Ein auf Petri-Netzen basierendes Geschäftsprozessmodell und dessen Anwendung im E-Commerce

Dissertation

zur Erlangung der Würde eines
Doktors der Wirtschafts- und Sozialwissenschaften,
vorgelegt der Wirtschafts- und Sozialwissenschaftlichen Fakultät
der Universität Freiburg (Schweiz)

von

Daniel Frauchiger
von Wyssachen BE

genehmigt von der Fakultät der
Wirtschafts- und Sozialwissenschaften
am 29. September 2010, auf Antrag von
Herrn Prof. Dr. Andreas Meier (erster Referent) und
Frau Prof. Dr. Stephanie Teufel (zweite Referentin)

Freiburg, 2010

Mit der Annahme einer Dissertation beabsichtigt die Wirtschafts- und Sozialwissenschaftliche Fakultät der Universität Freiburg nicht, zu den darin enthaltenen wissenschaftlichen Meinungen des Verfassers Stellung zu nehmen (Fakultätsbeschluss vom 23. Januar 1990).

Vorwort

Die Geschäftsprozessmodellierung und der dahinter steckende Gedanke, die Herausforderungen im organisatorischen Geschehen aus einer prozessorientierten Sicht anzupacken, ist nicht neu und doch hochaktuell. Dies zeigt sich unter anderem daran, dass der Anteil der Beratungsprojekte mit Schwerpunkt Geschäftsprozess, denen der Verfasser in seinem beruflichen Alltag begegnet, über die Jahre prozentual stets gleich hoch geblieben ist.

Die vorliegende Arbeit hat ihren Ursprung in einer Praktikumsstelle einer international tätigen Revisionsunternehmung, wo ich mich der Frage nach der geeigneten Darstellung von Prozessen zur Erkennung von operationellen Risiken widmete, und in der daraus entstandenen Lizentiatsarbeit. In den vier Jahren als Assistent und Doktorand in der Forschungsgruppe Informationssysteme der Universität Freiburg i. Ue. hatte ich die Gelegenheit, meine Kenntnisse zum Thema Geschäftsprozesse und zu vielen weiteren Themen, darunter besonders erwähnenswert das World Wide Web, welches laufend neue Formen der Zusammenarbeit hervorbringt, zu vertiefen, bevor ich mich entschlossen habe, nach vielen Jahren wieder in die Privatwirtschaft zurückzukehren.

Ein besonderer Dank geht an meinen hochgeschätzten und grosszügigen Doktorvater, Prof. Dr. Andreas Meier, welcher stets mit Rat und Tat zur Seite stand und einen mir in tiefer Erinnerung bleibenden 2wöchigen Lehraufenthalt in Vietnam ermöglicht hat, Frau Prof. Dr. Stephanie Teufel, welche freundlicherweise das Korreferat übernommen und viele wertvolle Hinweise geliefert hat, Herrn Dr. Dac Hoa Nguyen für seine spontane, grosszügige und herzliche Unterstützung, meinen ehemaligen Arbeitgeber in Basel, dank dem ich auf ein interessantes Gebiet gestossen bin, meinen heutigen Arbeitgeber, der

mir jederzeit die für ein Dissertationsvorhaben notwendigen Freiräume ermöglicht hat, und meine ehemaligen Arbeitskollegen an der Universität, welche mich begleitet haben, darunter Dr. Stefan Hüsemann, Dr. Andreea Ionas, Dr. Peter Küng, Dr. Dona Mommsen-Ghosh, Dr. Henrik Stormer, Dr. Nicolas Werro, Dr. Thomas Wettstein und Daniel Wismer. Die überaus grosse Geduld und stete Unterstützung meiner Partnerin Christa hat wesentlich zum Entstehen dieser Arbeit beigetragen, welche dank meinen Eltern und Josef Inauen überhaupt erst möglich war, ihnen danke ich ganz besonders.

Kurzfassung

Der Gedanke der prozessorientierten Sicht reicht viele Jahrzente zurück, hat sich aber erst in jüngerer Zeit unter dem Gesichtspunkt von Kunden- und Resultatausrichtung, Qualitätssicherung, Supply Chain Management, Effizienzsteigerung und damit Wettbewerbsfähigkeit durchgesetzt. Der Wechsel von einer Aufbau- hin zu einer Prozessorganisation erfolgt initial im Entwurf und in der Umsetzung des neuen Prozessmodells. Anschliessend ist es Aufgabe der Führung, diese Prozesse laufend zu überwachen und zu steuern. Dabei stellt sich zum einen die Frage nach dem geeigneten Modell und zum anderen die nach der geeigneten Modellsprache.

Die vorliegende Arbeit unterzieht zwei in der Praxis häufig anzutreffende Prozessmodelle sowie die mathematisch begründeten, formalen Petri-Netze einer Beurteilung aus Sicht der Wirtschaftsinformatik. Danach folgt ein Vorschlag, welcher auf den Petri-Netzen beruht, aber im Gegensatz zu diesen besser an die Bedürfnisse der Geschäftsprozessmodellierung angepasst ist, was in der anschliessenden Beurteilung gezeigt wird.

