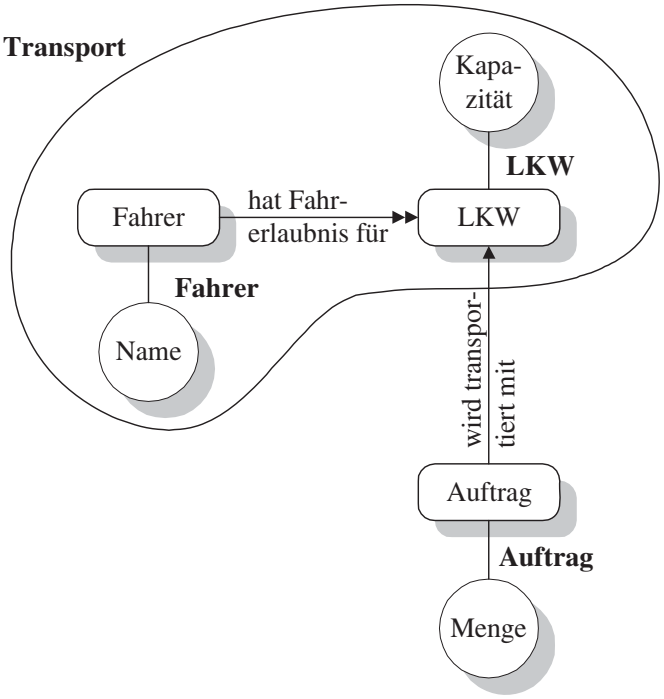

Bild 1 Ein Objektmodell in SimCon
Quelle: In Anlehnung an [Verk93, 22]

2. In der zweiten Schicht können einfache Objekttypen zu sogenannten Containerobjekten aggregiert werden. Solche Aggregationen ermöglichen eine einfache Koppelung von Objekt- und Prozeßmodellen. Für die Transportplanung werden die einfachen Objekte „Fahrer“ und „LKW“ zu einem Containerobjekt „Transport“ zusammengefaßt.

Die *Prozeßmodellierung* erfolgt in SimCon mit einem speziellen Petri-Netztyp, in dem Objekte als Marken durch das Netz bewegt werden. Die Stellen des Netzes sind fest mit einzelnen Objekttypen aus dem Objektmodell verknüpft und können ausschließlich Objekte diesen Typs aufnehmen. In Bild 2 ist ein Prozeßmodell für die Abwicklung von Transportaufträgen dargestellt. Im Netz besitzt jede Stelle eine Stellenbezeichnung und einen zugehörigen Objekttyp, der in Klammern hinter der Bezeichnung angegeben ist. Im „Warteraum“ befinden sich etwa Objekte des Typs „Fahrer“, während in der „Garage“ verschiedene „LKW“ abgestellt werden können. Falls auf einer Stelle unterschiedliche Objekte gespeichert werden sollen, müssen diese vorher in einem Containerobjekt zusammengefaßt werden. Bei der Abwicklung eines Transportes ist es zum Beispiel zweckmäßig , jedem „LKW“ einen „Fahrer“ zuzuordnen und diese anschließend gemeinsam auf die Reise zu schicken. Deshalb wurden die zwei Objekttypen „LKW“ und „Fahrer“ im Containertyp „Transport“ zusammengefaßt und anschließend den Stellen „LKW mit Fahrer“, „Beladener LKW“ und „Leerer LKW“ zugeordnet.

Zur Steuerung eines Ablaufes werden die Transitionen eines Netzes mit Hilfe einer *Manipulationssprache*, der SimCon Algebra, beschriftet. Diese Sprache kann zum Erzeugen, Verändern und Löschen von Objekten, aber auch zur Formulierung von speziellen Schaltbedingungen eingesetzt werden. In der Transition „Transport zusammenstellen“ soll zum Beispiel sichergestellt werden, daß die Kapazität des „LKW“ für den „Auftrag“ ausreicht und daß der „Fahrer“ eine gültige Fahrerlaubnis für den „LKW“ besitzt. Diese beiden Bedingungen können mit der graphischen Erweiterung der SimCon Algebra wie folgt formuliert werden:

$$\text{Auftrag}[\mathit{Menge}] \leq \text{LKW}[\mathit{Kapazität}] \wedge \text{LKW}[] \subseteq \text{LKW}[\xleftarrow{\textit{hat Fahrerlaubnis für}} \text{Fahrer}[]]$$

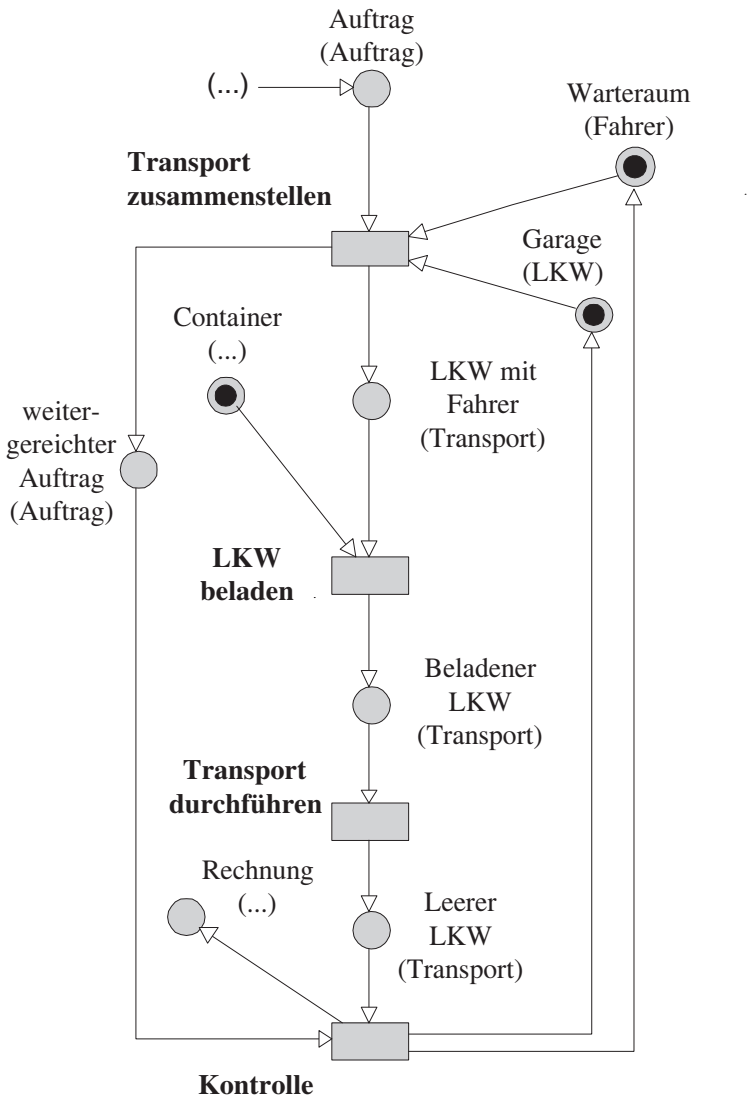

Bild 2 Ein Prozeßmodell in SimCon[2]
Quelle: In Anlehnung an [Verk93, 22]

Softwareentwicklungsphase und Vorgehensmodell

Der SimCon-Ansatz wurde zur Unterstützung der *Analyse-* und *Entwurfsphase* entwickelt. Der Schwerpunkt liegt dabei auf dem reibungslosen Übergang von der Analyse zum Entwurf, der – wie eingangs erwähnt – durch den Einsatz der selben Modellierungstechnik in beiden Phasen erreicht werden soll. Für die Phase der Anforderungsanalyse existieren keine Hilfsmittel.

In [Verk93] wird für die Anwendung von SimCon ein *prozeßorientiertes*, ein *datenorientiertes* und ein *kombiniertes Vorgehensmodell* vorgestellt. Als Spezialfall der kombinierten Vorgehensweise unterstützt das **Sim**Con/**I**ntegrated **O**bject **M**odel eine objektorientierte Systemmodellierung, bei der Objektlebenszyklen mit Hilfe von objektspezifischen Prozeßnetzen dargestellt werden.

[2] Aus Platzgründen kann nicht das vollständige Prozeßmodell angegeben werden. Die nicht näher spezifizierten Elemente sind mit „(...)“ gekennzeichnet.

3.2.2 Übersicht über bestehende Ansätze

Seit Beginn der neunziger Jahre sind eine Reihe von Ansätzen zur Einbettung von Objekten in Petri-Netze entwickelt worden (vgl. Tabelle 1). Petri-Netze werden dabei als globaler Steuerungs- und Kontrollmechanismus für IS verwendet, während die statischen Systemeigenschaften hauptsächlich durch die Struktur der Marken bestimmt werden. Die Stellen eines Netzes können Objekte aufnehmen, in den Transitionen werden Objekte manipuliert, Objektmethoden ausgeführt oder auf Objekteigenschaften zugegriffen (vgl. Bild 3).

Die Definition der Markenstrukur erfolgt entweder über eine graphische Notation (z.B. Macronet, NetCASE, OPM, SimCon) oder eine Spezifikationssprache (z.B. LOOPN, OOCPN, OBJSA). Häufig werden dabei weit verbreitete Techniken und Sprachen direkt oder in leicht modifizierter Form eingesetzt, wie etwa die Entity-Relationship-Modellierung [Chen76], die Object Modeling Technique (OMT) [RBPE91], die Unified Modeling Language (UML) [Burk97] oder eine objektorientierte Programmiersprache wie EIFFEL [Meye92]. Bei den Petri-Netzen werden häufig eigene Netztypen definiert, die sich teilweise in weiter verbreitete höhere Netztypen transformieren lassen, um deren analytische Eigenschaften nutzen zu können. Einige Autoren stellen aus diesem Grund den Bezug zu gefärbten Netzen, Prädikate/Transitionen- oder Stellen/Transitionen-Netzen her.

Die betrachteten Modellierungstechniken sind fast alle für die Analyse- und Entwurfsphase konzipiert. NetCASE ist der einzige Ansatz, in dem versucht wird, direkt aus den Systemmodellen lauffähige INGRES-Datenbankanwendungen [HeSW75] oder Java-Prototypen [ArGo98] zu generieren, um einen nahtlosen Übergang zur Implementierungsphase zu erreichen. Für den praktischen Einsatz der Arbeiten liefern die Autoren teilweise Anwendungsempfehlungen in Form von Vorgehensmodellen mit. Dabei werden datenorientierte, prozeßorientierte und kombinierte, d.h. hybride Vorgehensweisen vorgeschlagen.

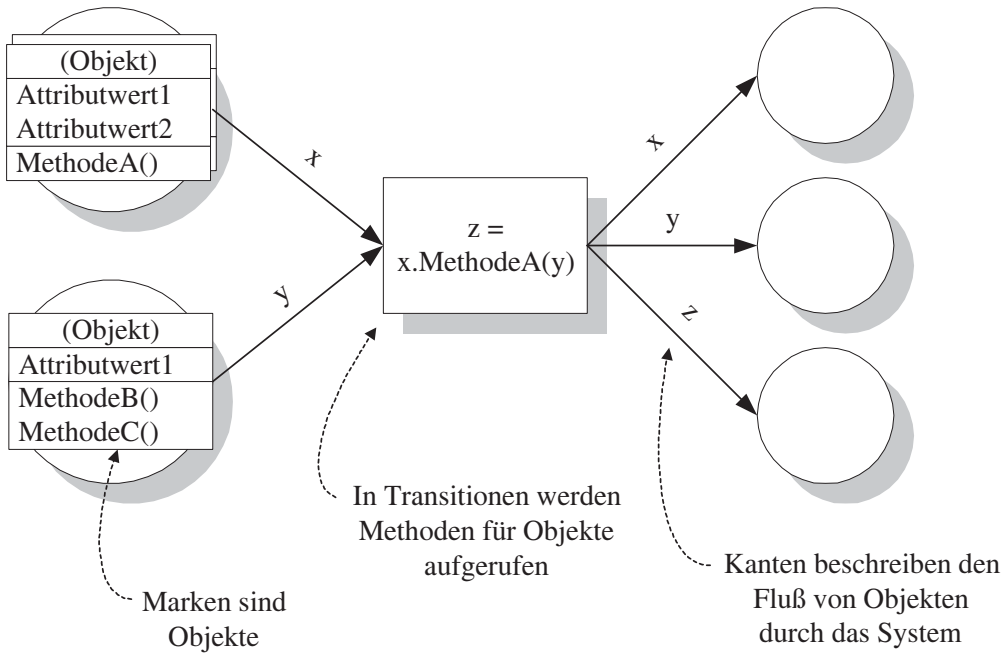

Bild 3 Einbettung von Objekten in Petri-Netze
Quelle: In Anlehnung an [Bast95, 1]

Ansatz	Objekt-orientierte Notation	Netztyp	Entwicklungsphase	Vorgehensmodell	Werkzeug
LOOPN Language for Object-Oriented Petri Nets [LaKe91]	eigene Sprache	gefärbte zeitbehaftete Netze	Entwurf	n.v.	LOOPN [Lako98]
Macronet [KeSB94]	Erweiterung der ER-Not. [Chen76]	eigener Netztyp	Analyse	n.v.	Macrotec
MOBY Modellierung von Bürosystemen [FlLi93]	Smalltalk-80 [GoRo83]	eigener Netztyp	Analyse	n.v.	MOBY
NetCASE Petri Net Based CASE [DLMR94; Marx95; Marx97]	basiert auf OMT [RBPE91]	Unterklasse von Pr/T-Netzen	Analyse, Entwurf und Realisierung	prozeß-orientiert, kombiniert	NEPTUN [Marx98]
OBJSA Nets [BaDM88]	OBJ2 [FGJM85]	eigener Netztyp	Entwurf	n.v.	ONE environment
OOCPN Object-Oriented Coloured Petri Nets [Engl93]	EIFFEL [Meye92]	gefärbte Netze	Entwurf	n.v.	n.v.
OPM Objekt-Prozeß-Modell [Burk94; PhBu95; PhBW95]	UML [Burk97]	Transformation in S/T- oder gefärbte Netze möglich	Analyse, Entwurf	prozeß-orientiert	OTW [Burk98]
SimCon Simple Integrated Model for Complex Object Networks [Verk93]	eigene Notation	eigener Netztyp	Analyse, Entwurf	daten-orientiert, prozeß-orientiert, kombiniert	ExSpect [Bakk98; HeSV91b; HeRV93]
THORN Timed Hierarchical Object-Related Nets [ScSW95]	C++ [Stro92]	eigener Netztyp	Entwurf, Realisierung	n.v.	THORN/DE [Wiet98]

Tabelle 1 Ansätze zur Einbettung von Objekten in Petri-Netze

3.3 Einbettung von Petri-Netzen in Objekte

3.3.1 Der OBM-Ansatz

Als Beispiel für die Einbettung von Petri-Netzen in Objekte wird das **O**bject/**B**ehaviour **M**odel (OBM), eine objektorientierte Diagrammtechnik, erläutert [KaSc91]. Für die Beschreibung der statischen Systemeigenschaften wird dabei eine Notation verwendet, die auf Konzepten der semantischen Datenmodellierung [HaMc81] basiert. Die Modellierung des Systemverhaltens erfolgt mit Diagrammen, die Petri-Netzen ähnlich sind. Sie lassen sich in Prädikate/Transitionen-Netze transformieren.

Modellierungskomponenten

In OBM wird zwischen zwei Typen von Diagrammen unterschieden:

- *Objektdiagramme* (Object Diagrams) stellen die Eigenschaften und Beziehungen zwischen Objekten dar, während
- *Verhaltensdiagramme* (Behaviour Diagrams) zur Modellierung von Objektlebenszyklen einzelner Objekte eingesetzt werden.

In einem *Objektdiagramm* visualisiert man Objekttypen durch Rechtecke. Ein Rechteck mit doppeltem Rahmen deutet an, daß von diesem Typ mehrere Objekte existieren können (vgl. Bild 4). Das Diagramm für unser Transportunternehmen enthält im Prinzip dieselben Systemelemente, die schon beim SimCon-Ansatz spezifiziert worden sind: „Fahrer“, „LKW“, „Aufträge“ und „Transporte“. Objekteigenschaften werden im jeweiligen Objektrechteck notiert. Falls es sich dabei um Attribute mit einem Standardobjekttypen handelt, wie etwa der „Name“ eines „Fahrers“, so wird dieser Typ durch ein Ellipse unterhalb des Attributnamens symbolisiert. Beziehungen zwischen Objekttypen werden nicht durch Pfeile, sondern über zusätzliche Attribute modelliert: Das Attribut „LKW“ im Objekttyp „Fahrer“ zeigt beispielsweise an, daß ein „Fahrer“ die Fahrerlaubnis für mehrere „LKW“ besitzt.

Die Lebenszyklen von Objekten werden in *Verhaltensdiagrammen* spezifiziert. In Bild 5 ist das entsprechende Diagramm für den Objekttyp „Transport“ dargestellt. Aktivitäten werden mit

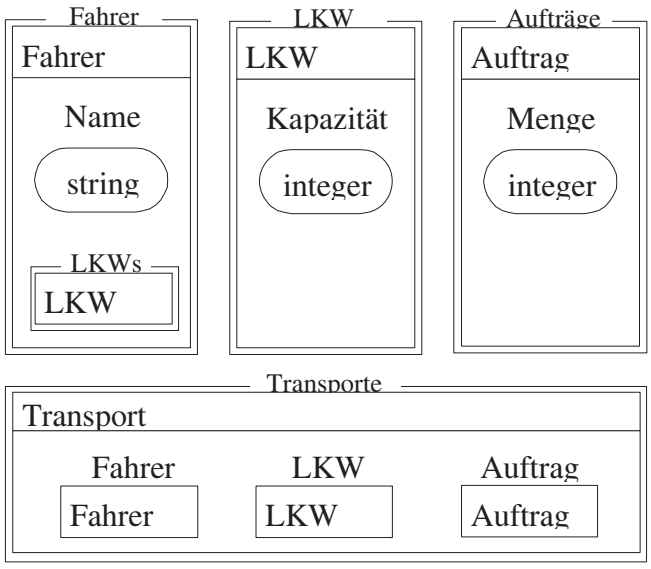

Bild 4 Ein Objektdiagramm in OBM

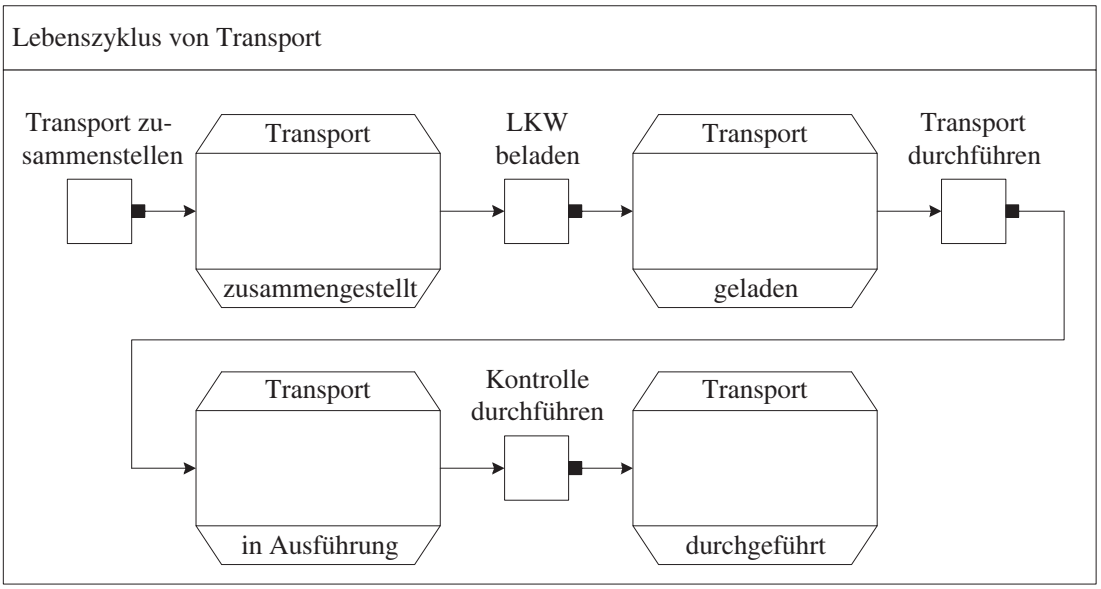

Bild 5 Ein Verhaltensdiagramm in OBM

Hilfe von Quadraten visualisiert und über Pfeile mit den Objektzuständen verbunden. Die Aktivität „Transport zusammenstellen“ führt nach ihrer Ausführung in den Zustand „Transport zusammengestellt“. Am Ende des Lebenszyklus wird ein „Transport“ nicht zerstört, sondern in den Endzustand „Transport durchgeführt“ überführt und steht somit für nachträgliche Analysen zur Verfügung.