Die meisten Geschäftsprozesse erfahren heutzutage eine informationstechnische Unterstützung, deren Realisierung nicht zuletzt aufgrund unterschiedlicher Modelle auf seiten Betrieb und auf seiten Informatik mit hohem Aufwand verbunden ist. Am Beispiel eines in einem Webshop ablaufenden Bestellprozesses wird gezeigt, wie mit dem vorgestellten Modellansatz sowohl die fachliche Sicht wie die technische Sicht im selben Modell abgebildet werden kann und so Potential für kürzere Entwicklungswege geschaffen werden kann.

Inhaltsverzeichnis

Abkürzungsverzeichnis

Abb.	Abbildung
Abs.	Absatz
al.	alii
APN	Authentic Petri Net
ARIS	Architektur integrierter Informationssysteme
bspw.	beispielsweise
bzgl.	bezüglich
d. h.	das heisst
DB	Datenbank
DV	Datenverarbeitung
e. V.	eingetragener Verein
EPK	Ereignisgesteuerte Prozessketten
ggü.	gegenüber
Ggs.	Gegensatz
GI	Gesellschaft für Informatik e. V.
GNU	GNU's not UNIX
GoM	Grundsätze ordnungsmässiger Modellierung
GPL	GNU General Public License
HTML	Hypertext Markup Language
HTTP	Hypertext Transfer Protocol
i. d. R.	in der Regel
i. Ue.	im Uechtland
IAUF	Institut für Automation und Unternehmensforschung
ICAM	Integrated Computer Aided Manufacturing
IDEF	ICAM Definition

IM	imaginäre Marke
insbes.	insbesondere
IT	Informationstechnologie
Kap.	Kapitel
MOF	Meta Object Facility
MVC	Model-View-Controller
OMG	Object Management Group
resp.	respektive
S.	Seite
s.	siehe
s. a.	siehe auch
s. u.	siehe unten
sog.	sogenannte
Tab.	Tabelle
TCP/IP	Transmission Control Protocol/Internet Protocol
u. a.	unter anderem
u. a. m.	und andere mehr
u. U.	unter Umständen
u. v. a.	unter vielen anderen
u. w.	und weitere
UML	Unified Modeling Language
URI	Uniform Resource Identifier
URL	Uniform Resource Locator
usw.	und so weiter
vgl.	vergleiche
WWW	World Wide Web
z. B.	zum Beispiel
z. T.	zum Teil

Abbildungsverzeichnis

Tabellenverzeichnis

Kapitel 1

Einleitung

Die Prozessbetrachtung hat sich in der betriebswirtschaftlichen Führung während der letzten Jahre etabliert. Dies zeigt sich exemplarisch im Titel des Vorworts von Malik in Stö09, der schlicht „Prozessmanagement als solides Handwerk“ lautet. Neun Jahre früher schreiben Osterloh und Frost in ihrem Vorwort, dass ihr Buch Praktikern und Studierenden zeigen will, dass „Business Reengineering keine Modewelle ist, sondern zum festen Bestandteil des Instrumentenkastens der Managerinnen und Manager werden wird“ (OF96). Dass sich ihre Prognose bestätigt hat, steht im Vorwort zur vierten Auflage desselben Werks (OF03). Ebenfalls machen sie auf das umfassendere Konzept des Prozess-Managements aufmerksam, in welchem das von Hammer und Champy geprägte Business Process Reengineering (HC93) verankert ist.

Prozess-Management zur Identifikation, Gestaltung und Operationalisierung von Geschäftsprozessen ist auf ein Prozessmodell zur Dokumentation und Kommunikation angewiesen. Welches Augenmerk diesem Prozessmodell zuzukommen hat, darüber gehen die Meinungen auseinander. Während sich Rosemann et al. (RSD08) intensiv mit der Gestaltung von Prozessmodellierung auseinandersetzen, kritisiert Stöger unter Hinweis auf RSD08 diese Reduzierung auf technische Belange (Stö09). Es mag sicher richtig sein, dass ob lauter Technik das Wesentliche vergessen gehen kann. Umgekehrt kann Wesentliches bei falscher oder falsch angewandter Technik verlorengehen. Ein Werkzeug ist letztlich ein Hilfsmittel, trotzdem soll seine Eignung für eine bestimmte Aufgabe sorgfältig geprüft werden, bevor es zum Einsatz gelangt;

immerhin haben die Aussagen des Werkzeuges normalerweise Einfluss auf die unternehmerischen Entscheidungen. Dasselbe gilt für das zugrunde gelegte Prozessmodell, woraus folgt, dass sich das Prozess-Management ebenfalls um die Frage der Modellqualität kümmern muss.

Ein qualitativ hochstehendes Modell ist auch dann hilfreich, wenn die im Rahmen des Prozess-Managements modellierten Prozesse software-technisch unterstützt werden sollen – idealerweise liegen diese möglichst formal und in der notwendigen Detaillierung vor, so dass sie sich ohne grosse Anpassungen in ein IT-System abbilden lassen[1].