Die Koordination von Lebenszyklen mehrerer Objekte wird in speziellen Verhaltensdiagrammen, den sogenannten *Aktivitätsspezifikationsdiagrammen,* beschrieben. Bild 6 zeigt solch ein Spezifikationsdiagramm für die Aktivität „Transport zusammenstellen“ des Objekttyps „Transport“. Hier werden Vorbedingungen und Auswirkungen der Aktivität formuliert: Bevor ein Transport zusammengestellt werden kann, muß ein „Fahrer im Warteraum“, ein „LKW in der Garage“ und ein „Auftrag erteilt“ sein. Nach dem Zusammenstellen ist der „Transport zusammengestellt“ und der entsprechende „Fahrer im LKW“. Durchgezogene Linien werden im Diagramm dann verwendet, wenn die zugehörigen Objekte beim Ausführen der Aktivität ihren Zustand ändern, andernfalls werden gestrichelte Linien benutzt. Die verschiedenen Symbole an den von der Aktivität wegführenden Pfeilen verwendet man zur Unterscheidung von Objekterzeugung und Objektveränderung.

OBM umfaßt noch weitere Diagrammtypen für die exakte Spezifikation von Verhaltensmodellen. Da deren Erläuterung jedoch nicht wesentlich zum Grundverständnis des Ansatzes beiträgt, wird an dieser Stelle darauf verzichtet[3].

Softwareentwicklungsphase und Vorgehensmodell

OBM kommt in der Analyse- und Entwurfsphase zum Einsatz. Allerdings beziehen sich die meisten Diagrammtypen auf die Beschreibung von Objektlebenszyklen und Aktivitäten, die bei anderen Methoden erst in einem fortgeschrittenen Stadium der Analyse verwendet werden. Da die Spezifikation von Aktivitäten sehr aufwendig ist, erscheint dies dann geboten, wenn die Objektstrukturen feststehen und hierbei nicht mehr mit großen Veränderungen gerechnet werden muß.

[3] Der interessierte Leser wird in diesem Zusammenhang auf [KaSc91] verwiesen.

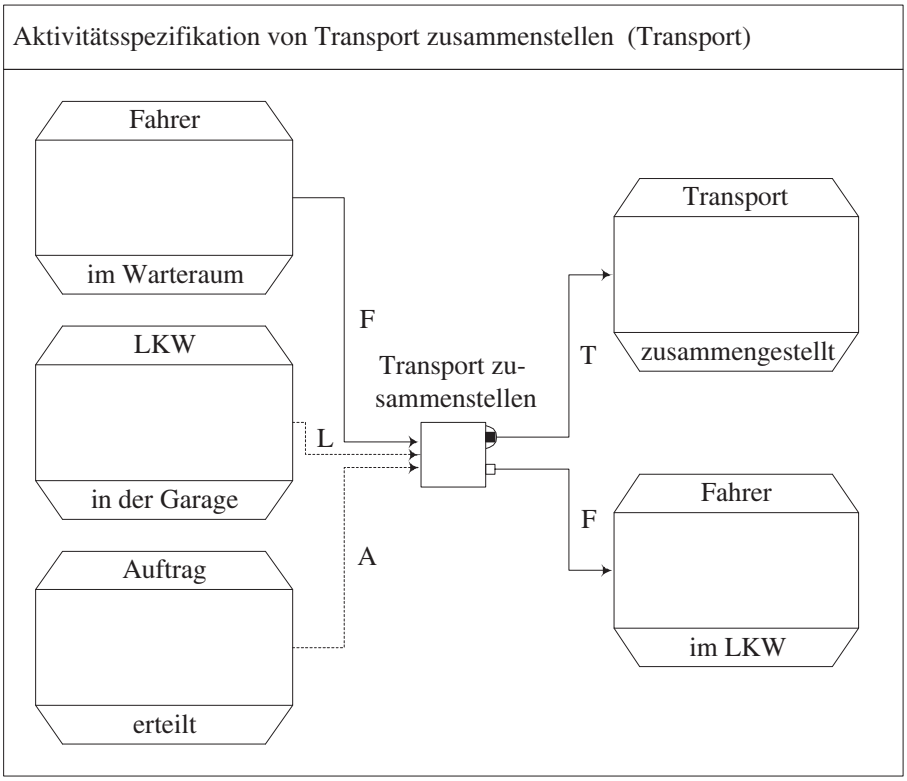

Bild 6 Ein Aktivitätsspezifikationsdiagramm in OBM

Für die Anwendung von OBM eignet sich ein objektorientiertes Vorgehensmodell, allerdings werden hierzu von den Autoren keine weiteren Angaben gemacht.

3.3.2 Übersicht über bestehende Ansätze

Bei der Einbettung von Petri-Netzen in Objekte wird die Strukturierung des Systems mit objektorientierten Techniken vorgenommen: Zunächst werden die relevanten Objekte der Diskurswelt und deren wechselseitige Beziehungen identifiziert, anschließend erfolgt die

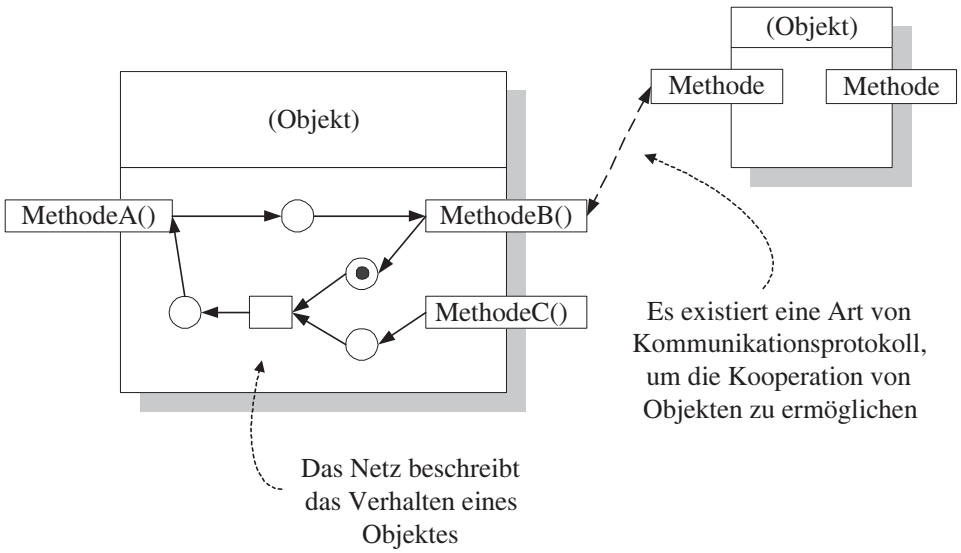

Bild 7 Einbettung von Petri-Netzen in Objekte
Quelle: In Anlehnung an [Bast95, 2]

Beschreibung des Objektverhaltens und der Kommunikation zwischen einzelnen Objekten mit Hilfe von Petri-Netzen (vgl. Bild 7).

Ansatz	Objektorientierte Notation	Netztyp	Entwicklungsphase	Vorgehensmodell	Werkzeug
CLOWN Class Orientation with Nets [BaCD95]	eigene Sprache	eigener Netztyp	Entwurf	objektorientiert	ONE environment
HOOD Nets [DiGi91]	eigene graphische Notation, ADA-Peudocode [Barn84]	höhere Netztypen, z.B. Pr/T-Netze	Entwurf	objektorientiert	n.v.
OBM Object/Behaviour Model [KaSc91]	eigene Notation, basiert auf semantischer Datenmodellierung [HaMc81]	Transformation in Pr/T-Netze möglich	Analyse, Entwurf	objektorientiert	Editor
OCPN Object Coloured Petri Nets [BeMo93; MoMa97]	beliebige objektorientierte Notationen verwendbar	Transformation in gefärbte Netze möglich	Analyse, Entwurf	objektorientiert	n.v.
OOBM Object-Oriented Behaviour Modelling for Real-Time Design [HaDi97]	beliebige objektorientierte Notationen verwendbar	gefärbte Netze	Entwurf	objektorientiert	n.v.
PAM Petri Net based Abstract Machine [BaCo95]	eigene Sprache	gefärbte Netze	Analyse, Entwurf	objektorientiert	n.v.
PROTOB [BaBr88; BaBr91]	eigene Notation und Sprache	eigener Netztyp	Analyse, Entwurf, Realisierung	objektorientiert	PROTOB

Tabelle 2 Ansätze zur Einbettung von Petri-Netzen in Objekte

Einige Ansätze erlauben die Koppelung der einzelnen Objektnetze und somit eine fundierte Analyse des Gesamtsystems. Im Unterschied zu den meisten objektorientierten Programmiersprachen kann durch die Verwendung von Petri-Netzen in Objekten nicht nur sequentielles, sondern auch nebenläufiges Verhalten von Objekten beschrieben werden.