1.1 Geschäftsprozess-Management

Das Geschäftsprozess-Management ist eine Führungsfunktion und hat die stete Optimierung der Prozesse zum Inhalt[2]. Unterschieden werden kann eine strategische und eine operative Komponente. Während auf der strategischen Ebene die Effektivität und damit die Auswahl der Tätigkeitsgebiete an Hand der Prozesslandschaft von Interesse ist, geht es auf der operativen Ebene darum, die einzelnen im Prozessmodell dokumentierten Prozesse effizient zu gestalten und zu unterhalten. Die Steuerung der Prozesse geschieht auf strategischer Ebene mittels aggregierter Kennzahlen, welche aus den unterschiedlichen Messungen der einzelnen Prozesse auf operativer Ebene gewonnen wurden.

Stö09 macht in seiner Definition die Faktoren *Resultatorientierung*, *Kundenorientierung*, *Beitrag ans Ganze*, *Kontrollier-, Mess- und Beurteilbarkeit*, *Wiederholbarkeit und Routine*, *Verantwortlichkeit* und *Führbarkeit* aus, ohne die nicht von einem Prozess-Management gesprochen werden kann. Aus Sicht von Systemansatz und Kybernetik beschreibt Stöger Prozess-Management als die Steuerung von Abläufen in einer Organisation als System, damit diese die Identität erhalten und den Zweck erfüllen kann (ebenda).

---

[1] Unter dem Begriff *Model Driven Software Development* (MDSD) bestehen Konzepte zur teilautomatisierten Code-Generierung (s. u. v. a. Lan09, PM07, SVEH07).

[2] Der Begriff des Geschäftsprozess-Managements ist unscharf, z. T. wird das von HC93 propagierte, einmalig durchzuführende Business Process Reengineering darunter subsumiert, so dass beim hier beschriebenen Geschäftsprozess-Management in der Literatur auch vom kontinuierlichen Geschäftsprozess-Management (NPW08) oder vom Business Process Improvement (Kül99) gesprochen wird.

Gad02 siedelt das Prozess-Management auf der fachlich-konzeptionellen Ebene an, wo die Ableitung der Prozesse aus den Vorgaben der darüber liegenden strategischen Ebenen erfolgt und damit die Verbindung zur Unternehmensplanung darstellt. Häs00 versteht unter Prozess-Management alle inhaltlichen, ablaufbezogenen und kulturellen Managementfunktionen, die im Zusammenhang mit dem Einsatz einer Prozessorganisation auszuüben sind.

Rüt90 betont den kontinuerlichen Charakter des Prozess-Managements als „die permanente Analyse und Verbesserung wichtiger interdisziplinaärer Arbeitsabläufe in einem Unternehmen“, und auch AK06 betonen die Kontinuität, indem sie vom permanenten Management von Geschäftsprozessen sprechen.

Falls keine oder keine durchgängige Prozessorganisation vorherrscht, sind vor der Etablierung eines Geschäftsprozess-Managements die entsprechenden Organisationsstrukturen zu schaffen und Kern-, Führungs- und Supportprozesse zu definieren, zu modellieren und zu implementieren (Kap. 2.1, s. dazu u. v. a. AK06, BMW09, FD08, Gad02, OF03).

1.2 Die Prozessorganisation

Mit der Prozesssicht, die sich in der Geschäftsprozessorganisation manifestiert, wird der Blick auf den Kunden gerichtet[3], es geht um die „Erfüllung des Kundenauftrages“ (OF03), die Abnehmerbedürfnisse treten in den Vordergrund (Häs00). Indem Prozesse abteilungsübergreifend gestaltet werden, entfallen die Schnittstellen zwischen den einzelnen Aufbauorganisationseinheiten; der für einen Prozess verantwortliche Prozesseigner koordiniert die von den einzelnen Stellen zu erbringenden Leistungen von aussen. Damit sind die Voraussetzungen für einen effizienten und raschen Ablauf gegeben.

Neu ist die Prozesssicht nicht. Bereits bei Nordsieck (Nor34) und Kosiol (Kos62) taucht der Prozessgedanke auf. Hingegen war es Gaitanides, der einen zur klassischen Aufbau- und Ablauforganisation überarbeiteten, prozessorientierten Ansatz vorstellte (Gai83). Gaitanides stellte die zeitliche und inhaltliche Verteilung von Aufgaben und Koordination *ziel*orientierter Hand-

[3] Mit Kunde ist hier der Empfänger der erbrachten Leistung gemeint, dies kann eine Person, eine Organisation oder ein Teil davon in Form eines nachfolgenden Prozesses innerhalb oder ausserhalb der erbringenden Organisation sein (s. a. FD08).

lungen der Organisationsmitglieder ins Zentrum und betrachtete die Aufbauorganisation als der Ablauforganisation nachfolgend („structure follows process“). Spätestens mit dem Erscheinen des von Hammer und Champy geprägten Begriffes des „Business Process Reengineering“ (HC93) ist die prozessorientierte Sicht auf die Unternehmensorganisation ins allgemeine Bewusstsein gerückt und wird seither intensiv dikutiert, auch wenn deren Vorschlag, bestehende Unternehmensstrukturen zu zerschlagen und neu aufzubauen („Reengineering is the fundamental rethinking and radical redesign of business processes [...]“), vielerorten als zu radikal kritisiert und verworfen wurde.