Für die Definition von Objekten wird meist eine eigene objektorientierte Sprache verwendet (vgl. Tabelle 2), die teilweise um eine graphische Notation ergänzt wird (HOOD Nets, PROTOB). OODT benutzt keine Sprache sondern ausschließlich Diagramme, während in OCPN und OOBM keine Vorschriften über die Notation von Objekten gemacht und damit der Einsatz beliebiger Notationen ermöglicht wird. Objektnetze werden mit Hilfe eigener Netztypen oder bekannter höherer Netztypen wie etwa mit Prädikate/Transitionen-Netzen oder gefärbten Netzen modelliert.

Die Zuordnung der Ansätze zu den einzelnen Softwareentwicklungsphasen ist etwas problematisch. Einige Autoren stellen zwar den Bezug ihrer Arbeiten zur Analysephase her, allerdings liegen die jeweiligen Stärken und Einsatzgebiete doch wohl eher in der Entwurfsphase, da über die Strukturierung des Systems in einzelne Objekte meist nichts ausgesagt wird, sondern diese bereits als Ausgangspunkt für die weitere Modellierung vorausgesetzt werden.

3.4 Beidseitige Integration

3.4.1 Vorstellung eines ausgewählten Ansatzes: PN-TOX

Der PN-TOX-Ansatz (**P**etri **N**et **T**ool for **O**bject Concurrency Specification) basiert auf der Idee, Informationssysteme aus einer Menge von unabhängigen, gleichzeitig agierenden Objekten zusammenzusetzen [Holv95; HoVe95]. Diese Objekte – auch Agenten genannt – sind durch ein eigenständiges Verhalten gekennzeichnet, das mit Hilfe von Petri-Netzen modelliert wird. Damit soll die Beschreibung großer dynamischer Systeme ermöglicht werden, die sich nicht mit globalen Steuerungsmechanismen beherrschen lassen.

PN-TOX zeichnet sich insbesondere durch die Möglichkeit aus, unterschiedlichen Netztype für die Modellierung der einzelnen Agenten zu verwenden. Jede Systemkomponente kann so mit dem Netztyp beschrieben werden, der sich am besten für das jeweilige Problemfeld eignet. Dies führt zu einer problemadäquaten und flexiblen Modellierungstechnik.

Modellierungskomponenten

Die Modellierung in PN-TOX setzt zunächst die Identifikation autonomer Objekte der Anwendungsdomäne voraus. Dafür können beliebige objektorientierte Analysemethoden eingesetzt werden. Die aktiven Objekte des Systems werden anschließend jeweils um ein *Petri-Netz-Objekt* erweitert, in dem das interne Objektverhalten und die Schnittstelle zur Außenwelt beschrieben werden. Danach erfolgt die Definition von *Koordinationsmechanismen* zwischen einzelnen Objekten mit Hilfe spezieller Kompositions- und Koordinationsobjekte. Zusätzlich können unterschiedliche Formen von Vererbungsbeziehungen zwischen Objekten in das Modell aufgenommen werden, auf die im folgenden jedoch nicht näher eingegangen wird[4].

[4] Der interessierte Leser wird in diesem Zusammenhang auf [HoVe95] verwiesen.

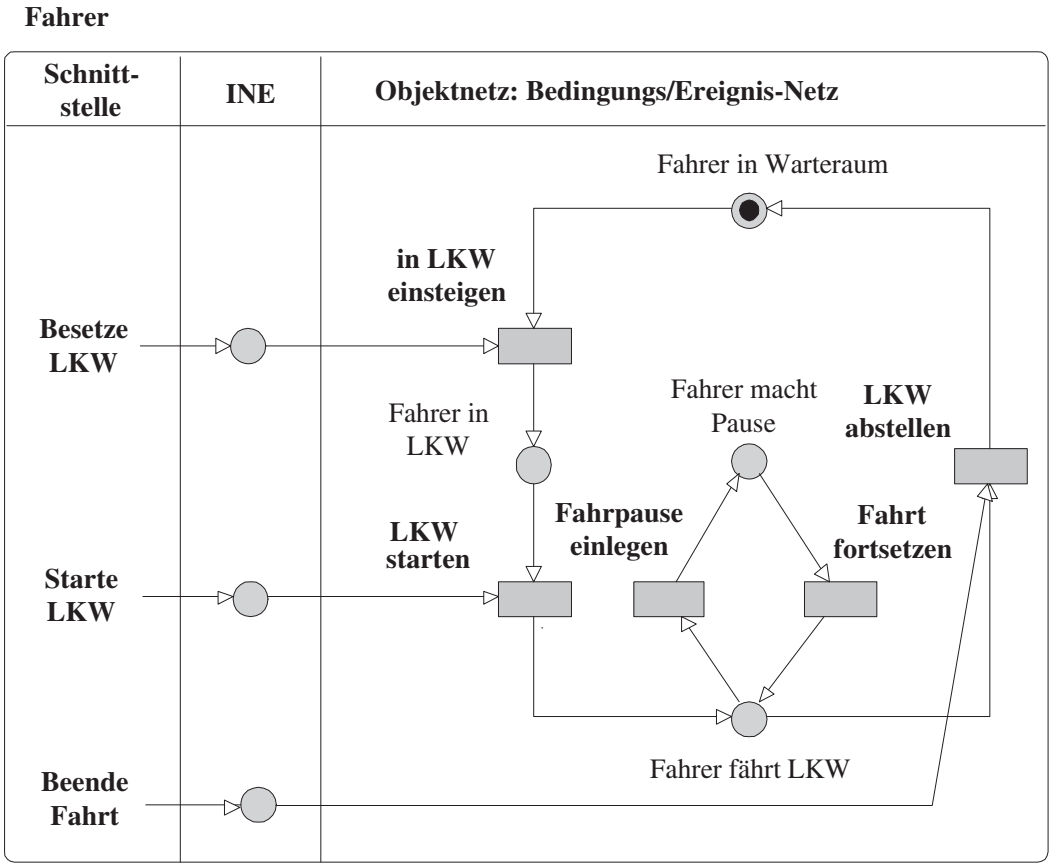

Bild 8 Ein Petri-Netz-Objekt in PN-TOX

Als Beispiel eines *Petri-Netz-Objektes* ist in Bild 8 ein Objekt „Fahrer“ für die Abwicklung eines Transportes abgebildet, das aus folgenden Komponenten besteht:

1. Ein *Objektnetz* definiert das interne Verhalten eines Fahrers. Im vorliegenden Fall wird ein einfaches Bedingungs/Ereignis-Netz verwendet, um den Lebenszyklus eines Fahrers festzulegen. Die Startmarkierung auf der Stelle „Fahrer im Warteraum“ gibt an, daß sich ein Fahrer zunächst im Warteraum befindet, wenn er neu in das System eingeführt wird. Es ist zu beachten, daß ein Objektnetz immer in einem Petri-Netz-Objekt eingekapselt ist und nicht direkt von außen manipuliert werden kann.
2. Um den Empfang von Nachrichten anderer Objekte zu ermöglichen, wird eine öffentliche *Schnittstelle* zur Verfügung gestellt. Diese Schnittstelle besteht aus Methoden, die keinen direkten Zugriff auf das Objektnetz besitzen und nicht mit Transitionen zu verwechseln sind. Ein Fahrer kann z.B. die Aufforderungen „Besetze LKW“, „Starte LKW“ oder „Beende Fahrt“ von anderen Objekten erhalten.
3. Die Schnittstellenmethoden werden über eine Erweiterung des Objektnetzes, der sogenannten „**i**nterface **n**et **e**xtension“ (INE), mit diesem verknüpft. Die INE besteht meist aus einigen Stellen, die direkt mit Transitionen des Objektnetzes verbunden sind. Wenn ein Fahrer beispielsweise die Aufforderung „Besetze LKW“ erhält, wird von dieser Methode eine Marke in die entsprechende Stelle der INE eingefügt. Daraufhin kann die Transition „in LKW einsteigen“ schalten und der Fahrer in den Zustand „Fahrer in LKW“ versetzt werden. In den Methoden wird sichergestellt, daß die Marken korrekt in das Netz eingefügt werden, und somit unterschiedliche Netztypen miteinander verknüpft werden können. In der Realisierungsphase müssen die Implementierungen dieser Methoden und deren Aufrufe in Form von Codesegmenten einer speziellen Programmiersprache spezifiziert werden.