1.3 Zielsetzung und Forschungsfragen

Ziel der vorliegenden Arbeit ist es, die für die Unterstützung des Prozess-Managements entscheidenden Merkmale eines Prozessmodells zu identifizieren und einen Vorschlag eines Prozessmodells zu unterbreiten.

Die optimierte Unterstützung der betriebswirtschaftlichen Prozesssteuerung, die Abbildung unterschiedlicher Daten mit unterschiedlicher Zielsetzung sowie der Grad der Generizität eines Modells führen zu folgenden 4 Forschungsfragen:

1. Modell: Was zeichnet ein qualitativ hochstehendes[4] Modell aus, d. h., wie soll ein Modell beschaffen sein, damit es die aus dem Prozess-Management hervorgehenden Anforderungen bestmöglich zu erfüllen vermag?

2. Modellsprache: Wie soll eine Modellsprache, die geeignet ist, ein qualitativ hochstehendes Modell zu erstellen, beschaffen sein?

3. Wie ist die Beschaffenheit von zwei in der Praxis weitverbreiteten Methoden (EPK als Vertreter der Wirtschaftsinformatik und Aktivitäten aus UML als Vertreter der Informatik) und von Petri-Netzen sowie von den mit diesen Methoden erzeugten Modellen?

[4] „Qualität“ wird hier und im folgenden in Anlehnung an ISO 9000 als der Grad, in dem ein Satz inhärenter Merkmale Anforderungen erfüllt, verstanden.

4. Wie sollte eine Prozessmodellmethode aussehen, damit die Antwort auf Forschungsfragen 1 und 2 erfüllt ist, und wie weit erfüllen die vorgeschlagenen Authentic Petri Net (APN) die Anforderungen?

1.4 Forschungsmethodik

Die Wirtschaftsinformatik nimmt seit jeher die Rolle einer Mittlerin zwischen den Belangen der Wirtschaftswissenschaften, primär der Betriebswirtschaftslehre, und den Methoden der Informatik ein. Die Wirtschaftsinformatik, zu der sich die vorliegende Arbeit zählt, gilt seit mehr als 20 Jahren als eigenständige Disziplin (LWS08). Ihre Wurzeln reichen jedoch weiter zurück. So erwähnt Sch07 das im Jahre 1958 gegründete Institut für Automation und Operations Research[5] der Universität Freiburg i. Ue. als den ersten Lehrstuhl der Wirtschaftsinformatik überhaupt. Ihren Ursprung hat die Wirtschaftsinformatik im damaligen technologischen Umschwung. In den 1940er Jahren wurden Rechenanlagen für die Forschung entwickelt, und ab 1950 kamen erste Grossrechner für die Verarbeitung betrieblicher Daten auf den Markt (Sch07). Diese angewandte Informatik ist denn auch Begründung der Wirtschaftsinformatik. Ihr Gegenstand sind Informationssysteme in Wirtschaft und Verwaltung. Die Betrachtung umfasst die Aufgaben der Informationssysteme, den Menschen als Aufgabenträger sowie die Entwicklung und das Management betrieblicher Anwendungssysteme (vgl. FS08, HS09, LWS08); LWS08 erwähnen zusätzlich die (Weiter-)Entwicklung von Theorien zur betrieblichen Informationsverarbeitung.

Zwar wird die Wirtschaftsinformatik auf Grund ihrer Zugehörigkeit zu den Wirtschaftswissenschaften zu den Sozialwissenschaften gerechnet, doch findet sich unter den klassischen Untersuchungsmethoden der Sozialwissenschaften keine, die für den Untersuchungsgegenstand geeignet scheint. Die Hauptproblematik, die sich auf der Suche nach einer geeigneten Methode stellte, war die Absicht, zu Beginn einen Artefakt zu erzeugen[6] und diesen anschliessend einer Bewertung zu unterziehen. Diese Aufgabe wird exakt im von MS95 vorgeschlagenen Vorgehensmodell abgedeckt. Das Vorgehensmo-

[5] Gemäss KAPP08 war die Bezeichnung eingedeutscht und lautete „Institut für Automation und Unternehmensforschung (IAUF)“.

[6] genauer: einen bestehenden Artefakt aus Fra01 zu übernehmen und zu überarbeiten

dell basiert auf dem Design-Science-Paradigma, welches problemlösungsorientiert ausgerichtet ist und seine Wurzeln in den Ingenieurdisziplinen und den Sciences of the Artificial hat (Sim96).

Design Science besteht aus den zwei Basisaktivitäten *build* (zu deutsch etwa „bilden, errichten“) und *evaluate* („bewerten, evaluieren“). Erstere ist der Prozess zur Errichtung eines Artefakts mit einem bestimmten Ziel, letztere der Prozess, die Leistung des Artefakts zu bestimmen (MS95). Als weitere Basisaktivitäten, jedoch aus dem naturwissenschaftlichen Umfeld, nennen MS95 *theorize* und *justify*. Aus der Basisaktivität *build* resultieren die Ergebnisse *constructs* („Konstrukte“), *models* („Modelle“), *methods* („Methoden“) und *instantiations* („Instantiierungen, Instanzen, Implementierungen“), welche mittels der Basisaktivität *evaluate* auf ihre Nützlichkeit hin untersucht werden.

In HM03, HMPR04 wird auf Basis der Arbeiten von MS95 ein konzeptionelles Rahmenwerk für die Informationssystemforschung (s. Abb. 1.1) vorgestellt. Dieses kombiniert das behavioristische Paradigma mit dem des Design Science (ebenda). Auf Grund seiner Ausrichtung auf wirtschaftswissenschaftliche Bedürfnisse innerhalb eines abgesteckten Untersuchungsfeldes und seines Beitrags zur Wissensbasis für weitere Untersuchungen liefert das Rahmenwerk Hilfestellung bei der Beurteilung der jeweiligen Beiträge beider Paradigmen.

Sowohl MS95 wie HM03, HMPR04 betonen die Nützlichkeit als Ziel des Design-Science-Paradigmas[7], welches gemäss HM03 versucht, „to create innovations, or *artifacts*, that embody the ideas, practices, technical capabilities, and products required to efficiently accomplish the analysis, design, implementation, and use of information systems“. HMPR04 geben dem Leser einen Leitfaden in Form von 7 Richtlinien in die Hand[8]:

1. Design as an Artifact: Design-Science-Forschung erzeugt einen realisierbaren Artefakt in Form eines Konstrukts, Modells, einer Methode oder Instantiierung.

[7] ohne dabei die Wahrheit als Ziel der behavioralistischen Wissenschaft aus den Augen zu lassen

[8] Die Beschreibung wurde vom Autor ins Deutsche übersetzt, die Bezeichnung aus Gründen der Genauigkeit in Englisch belassen.

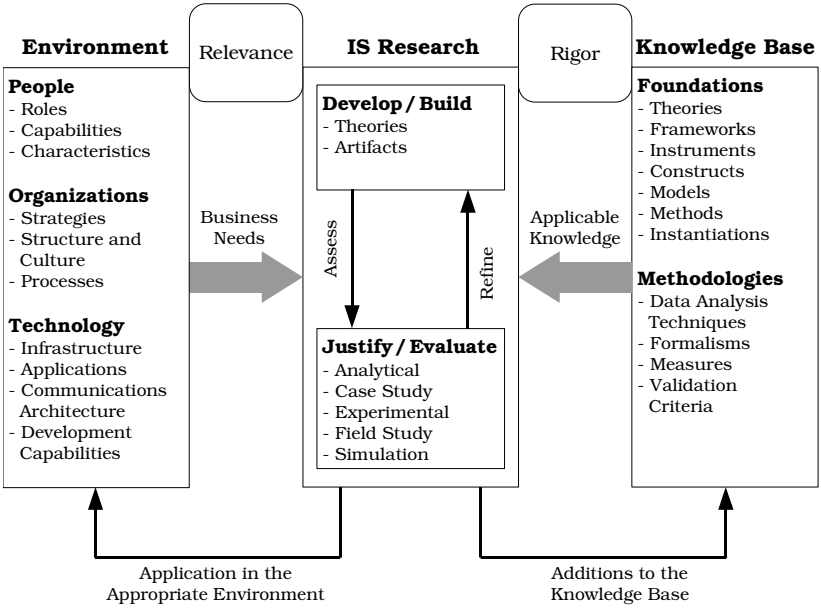


Abbildung 1.1: *Informationssystemrahmenwerk. Nach HMPR04.*

2. Problem Relevance: Ziel der Design-Science-Forschung ist es, technologiegestützte Lösungen zu bedeutenden Problemen aus der Wirtschaft zu entwickeln.

3. Design Evaluation: Nützlichkeit, Qualität und Effizienz eines Design-Artefakts werden mittels geeigneter Evaluation fundiert nachgewiesen.

4. Research Contributions: Wirksame Design-Science-Forschung erzeugt klare und nachweisbare Beiträge auf den Gebieten der Design-Artefakte, Design-Grundlagen und/oder Design-Methoden.

5. Research Rigor: Design-Science-Forschung stützt sich auf die Anwendung formaler Methoden hinsichtlich Konstruktion und Evaluation des Design-Artefakts.

6. Design as a Search Process: Die Suche nach einem zielführenden Artefakt setzt voraus, dass die vorhandenen Mittel zur Erreichung des gewünschten Ergebnisses unter Einhaltung der gegebenen Rahmenbedingungen im Problemumfeld eingesetzt werden.

7. Communication of Research: Design-Science-Forschung muss so kommuniziert werden, dass sie wahrgenommen wird, und zwar sowohl gegenüber der technisch- wie der management-orientierten Interessensgruppe.

1.5 Vorgehen, Ergebnisse und Aufbau der Arbeit

Die Zielsetzung aus Abschnitt 1.3 entspricht den *Business Needs* in Abb. 1.1. Diese werden in Form von Qualitätskriterien zur Beurteilung von Geschäftsprozessmodellen in Kap. 2 unter Beizug vorhandener Grundlagen und Methodiken aus der Wissensbasis (s. Abb. ebenda) hergeleitet und beschrieben und dienen innerhalb des Untersuchungsprozesses der Beurteilung der vorgeschlagenen Modelliermethode, entsprechend dem *Assess* in der Abb. (ebenda).