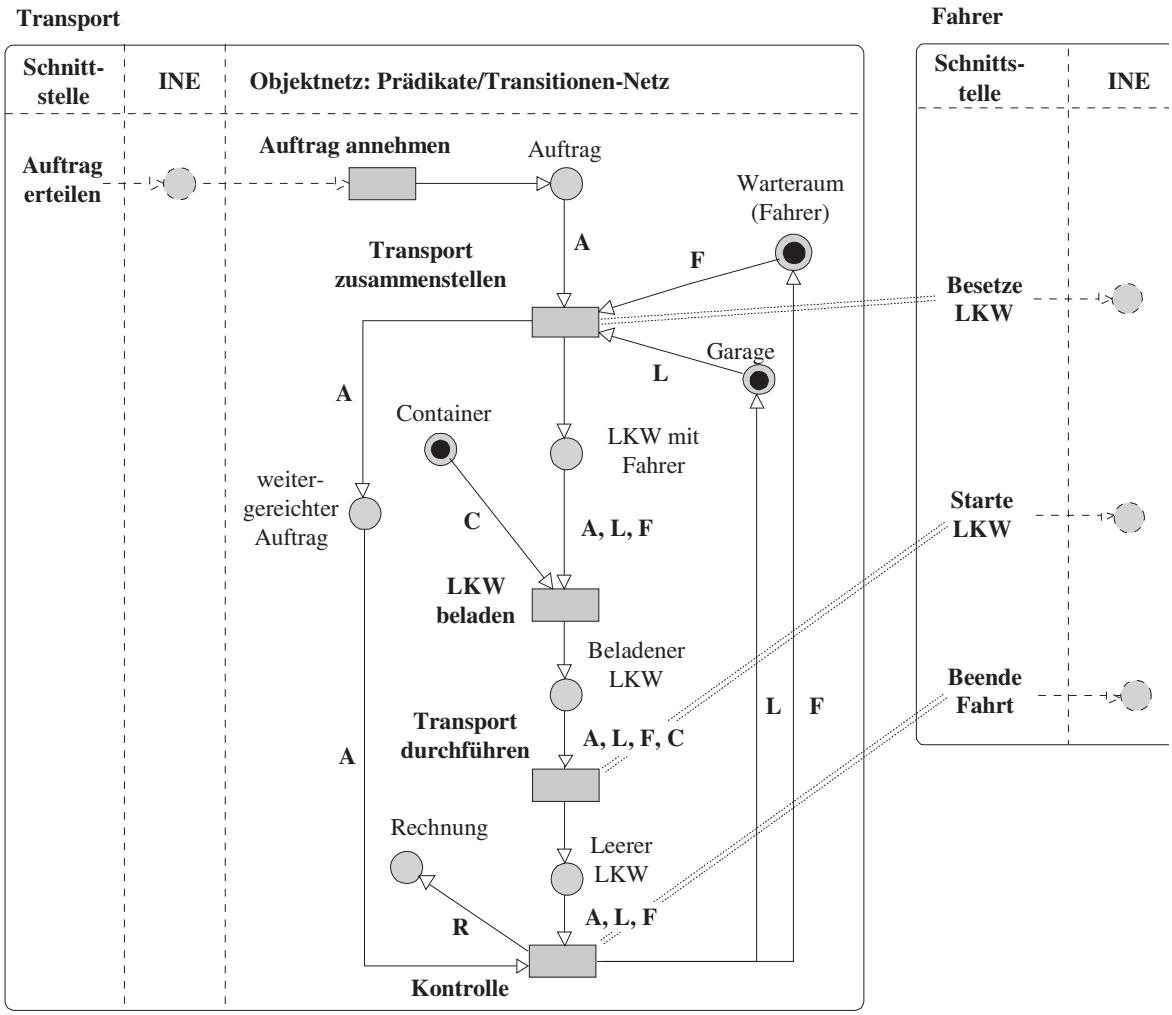

Bild 9 Ein Koordinationsobjekt in PN-TOX[5]

Die *Koordination* zwischen Objekten kann durch zwei Konstrukte beschrieben werden:

- Für eine einfache Kooperation zwischen Objekten können deren Objektnetze mit Hilfe von *Kompositionsobjekten* miteinander verknüpft werden. Der Informationsaustausch zwischen den Objekten erfolgt dabei im Prinzip durch die Synchronisation von Transitionen. Dieses Vorgehen hat allerdings die Nachteile, daß (a) die Kapselung der Objektnetze aufgebrochen werden muß und (b) keine unterschiedlichen Netztypen zur Modellierung der kooperierenden Objekte verwendet werden können.
- Demgegenüber ermöglichen *Koordinationsobjekte* die Spezifikation komplexer Kooperationsszenarien zwischen vielen Objekten unter Vermeidung der oben genannten Nachteile. Als Beispiel für ein Kooperationsobjekt ist in Bild 9 die bereits bekannte Transportabwicklung als Petri-Netz-Objekt dargestellt. Der Ablauf wurde mit einer speziellen Form von Prädikate/Transitionen-Netzen modelliert, in dem Petri-Netz-Objekte als Prädikate zugelassen werden. So können beispielsweise im Warteraum Objekte des Typs „Fahrer“ abgelegt und in der Transition „Transport zusammenstellen“ auf die Schnittstellenmethode „Besetze LKW“ des Fahrers zugegriffen werden. Der Zugriff auf Methoden wird über die

[5] Aus Gründen der Übersichtlichkeit wurde das Modell auf die Objekte „Transport“ und „Fahrer“ beschränkt. Die Kooperation zwischen einem „Transport“ und diversen „Aufträgen“, „LKW“ und „Containern“ erfolgt analog.

Beschriftung von Transitionen realisiert, die in der Graphik durch doppelt gestrichelte Linien angedeutet sind.

Softwareentwicklungsphase und Vorgehensmodell

PN-TOX sollte im Anschluß an eine objektorientierte Systemanalyse in der Entwurfs- und Realisierungsphase eingesetzt werden. Durch die Fokusierung auf eigenständige Systemobjekte und die Einsatzmöglichkeit unterschiedlicher Netztypen ist der Ansatz insbesondere für die komponentenbasierte Softwareentwicklung interessant.

Über die exakte Vorgehensweise bei der Modellierung werden keine direkten Aussagen in den angegebenen Quellen gemacht, allerdings kann davon ausgegangen werden, daß eine objektorientierte Vorgehensweise sinnvoll ist.

3.4.2 Übersicht über bestehende Ansätze

Die Ansätze zur beidseitigen Integration von Objekten und Petri-Netzen können als Weiterentwicklung der Einbettung von Netzen in Objekte verstanden werden. Auch hier werden zunächst Objekte verwendet, um das System zu strukturieren, und anschließend wird das Verhalten der Objekte mit Hilfe von Netzen beschrieben. Allerdings können die Objektnetze nicht nur anonyme Marken, sondern auch Referenzen auf andere Objekte enthalten, deren Verhalten wiederum durch Netze beschrieben wird u.s.w. (vgl. Bild 10). Diese Verschachtelung ermöglicht eine mehrstufige Modellierung von nebenläufigem Verhalten.

Aus Tabelle 3 ist ersichtlich, daß die meisten Ansätze dieser Kategorie eine eigene objektorientierte Sprache zur Definition der Systemstruktur verwenden. Graphische Elemente werden für die Modellierung der Systemstatik nur rudimentär eingesetzt. Die Definition von Abläufen erfolgt mit selbst entwickelten Petri-Netzen; ausschließlich PN-TOX ermöglicht die Modellierung von Objektnetzen mit Hilfe gängiger Netztypen. Bei diesem Ansatz ist besonders hervorzuheben, daß das Verhalten verschiedener Objekte eines Systems unter Einsatz unterschiedlicher Netztypen modelliert werden kann.

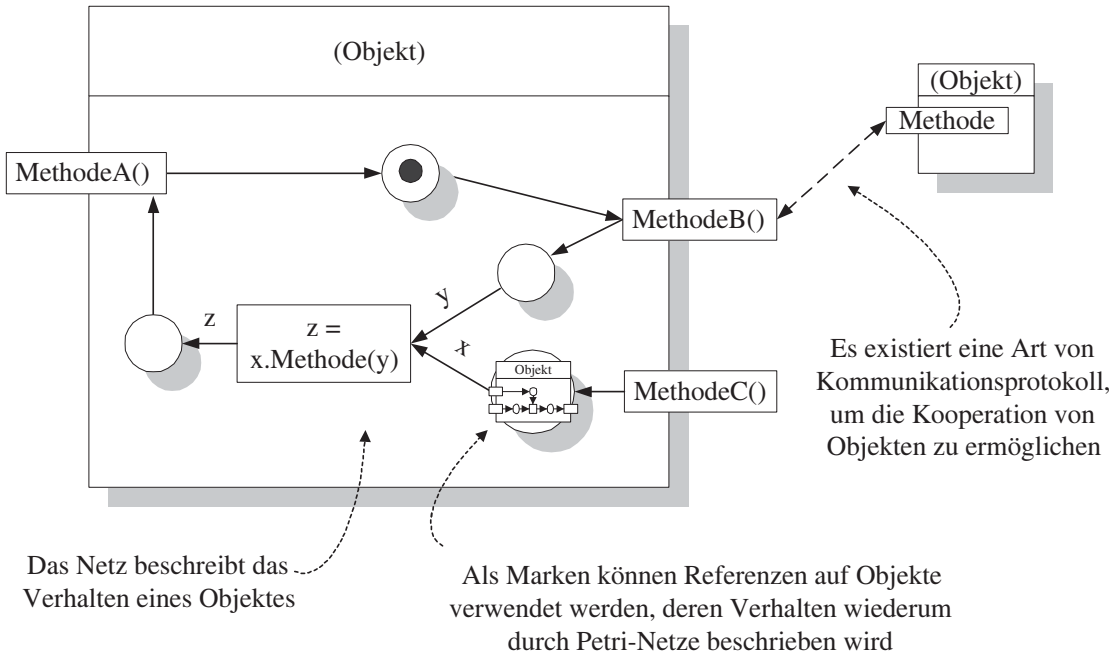

Bild 10 Beidseitige Integration von Objekten und Petri-Netzen

Ansatz	Objekt-orientierte Notation	Netztyp	Entwicklungsphase	Vorgehensmodell	Werkzeug
CO-OPN/2 Concurrent Object-Oriented Petri Nets [BiBu94; BiBG96]	eigene Sprache	eigener Netztyp	Entwurf, Realisierung	objekt-orientiert	SANDS [SAND98]
COOs Cooperative Objects [BaPa93; Sibe93; Sibe94]	eigene Sprache	eigener Netztyp	Entwurf, Realisierung	objekt-orientiert	SYROCO [SiHT95]
HOON Higher-Order Object Nets [LöWH95]	basiert auf OMT und ER-Notation	eigener Netztyp	Analyse, Entwurf	objekt-orientiert	n.v.
LOOPN++ Language for Object Petri Nets [Lako94; LaKe94a; LaKe94b; Lako97]	eigene Sprache	Transformation in gefärbte Netze möglich	Entwurf	objekt-orientiert	LOOPN++ [Lako98]
OOPN Object-Oriented Petri Nets [CeJa96; CeJa97]	eigene Notation, Smalltalk [GoRo83]	eigener Netztyp	Analyse, Entwurf	objekt-orientiert	PNtalk [Vojn98]
OOPNL Object-Oriented Petri Net Language [Esse97]	eigene Notation	eigener Netztyp	Entwurf	n.v.	Codesign [Code98]
OOPr/T Nets Object-Oriented Pr/T-Nets [Phil97]	OMT [RBPE91]	Pr/T-Netze	Entwurf, Realisierung	objekt-orientiert	n.v.
OPN Object Petri Nets [Valk95; Valk96]	n.v.	eigener Netztyp	Entwurf	objekt-orientiert	n.v.
PN-TOX Petri Net Tools for Object Concurrency Specification [Holv95; HoVe95; HoVe96a; HoVe96b]	eigene Notation	Verwendung von unterschiedlichen Netztypen möglich	Entwurf, Realisierung	objekt-orientiert	PN-TOX [Holv98]

Tabelle 3 Ansätze zur beidseitigen Integration von Objekten und Petri-Netzen

Bei den aufgeführten Arbeiten handelt es sich häufig um neuere Forschungsergebnisse, deshalb fehlt im Unterschied zu den vorangegangenen Abschnitten meist der Bezug zu bekannten höheren Netztypen und damit auch die entsprechenden Analysemöglichkeiten.