In Kap. 3 werden in einem ersten Schritt zwei weitverbreitete Prozessmodelle als Vertreter der Grundlagen aus der Wissensbasis in Abb. 1.1 vorgestellt und an den in Kap. 2 erarbeiteten Kriterien gemessen. Das eine Prozessmodell nimmt die Sicht aus Betriebswirtschaft und Wirtschaftsinformatik, das andere die Sicht aus Informatik/Wirtschaftsinformatik ein.

Nicht alle Kriterien werden gleichermassen erfüllt. In Kap. 4 wird daher mit den Petri-Netzen ein Vertreter aus der theoretischen Informatik hinzugezogen und ebenfalls an Hand der Kriterien beurteilt.

Trotz ihrer Einfachheit und ihrem Formalismus werden Petri-Netze nur zögerlich in der betriebswirtschaftlichen Modellierung eingesetzt. Daher wird in Kap. 5 eine Variante von Petri-Netzen – die Authentic Petri Net (APN) – hergeleitet und als Artefakt an den Kriterien gemessen und verfeinert.

Eine Überführung in das Anwendungsumfeld (*Application in the Appropriate Environment*, Abb. 1.1) wird an Hand eines einfachen Beispiels aus dem E-Commerce nach einer kurzen Einführung in die Implementierung von Web-Anwendungen in Kap. 6 gezeigt. Die Beiträge zur Wissensbasis (*Additions to*

the Knowledge Base, ebenda) und offengebliebene Fragen werden abschliessend in Kap. 7 diskutiert.

Kapitel 2

Die Geschäftsprozessmodellierung und die Herleitung von Beurteilungskriterien

2.1 Die Rolle der Modellierung im Geschäftsprozess-Management

Für den Aufbau und die Etablierung eines Geschäftsprozess-Managements existieren verschiedene Vorgehensmodelle, die nach einer Vorbereitungs- und einer Initialisierungsphase die Erhebung des aktuellen Problemumfeldes mit anschliessender Synthese und Implementierung vorsehen und als letzte Phase die wiederkehrende Optimierung der Prozesse zum Inhalt haben[1].

Damit das Prozess-Management in der Lage ist, die notwendige Unterstützung zum Erreichen der strategischen und operativen Vorgaben zu leisten, wird ein Modell benötigt, welches es erlaubt, eine Prozesslandkarte sowie Zuständigkeiten, Messgrössen, Lokalitäten, Ressourcenbedarf, zu erstellende Leistung u. w. zu erfassen. Zusätzlich stellt sich die Frage, wie das Modell zielgruppengerecht aufbereitet werden kann, bspw. indem es möglich ist, unterschiedliche Sichten zur Verfügung zu stellen. Letztlich soll die Prozessanalyse, -gestaltung und -umsetzung gemäss KK97 unterstützt werden.

[1] Beispiele sind das Vorgehensmodell von FD08 und dasjenige von Stö09.

Nach Stachowiak (Sta73) sind „Modelle stets Modelle von etwas, nämlich Abbildungen, Repräsentationen natürlicher oder künstlicher Originale, die selbst wieder Modelle sein können“ (Abbildungsmerkmal). Als Original kann jede von einem natürlichen oder maschinellen kognitiven Subjekt erfahrbare Entität verstanden werden (ebenda).

„Modelle erfassen im allgemeinen nicht alle Attribute des durch sie repräsentierten Originals, sondern nur solche, die den jeweiligen Modellerschaffern und/oder Modellbenutzern relevant scheinen“ (Sta73). Dieses Weglassen von nichtrelevanten Attributen bezeichnet Stachowiak als das Verkürzungsmerkmal, das Teile des pragmatischen Merkmals vorwegnimmt, welches den Nutzen und den Zweck eines Modells hervorhebt: „Modelle sind ihren Originalen nicht per se eindeutig zugeordnet. Sie erfüllen ihre Ersetzungsfunktion a) für bestimmte – erkennende und/oder handelnde, modellbenutzende – Subjekte, b) innerhalb bestimmter Zeitintervalle und c) unter Einschränkung auf bestimmte gedankliche oder tatsächliche Operationen“ (ebenda).

2.2 Herleitung von Kriterien zur Bewertung von Geschäftsprozessmodellen

Konzeptuelle Modelle sind eine Sammlung von Spezifikationen zu einem bestimmten Problem (vgl. LSS94, KSJ06). Konzeptuelle Modelle sind formal, und ihre Objekte haben einen direkten Bezug zu den Entitäten der Diskurswelt (vgl. Sim94). Sie dienen dazu, den Entwurf und die Analyse von Informationssystemen innerhalb des Entwicklungsprozesses zu unterstützen, indem sie eine innerhalb der betreffenden Domäne gültige Diskussionsbasis schaffen (vgl. RW05, Fra98, MPGP05). Zur Beschaffenheit von konzeptuellen Modellen, zu denen auch die Prozessmodelle gezählt werden können, entstehen laufend neue Arbeiten[2].