Das Haupteinsatzgebiet der genannten Ansätze ist der Entwurf von Anwendungssystemen und das Generieren von Prototypen. Dabei wird implizit ein objektorientiertes Vorgehensmodell vorausgesetzt.

4 Anwendungsgebiete

Es bleibt noch die Frage offen, für welche Anwendungsgebiete die vorgestellten Ansätze konzipiert wurden. Dies festzustellen ist kein leichtes Unterfangen, zumal in den angegebenen Quellen der Anwendungsbezug auf sehr unterschiedliche Weise hergestellt wird. Während einige Autoren zur Veranschaulichung ihrer Arbeiten konkrete Anwendungsbeispiele verwenden (z.B. Macronet, MOBY, OBM, OPM, SimCon), versuchen sich andere an Übungsbeispielen, die bestimmte Problemeigenschaften aufweisen. So werden etwa mit Hilfe des „dining philosopher

Anwendungs-richtungen	**Anwendungsgebiete**	**Ansätze**
Technische Informatik	Modellierung & Simulation verteilter und nebenläufiger Systeme, Simulation diskreter Systeme	CLOWN, CO-OPN/2, COOs, LOOPN, LOOPN++, OBJSA, OOCPN, OOPN, PAM, PROTOB, PN-TOX, THORN
	Modellierung von Netzwerkprotokollen, objektorientierten Betriebssystemen, Echtzeitsystemen, eingebetteten Systemen	LOOPN, LOOPN++, OOBM, OOPNL, PROTOB, THORN
Softwaretechnik	Modellierung von Benutzeroberflächen	COOs
	Entwicklung von Informationssystemen, Design von Datenbankanwendungen	HOOD Nets, MOBY, NetCASE, OBM, OCPN, OOPN, OOPr/T-Nets, OPM, SimCon
	Prototyping objektorientierter Designs	LOOPN, LOOPN++
Wirtschafts-informatik	Unternehmensmodellierung, Geschäftsprozeßanalyse und -modellierung	Macronet, OPM
	Büroinformationssysteme, Workflowsysteme	HOON, Macronet, MOBY
	Logistiksteuerung	SimCon
	Fertigungssteuerung, flexible Fertigungssysteme, Automatisierungstechnik	PROTOB, OPM, THORN

Tabelle 4 Anwendungsgebiete objektorientierter Petri-Netz-Ansätze

problems“ (z.B. COOs, LOOPN++, PN-TOX) die aus der Nebenläufigkeit von Prozessen resultierenden Konflikte und mögliche Synchronisationsmechanismen modelliert. Hinweise auf konkrete Anwendungsgebiete erfolgen z.T. vage, was zwangsläufig zu einer subjektiven und groben Einordnung führt. Trotzdem wird mit Tabelle 4 der Versuch unternommen, einen Überblick über Anwendungsrichtungen und -gebiete zu vermitteln.

Die Übersicht klassifiziert die Anwendungen der Methoden in drei Richtungen. Die von ihrem Gegenstandsbereich eher technikorientierten Anwendungen werden der *Technischen Informatik* zugerechnet [Krüc90, 298]. Hinweise auf konkrete betriebswirtschaftliche Anwendungen führen zu einer Einordnung in die *Wirtschaftsinformatik*. Die *Softwaretechnik* kann als Schnittmenge der beiden vorgenannten Richtungen aufgefaßt werden. Sie beschäftigt sich mit der Bereitstellung und systematischen Verwendung von Methoden und Werkzeugen für die Herstellung von Software und ist sowohl für die Technische Informatik als auch für die Angewandte Informatik (hier: Wirtschaftsinformatik) von Bedeutung [Balz96, 35]. Dabei wird deutlich, daß die gewählten Kategorien nicht ausschließlich sind. Einige Methoden kommen in mehreren Anwendungsrichtungen bzw. -gebieten gleichzeitig vor. Zudem ist nicht auszuschließen, daß einzelne Methoden ein breiteres Anwendungsspektrum besitzen können als ursprünglich von den Autoren intendiert. Daher besitzt die Übersicht einen vorläufigen Charakter.

5 Fazit

Der vorliegende Beitrag zeigt, daß sich in den letzten Jahren eine Vielfalt von methodischen Ansätzen zur Integration von objektorientierten Konzepten und Petri-Netzen entwickelt hat. Diese lassen sich je nach Integrationsrichtung in drei unterschiedliche Kategorien unterteilen. Dabei können Objekte in Petri-Netze eingebettet, Petri-Netze in Objekte gekapselt sowie Objektnetze spezifiziert werden, die als Marken wiederum Referenzen auf andere Objekte enthalten.

Die unterschiedlichen Richtungen der Anwendungsbeispiele werfen die Frage auf, wie relevant eine fundierte Beschreibung dynamischer Systemeigenschaften für betriebliche Problemstellungen ist bzw. zukünftig sein wird. Bisher spielten diese Überlegungen eine „eher untergeordnete Rolle“, da für betriebliche IS die Ausführung bestimmter Operationen nur in Ausnahmefällen zeitkritisch war oder Ungenauigkeiten in der Ablaufsteuerung kaum zu extrem negativen Auswirkungen geführt haben [AGWW98, 255]. Ob diese Sichtweise in einer Zeit aufrechterhalten werden kann, in der zunehmend primäre Wertschöpfungsaktivitäten von IS durchdrungen werden, ist allerdings fraglich. Komplexe Produktions- oder Logistiksysteme mit integrierten Just-in-Time- oder Supply-Chain-Management-Funktionen lassen erahnen, daß die Bedeutung der Steuerung und Koordination betrieblicher Prozesse zunimmt.

Trotz der bisherigen Integrationsmöglichkeiten von Objekten und Petri-Netzen und der dabei erreichten Fortschritte bleiben noch grundsätzliche Fragen offen. Werden Petri-Netze in Objekte eingebettet, so erhält man *ein einziges,* aus vielen Klassen bestehendes Modell zur Beschreibung der statischen Eigenschaften des gesamten Systems. Dadurch wird jedoch keine ausreichende Komplexitätsreduktion für die Darstellung großer Systeme erreicht. Es wäre demnach zu untersuchen, wie sich Klassen zu größeren Einheiten in Form von statischen Systemkomponenten zusammenfassen lassen. Andererseits bietet die Einbettung von Objekten in Petri-Netzen zwar prinzipiell die Möglichkeit der hierarchischen Strukturierung von dynamischen Systemeigenschaften durch die Verfeinerung von Netzstrukturen, jedoch existieren keine Methoden, wie Ablaufnetze systematisch erstellt und in einzelne Prozeßkomponenten verfeinert werden können. Zudem wird das Systemverhalten auch hier durch *ein einziges* Modell beschrieben, das sich nicht ohne weiteres in austauschbare und wiederverwendbare

Komponenten zerlegen läßt. Die Modellierung großer Systeme bleibt damit aufwendig und die erstellten Modelle werden schnell unübersichtlich.

Verfolgt man den Gedanken zur Bildung von Systemkomponenten weiter, so kommt der beidseitigen Integration von Objekten und Petri-Netzen eine besondere Bedeutung zu. Hier erfolgt in einigen Ansätzen, wie am konkreten Beispiel aufgezeigt wurde, die Modellierung des internen Objektverhaltens durch Petri-Netz-Objekte und die Interaktion zwischen Objekten durch Koordinationsobjekte, die ebenfalls durch Petri-Netze beschrieben werden. Dadurch können elementare Objekte und ihre internen Verhaltenseigenschaften prinzipiell zu umfassenderen Komponenten aggregiert werden, wobei die Wechselbeziehungen zwischen den Objekten ebenfalls in die Beschreibung eingehen.

Die Einführung einer geeigneten Komponentensicht besitzt somit eine große Bedeutung im Hinblick auf die umfassende und übersichtliche Modellierung von IS und stellt ein näher zu untersuchendes Anwendungspotential für bestimmte Ansätze zur beidseitigen Integration von Objekten und Petri-Netzen dar. Durch die Einführung eines geeigneten Komponentenbegriffs in die Welt objektorientierter Petri-Netze kann ein wichtiger Beitrag zur Modellierung komplexer Systeme geleistet werden. Zudem stünde damit eine mächtige Modellierungstechnik für die komponentenbasierte Softwareentwicklung und Modellierung von Geschäftsobjekten (business objects) zur Verfügung.

Literatur

[AaWa91] *van der Aalst, W.M.P.; Waltmans, A.W.:* Modelling logistic systems with ExSpect. In: *Sol, H.G.; van Hee, K.M. (Hrsg.):* Dynamic Modelling of Information Systems. Elsevier Science Publishers B.V., North-Holland 1991, S. 269-287.

[AGWW98] Alpar, P.; Grob, H.L.; Weimann, P.; Winter, R.: Unternehmensorientierte Wirtschaftsinformatik: Eine Einführung in die Strategie und Realisierung erfolgreicher IuK-Systeme, Braunschweig u.a. 1998.