Im Unterschied zu gängigen wissenschaftlichen Aussagen sind konzeptuelle Modelle nicht unbedingt darauf ausgerichtet, den jeweils interessierenden Wirklichkeitsausschnitt authentisch zu beschreiben (Fra00), vielmehr hängt das Ergebnis vom eingenommenen Standpunkt und vom Zweck der Modellierung ab. Daher sind Versuche, Aussagen zur semantischen Korrekt-

[2] Allerdings sei bereits an dieser Stelle vermerkt, dass eine völlig objektive Beurteilung von konzeptionellen Modellen und vor allem von Modellierungssprachen nicht zu erwarten ist (Fra00).

heit von Informationsmodellen herzuleiten (vgl. HA04) problematisch. Ferner ist gemäss Fra00 die Beurteilung der verwendeten Modelliersprache Voraussetzung für die Bewertung eines Modells: Wenn eine Modelliersprache bspw. bestimmte semantische Konstrukte nicht bereitstellt, können die zugehörigen Modelle diese Konstrukte nicht ausdrücken; dieser Mangel darf dann nicht den Modellen angelastet werden. Es stellt sich daher bei der Frage nach der Qualität von Modellen (vgl. 1.3, 1) auch die nach der Qualität der dazugehörigen Modellsprache (vgl. 1.3, 2) und die nach der Qualität des Modellierprozesses.

Das Forschungsgebiet zur qualitativen Bewertung von Modellen ist noch jung (NPGP05, Rec06), erkennbar daran, dass es zwar eine Reihe von Arbeiten gibt, diese jedoch auf unterschiedlichen Ansätzen beruhen und nicht geeignet sind, direkt allgemeingültige Empfehlungen abzuleiten: Die Qualität als Mass der Erfüllung von Anforderungen kann nicht losgelöst vom Verwendungszweck beurteilt werden.

Arbeiten zur Bewertung von konzeptuellen Modellen sind als Kriterienliste (MS94, BRS95, BRU00, Sch97), als Massmodell (vtKB03, MPGP05, RRIG06) oder als Rahmenwerk (LSS94, KSJ06, vHPv07, Rec05) formuliert.

Im folgenden werden die oben aufgeführten Ansätze vorgestellt, um anschliessend einen Vorschlag einer Kriterienliste für die Evaluation von Geschäftsprozessmodellen vorzustellen. Dabei kann es sich lediglich um einen Entwurf von vorübergehender Aussagekraft handeln: Der Versuch, die konzeptuelle Modellierung auf eine theoretische Basis zu stellen, muss zum heutigen Zeitpunkt als nicht erreicht bezeichnet werden. Die vorgeschlagene Kriterienliste ist daher als pragmatischer Ansatz im Sinne eines „Best of Breed“ zu verstehen, stets aus der Perspektive der Wirtschaftsinformatik, mit dem Ziel, in der Praxis auftauchende Fragestellungen für die an der Modellierung direkt beteiligten Parteien zufriedenstellend zu beantworten.

2.2.1 Sichtenübergreifende Kriterien

2.2.1.1 Die Grundsätze ordnungsmässiger Modellierung nach Becker, Rosemann, Schütte

Basierend auf den oben erwähnten Arbeiten zur Qualität von Datenmodellen (BFN85, BCN91, MS94, s. a. 2.2.2.1), schlagen BRS95 in Analogie zu den

„Grundsätzen ordnungsmässiger Buchführung GoB“ Grundsätze ordnungsmässiger Modellierung GoM vor. Dabei handelt es sich um einen Bezugsrahmen in Form von Gestaltungsempfehlungen für die Informationsmodellierung, der einen Beitrag zur Erhöhung der Qualität von Informationsmodellen leisten soll (RSD08)[3]. Je nach Ziel des zu schaffenden Modells werden die Gestaltungsempfehlungen entsprechend konkretisiert. Die Konkretisierung erfolgt in unterschiedlichen Sichten, in GoM vorgeschlagen wird ein Ordnungsrahmen, der sich an der ARIS-Architektur nach Sch92 orientiert und die Daten-, die Funktions- und die Organisationsschicht umfasst. Die Gestaltungsempfehlungen setzen sich aus den folgenden sechs Grundsätzen zusammen:

1. Der Grundsatz der *Richtigkeit* richtet sich sowohl an die syntaktische wie an die semantische Korrektheit. Ein Modell ist dann syntaktisch richtig, wenn es vollständig und konsistent gegenüber dem ihm zugrundeliegenden Metamodell und damit formal korrekt ist. Für die semantische Richtigkeit muss das Modell struktur- und verhaltenstreu gegenüber dem Objektsystem sein und sowohl in sich wie zu anderen Modellen widerspruchsfrei sein (BRS95, Sch98b).

2. Ein Modell erfüllt dann den Grundsatz der *Relevanz*, wenn der reale Ausschnitt so gewählt wird, dass die mit der Modellierung verfolgten Ziele erreicht werden können, und alle ausschlaggebenden Modellelemente enthalten sind. Das Modell ist genau dann relevant, wenn bei Weglassen von Informationen der Nutzen des Modells sinkt (BRS95, Sch98b).

3. Mit dem Grundsatz der *Wirtschaftlichkeit* soll sichergestellt werden, dass die Modellierungsaktivitäten in einem angemessenene Kosten-Nutzen-Verhältnis zueinander stehen. Durch die Nutzung von Referenzmodellen und Massnahmen zur Wiederverwendung kann gemäss RSD08 die Wirtschaftlichkeit gefördert werden.