[ArGo98] *Arnold, K.; Gosling, J.:* The Java Programming Language, Second Edition. Addison-Wesley, Reading 1998.

[BaBr88] *Baldassari, M.; Bruno, G.:* An environment for object-oriented conceptual programming based on PROT Nets. In: *Rozenberg (Hrsg.):* Advances in Petri Nets, Zaragoza/Spain 1988. LNCS 340. Springer, Berlin 1988, S. 1-19.

[BaBr91] *Baldassari, M.; Bruno, G.:* PROTOB: An object oriented methodology for developing discrete event dynamic systems. In: Computer Languages 16 (1991) 1, S. 39-63.

[BaCD95] *Battiston, E.; Chizzoni, A.; De Cindio, F.:* Inheritance and Concurrency in CLOWN. In: *Agha, G.; de Cindio, F. (Hrsg.):* Proceedings of the 1st Workshop on Object-Oriented Programming and Models of Concurrency. Turin/Italy 1995.

[BaCo95] *Bachatène, H.; Coriat, M.:* PAM: A Petri net-based Abstract Machine for the Specification of Concurrent Systems. In: *Agha, G.; de Cindio, F. (Hrsg.):* Proceedings of the 1st Workshop on Object-Oriented Programming and Models of Concurrency. Turin/Italy 1995.

[BaDM88] *Battiston, E.; De Cindio, F.; Mauri, G.:* OBJSA Nets: A class of high-level nets having objects as domains. In: *Rozenberg (Hrsg.):* Advances in Petri Nets 1988. LNCS 340. Springer, Berlin 1988, S. 20-43.

[Bakk98] *Bakkenist Managment Consultants:* ExSpect: Executable Specification Tool. http://www.bakkenist.nl/it/itlogist.htm#exspect, 1998.

[Balz96] *Balzert, H.:* Lehrbuch der Software-Technik: Software-Entwicklung, Berlin u.a. 1996.

[BaPa93] *Bastide, R.; Palanque, P.:* Cooperative objects: a concurrent Petri Net based object-oriented language. In: Proceedings of the IEEE International Conference on Systems, Man and Cybernetics, Le Touquet/France 1993, vol. 3. IEEE, New York 1993, S. 286-291.

[Barn84] *Barnes, J.G.:* Programming in ADA. Addison-Wesley, London u.a. 1984.

[Bast95] *Bastide, R.:* Approaches in unifying Petri Nets and the Object-Oriented Approach. In: *Agha, G.; de Cindio, F. (Hrsg.):* Proceedings of the 1st Workshop on Object-Oriented Programming and Models of Concurrency. Turin/Italy 1995.

[Baum90] *Baumgarten, B.:* Petri-Netze: Grundlagen und Anwendungen, Mannheim u.a. 1990.

[BeMo93] *Becker, U.; Moldt, D.:* Object-oriented concepts for coloured Petri Nets. In: Proceedings of the IEEE International Conference on Systems, Man and Cybernetics, Le Touquet/France 1993, vol. 3. IEEE, New York 1993, S. 279-285.

[BiBG96] *Biberstein, O.; Buchs, D.; Guelfi, N.:* COOPN/2 : A Specification Language for Distributed Systems Engineering. Technical Report 96/167, Swiss Federal Institute of Technology, Lausanne 1996.

[BiBu94] *Biberstein, O.; Buchs, D.:* An Object Oriented Specification Language based on Hierarchical Petri Nets. Technical Report 94/76, Swiss Federal Institute of Technology, Lausanne 1994.

[Burk94] *Burkhardt, R.:* Modellierung dynamischer Aspekte mit dem Objekt-Prozeß-Modell. Dissertation, Fakultät für Informatik und Automatisierung der Technischen Universität Ilmenau, Ilmenau 1994.

[Burk97] *Burkhardt, R.:* UML - Unified Modeling Language. Addison-Wesley, Bonn u.a. 1997.

[Burk98] *Burkhardt, R.:* Objekttechnologie-Werkbank. http://www.theoinf.tu-ilmenau.de/proinf/otw/otw.html, Ilmenau 1998.

[CeJa96] *Ceska, M.; Janousek, V.:* Object Orientation in Petri Nets. In: Proceedings of the 22nd ASU Conference on Object Oriented Modelling and Simulation. Clermont-Ferrand 1996, S. 69-80.

[CeJa97] *Ceska, M.; Janousek, V.:* A Formal Model for Object Oriented Petri Nets Modeling. In: Advances in Systems Science and Applications (1997) Special Issue.

[Chen76] *Chen, P.:* The Entity Relationship Model - Towards a Unified View of Data. In: ACM Transactions on Database Systems 1 (1976) 1, S. 9-36.

[Code98] CodeSign. http://www.tik.ee.ethz.ch/~codesign/, Zürich 1998.

[DeMa78] *DeMarco, T.:* Structured Analysis and System Specification, Englewood Cliffs 1978.

[DiGi91] *Di Giovanni, R.:* HOOD Nets. In: *Rozenberg (Hrsg.):* Advances in Petri Nets, Paris/France 1991. LNCS 524. Springer, Berlin 1991, S. 140-160.

[DLMR94] *Dahr, M.; Lautenbach, K.; Marx, T.; Ridder, H.:* NET CASE: Towards a Petri Net Based Technique for the Development of Expert/Database Systems. http://www.uni-koblenz.de/universitaet/fb4/publications/GelbeReihe/RR-2-94.ps.gz, Koblenz-Landau 1994.

[Engl93] *English, S.L.:* Coloured Petri Nets for object-oriented modelling. Dissertation, University of Brighton, Brighton 1993.

[Esse97] *Esser, R.:* An object oriented Petri net language for embedded system design. In: Proceedings of the 8th International Workshop on Software Technology and Engineering

Practice incorporating Computer Aided Software Engineering, London/UK 1997. IEEE, Los Alamitos 1997, S 216-223.

[Fers97] *Ferstl, O.K.:* Objektorientierte Entwurfsmethode. In: *Mertens, P. (Haupt-Hrsg.):* Lexikon der Wirtschaftsinformatik, 3. Auflage, Berlin u.a. 1997, S. 288-289.

[FeSi98] *Ferstl, O.K.; Sinz, E.J.:* Grundlagen der Wirtschaftsinformatik, Band 1, 3. Auflage, München u.a. 1998.

[FGJM85] *Futatsugi, K.; Goguen, J.A.; Jouannaud, J.P.; Meseguer, J.:* Principles of OBJ2. In: Proceedings of the 12th Symposium on Principles of Programming Languages. ACM, 1985, S. 52-66.

[FlLi93] *Fleischhack, H.; Lichtblau, U.*: MOBY - A Tool for High Level Petri Nets with Objects. In: Proceedings of the IEEE International Conference on Systems, Man and Cybernetics, Le Touquet/France 1993, vol. 4. IEEE, New York 1993, S. 644-649.

[GeLa81] *Genrich, H.; Lautenbach, K.:* System Modelling with High Level Petri Nets. In: Theoretical Computer Science (1981) 13, S. 109-136.

[Genr86] *Genrich, H.J.:* Predicate/Transition Nets. In: Advances in Petri Nets 1986. LNCS 254. Springer, Berlin 1987, S. 207-247.

[GoRo83] *Goldberg, A.; Robson, D.:* Smalltalk-80: The Language and its Implementation. Addison-Wesley, Reading 1983.

[HaDi97] *Hanish, A.A.; Dillon, T.S.:* Object-oriented behaviour modelling for real-time design. In: Proceedings of the 3rd International Workshop on Object-Oriented Real-Time Dependable Systems, Newport Beach/USA 1997. IEEE, Los Alamitos/USA 1997, S. 74-82.

[HaMc81] *Hammer, M.; McLeod, D.:* Database Description with SDM: A Semantic Data Model. In: ACM Transaction on Database Systems 6 (1981) 3, S. 351-386.

[Hare87] *Harel, D.*: Statecharts: A Visual Formalism for Complex Systems. In: Science of Computer Programming 8 (1987), S. 231-274.

[HeRV93] *Hee, K.M..v.; Rambags, P.M..P.; Verkoulen, P.A..C.:* Specification and simulation with ExSpect. In: *Lauer, Peter E. (Hrsg.):* Functional programming, concurrency, simulation and automated reasoning. International lecture series 1991-1992. Springer, Berlin u.a. 1993, S. 296-328.

[HeSV88] *Hee, K.M..v.; Somers, L.J.; Voorhoeve, M.:* ExSpect, the Functional Part. Computing Science Note 88/20, Eindhoven University of Technology 1988.

[HeSV89] *Hee, K.M..v.; Somers, L.J.; Voorhoeve, M.:* Executable Specifications for Distributed Information Systems. In: *Falkenberg, E.D.; Lindgreen, P. (Hrsg.):* Information System Concepts: An In-depth Analysis. IFIP, North-Holland 1989.

[HeSV91a] *Hee, K.M..v.; Somers, L.J.; Voorhoeve, M.:* A formal framework for dynamic modelling of information systems. In: *Sol, H.G.; van Hee, K.M. (Hrsg.):* Dynamic Modelling of Information Systems. Elsevier Science Publishers B.V., Noth-Holland 1991, S. 227-236.

[HeSV91b] *Hee, K.M..v.; Somers, L.J.; Voorhoeve, M.:* The ExSpect tool. In: *Prehn, S.; Toetenel, H. (Hrsg.):* Formal Software Development Methods. LNCS 551. Springer, Berlin 1991, S. 683-684.

[HeSW75] *Held, G.; Stonebraker, M.R.; Wang, E.:* INGRES: A Relational Data Base System. In: Proceedings of the NCC, vol. 44. AFIPS Press, 1975, S. 409-416.

[Holv95] *Holvoet, T.:* Agents and Petri Nets. In: The Petri Net Newsletter (1995) 49, S. 3-8.

[Holv98] *Holvoet, T.:* The PN-TOX Home Page. http://www.cs.kuleuven.ac.be/~tom/ PNTOX/, Leuven 1998.

[HoVe95] *Holvoet, T.; Verbaeten, P.:* PN-TOX: a Paradigm and Development Environment for Object Concurrency Specifications. In: *Agha, G.; de Cindio, F. (Hrsg.):* Proceedings of the 1st Workshop on Object-Oriented Programming and Models of Concurrency. Turin/Italy 1995.

[HoVe96a] *Holvoet, T.; Verbaeten, P.:* Synchronization Specifications for Agents with Net-based Behaviour Description. In: Proceedings of CESA IMACS Conference, Symposium on Discrete Events and Manifacturing Systems. Lille 1996, S. 613-618.

[HoVe96b] *Holvoet, T.; Verbaeten, P.:* Using Petri Nets for Specifying Active Objects and Generative Communisation. In: *Agha, Gul; Yonezawa, Akinori; De Cindio, Fiorella (Hrsg.):* Proceedings of the Workshop on Object-Oriented Programming and Models of Concurrency within the 17th International Conference on Application and Theory of Petri Nets. Osaka 1996.

[Jens96] *Jensen, K.:* Coloured Petri Nets. Springer, Berlin u.a. 1996.

[KaSc91] *Kappel, G.; Schrefl, M.:* Using An Object-Oriented Diagram Techinque For The Design Of Information Systems. In: *Sol, H.G.; van Hee, K.M. (Hrsg.):* Dynamic Modelling of Information Systems. Elsevier Science Publishers B.V., North-Holland 1991, S. 121-164.

[KeSB94] *Keller, R.K.; Shen, X.; Bochmann, G.v.:* Macronet - A Simple, yet Expressive and Flexible Formalism for Business Modelling. In: Proceedings of the Workshop on Computer-Supported Cooperative Work, Petri Nets and Related Formalisms within the 15th International Conference on Application and Theory of Petri Nets. Zaragoza/Spain 1994, S. 51-55.

[Krüc90] *Krückeberg, F.:* Informatik. In: *Spaniol, O.; Krückeberg, F. (Hrsg.):* Lexikon Informatik und Kommunikationstechnik, Düsseldorf 1990, S. 297-299.

[LaKe91] *Lakos, C.A.; Keen, C.D.:* Simulation with Object-Oriented Petri Nets. In: Proceedings of the Australian Software Engineering Conference. Sydney 1991.

[LaKe94a] *Lakos, C.A.; Keen, C.D.:* Information Systems Modelling Using LOOPN++, an Object Petri Net Scheme. In: Proceedings of the International Working Conference on Dynamic Modelling and Information Systems. 1994.

[LaKe94b] *Lakos, C.A.; Keen, C.D.:* LOOPN++: A New Language for Object-Oriented Petri Nets. Technical Report TR94-4, Computer Science Department, University of Tasmania 1994.

[Lako94] *Lakos, C.A.:* Object Petri Nets - Definition and Relationship to Coloured Nets. Technical Report TR94-3, Computer Science Department, University of Tasmania 1994.

[Lako97] *Lakos, C.A.:* Object Oriented Modelling with Object Petri Nets. In: Advances in Petri Nets. LNCS. Springer, Berlin 1997.

[Lako98] *Lakos, C.:* Object Petri Nets. http://www.eecs.utas.edu.au/Deptinfo/research/groups/csrg/nwsig/opn.html, Hobart/Tasmania 1998.

[LöWH95] *Löwe, M.; Wikarski, D.; Han, Y.:* Higher-Order Object Nets and their Application to Workflow Modeling. Forschungsbericht 95-34, FB Informatik, Technische Universität Berlin 1995.

[Marx95] *Marx, T.:* NetCASE - a Petri Net based Method for Database Application Design and Generation. Fachbericht Informatik 11-95, Universität Koblenz-Landau 1995.

[Marx97] *Marx, T.:* APRIL- Visualisierung der Anforderungen. Fachbericht Informatik 7-97, Universität Koblenz-Landau 1997.

[Marx98] *Marx, T.:* Das Werkzeug NEPTUN. http://www.uni-koblenz.de/~agpn/neptun/neptun-tool.html, Koblenz-Landau 1998.

[Meye92] *Meyer, B.:* Eiffel: The Language. In: Prentice Hall, 1992.

[MoMa97] *Moldt, D.; Maier, C.:* Object Coloured Petri Nets - a Formal Technique for Object Oriented Modelling . In: *Farwer, B.; Moldt, D.; Stehr, M.-O. (Hrsg.):* Petri Nets in System Engineering, Modelling, Verification and Validation, Bericht FBI-HH-B-205/97. Fachbereich Informatik, Universität Hamburg, 1997, S. 11-19.

[NüZi97] *Nüttgens, M.; Zimmermann, V.*: Objektorientierte Ereignisgesteuerte Prozeßkette (oEPK) - Methode und Anwendung. In: *Scheer, A.-W. (Hrsg.)*: Veröffentlichungen des Instituts für Wirtschaftsinformatik (IWi), Heft 141, Universität des Saarlandes, 1997.

[Ober96] *Oberweis, A.:* Modellierung und Ausführung von Workflows mit Petri-Netzen, Stuttgart u.a. 1996.

[Patz82] *Patzak, G.:* Systemtechnik - Planung komplexer innovativer Systeme: Grundlagen, Methoden, Techniken, Berlin u.a. 1982.

[PhBu95] *Philippow, I.; Burkhardt, R.:* Dynamische Verhaltensmodellierung mit dem Objekt-Prozeß-Modell. http://141.24.76.100/proinf/otw/psdocs/Braunsch.ps.gz, 1995.

[PhBW95] *Philippow, I.; Burkhardt, R.; Wolf, M.:* Objektorientierte Modellierung mit dem Objekt-Prozeß-Modell. EMISA-Fachgruppentreffen der GI Fachgruppe 2.5.2, 1995.

[Phil97] *Philippi, S.:* System modelling using Object-Oriented Pr/T-Nets. Fachbericht Informatik 25-97, Universität Koblenz-Landau 1997.

[Rati97] *Rational Software Corporation*: Unified Modeling Language, version 1.1. http://www.rational.com/uml/documentation.html, 1997.

[RBPE91] *Rumbaugh, J.; Blaha, M.; Premerlani, W.; Eddy, F.; Lorensen, W.:* Object-Oriented Modelling and Design. Prentice-Hall, Englewood Cliffs 1991.

[Reis85] *Reisig, W.:* Systementwurf mit Netzen. Springer, Berlin u.a. 1985.

[RoWi91] *Rosenstengel, B.; Wienand, U.:* Petri-Netze - eine anwendungsorientierte Einführung. Vieweg, Wiesbaden 1991.

[SAND98] *Concurrency and Formal Methods Group, University of Geneva and Swiss Federal Institute of Technology in Lausanne (EPFL):* Available Software: SANDS. http://lglwww.epfl.ch/Conform/soft.html, 1998.

[ScSW95] *Schöf, S.; Sonnenschein, M.; Wieting, R.*: Efficient Simulation of THOR Nets. In: Proceedings of the 16th International Conference on Application and Theory of Petri Nets, Torino/Italy 1995. LNCS 935. Springer, Berlin 1995, S. 412-431.

[Sibe93] *Sibertin-Blanc, C.:* A Client-Server Protocol for the Composition of Petri Nets. In: Application and Theory of Petri Nets. LNCS 691. Springer, Berlin 1993, S. 377-396.

[Sibe94] *Sibertin-Blanc, C.:* Cooperative nets. In: *Valette, R. (Hrsg.):* Proceedings of the 15th International Conference on Application and Theory of Petri Nets, Zaragoza/Spain 1994. LNCS 815. Springer, Berlin 1994, S. 471-490.

[SiHT95] *Sibertin-Blanc, C.; Hameurlain, N.; Touzeau, P.:* SYROCO: a C++ implementation of Cooperative Objects. In: *Agha, G.; de Cindio, F. (Hrsg.):* Proceedings of the 1st Workshop on Object-Oriented Programming and Models of Concurrency. Turin/Italy 1995.

[Sinz91] *Sinz, E.J.:* Objektorientierte Analyse. In: WIRTSCHAFTSINFORMATIK, Heft 5/1991, S. 455-457.

[Stro92] *Stroustrup, B.:* Die C++-Programmiersprache. Addison-Wesley, Bonn u.a. 1992.[Valk95] *Valk, R.:* Petri Nets as Dynamical Objects. In: Proceedings of the 1st Workshop on Object-Oriented Programming and Models of Concurrency. Turin/Italy 1995.

[Valk96] *Valk, R.:* On Processes of Object Petri Nets. Fachbericht 185/96, Fachbereich Informatik, Universität Hamburg 1996.

[Verk93] *Verkoulen, P.A..C.:* Integrated Information Systems Design. Dissertation, Technische Universität Eindhoven, Eindhoven 1993.

[Vojn98] *Vojnar, T.:* PNtalk System. http://www.fee.vutbr.cz/UIVT/homes/vojnar/PNtalk/pntalk.system.html, Brno 1998.

[Wiet98] *Wieting, R.:* DNS: Information about the Tools. http://offis.offis.uni-oldenburg.de/projekte/dns/dns4.htm, Oldenburg 1998.

[Zele95] *Zelewski, S.:* Petrinetzbasierte Modellierung komplexer Produktionssysteme, Band 9: Beurteilung des Petrinetz-Konzepts, Arbeitsbericht Nr. 14 des Instituts für Produktionswirtschaft und Industrielle Informationswirtschaft der Universität Leipzig, Leipzig 1995.

Der Beitrag gibt einen systematischen Überblick über verfügbare Ansätze zur Integration von Objekten und Petri-Netzen. Dabei geht es weniger um die Analyse und Bewertung von Integrationsansätzen im einzelnen, sondern vielmehr darum, grundlegende Integrationsrichtungen und ihre spezifischen Eigenschaften herauszuarbeiten sowie Anhaltspunkte zum daraus resultierenden Nutzenpotential für die Entwicklung betrieblicher Informationssysteme zu gewinnen.

ISSN 1864-9300