[3]Schütte distanziert sich später in Form der *GoM II* von einigen Überlegungen der GoM und ersetzt die beiden Forderungen nach Richtigkeit und Relevanz durch die der *Konstruktionsadäquanz* und der *Sprachadäquanz* und misst dem erkenntnistheoretischen Sachverhalt mehr Bedeutung zu (vgl. dazu Sch97, Sch98b, Fra00). Gemäss Beobachtungen von Schütte hat dies allerdings zur Folge, dass die Anwender die GoM II als weniger verständlich einstufen (Sch99).

4. Der Grundsatz der *Klarheit* besagt, dass ein Modell so auf den Adressaten zugeschnitten sein soll, dass dieser es versteht. Von BRS95, Sch98b erwähnt werden die Lesbarkeit, die Strukturiertheit und die Übersichtlichkeit; nach RSD08 kann die intuitive Lesbarkeit dadurch gewährleistet werden, dass die zum Modellverständnis vorausgesetzten methodischen Kenntnisse gering gehalten werden.

5. Mit dem Grundsatz der *Vergleichbarkeit* wird die modellübergreifend konforme Anwendung der Modellierungsempfehlungen gewährleistet, was sich bspw. in einer vereinfachten Konsolidierung unabhängig voneinander erstellter Modelle in einem umfassenden Modellierungsprojekt niederschlagen kann (RSD08). BRS95 unterteilen noch in eine syntaktische und eine semantische Komponente, wobei der syntaktische Teil die Kompatibilität von mit unterschiedlichen Methoden erstellten Modellen umfasst und der semantische Teil die inhaltliche Modellvergleichbarkeit diskutiert.

6. Der Grundsatz des *systematischen Aufbaus* fordert die Einordnung von Modellen in eine Informationssystemarchitektur, welche einen strukturierenden Rahmen für unterschiedliche Beschreibungssichten bildet (Sch98b, BRS95), damit eine sichtenübergreifende Konsistenz der einzelnen Sichtenmodelle gegeben ist.

2.2.1.2 Ontologiebasierte Analyse

Ontologie ist die philosophische Disziplin des Seins. Sie geht auf Aristoteles zurück und beschäftigt sich mit der Beschaffenheit und der Struktur von Gegenständen, Eigenschaften, Ereignissen, Abläufen und Beziehungen in der Realität (Smi09). Der Begriff findet seit einigen Jahren vermehrt auch ausserhalb der Philosophie Verwendung. Angefangen bei der Künstlichen Intelligenz und dem Wissens-Management, mittlerweilen in etlichen Disziplinen, wie u. v. a. der Biologie, Medizin, den Geowissenschaften und Informationswissenschaften, entstehen unter dem Begriff *Ontology Engineering* Ontologien, mit dem Ziel, eine allgemeingültige Verständigungsebene zu schaffen und mittels Inferenzregeln Aussagen ableiten zu können.

GG95 identifiziert ausserhalb der Philosophie 6 Begriffsverwendungen, die unter Ontologie entweder eine formale oder informale begriffliche Entität ver-

stehen und damit den semantischen Aspekt betonen oder die die Syntaktik ins Zentrum der Betrachtung stellen und die Ontologie als Vokabular oder Spezifikation einer Logik verstehen. Gru93 definiert eine[4] Ontologie als „[...] an explicit specification of a conceptualization“, GG95 präzisiert *conceptualization* als „an intensional semantic structure which encodes the implicit rules constraining the structure of a piece of reality“. Als Definition einer Ontologie[5] schlägt GG95 vor: „([S]ense 1) a logical theory which gives an explicit, partial account of a *conceptualization*; (sense 2) synonym of *conceptualization*.“ Smi98 schlägt gleichermassen die terminologische Unterscheidung in „referent- or reality-based R-ontology“ und „elicited or epistemological E-ontology “ vor (vgl. a. Fon07). Als *R-Ontologie* bezeichnet Smi98 die Theorie, wie eine gegebene Bezugsdomäne (die das ganze Universum umfassen kann) strukturiert ist, welcher Art die darin enthaltenen Entitäten und deren Beziehungen untereinander sind usw. Die *E-Ontologie* bezeichnet Smith als die Theorie, wie ein gegebenes Individuum, eine gegebene Gruppe, Sprache oder Wissenschaft eine gegebene Domäne in Begriffe fasst, als die Theorie des ontologischen Inhalts gewisser Repräsentationen (ebenda).

Der Begriff der Ontologien hat sich in den letzten Jahren auch in den Informationswissenschaften etabliert. Gemäss neuer Definition von Gru09 verstehen die Informationswissenschaften unter einer Ontologie „[...] a set of representational primitives with which to model a domain of knowledge or discourse“, wobei „representational primitives“ typischerweise Klassen (oder Mengen), Attribute (oder Eigenschaften) und Beziehungen (oder Beziehungen unter Klassenmitgliedern) sind, definiert durch Angabe ihrer Bedeutung[6] und ihrer Einschränkungen hinsichtlich einer logisch konsistenten Anwendung (ebenda). Nach SZ99, DSZ03 koexistieren mehrere Ontologien nebeneinander, welche von Menschen geschaffene Artefakte darstellen, konstituiert durch ein spezifisches Vokabular, mit welchem ein bestimmter Ausschnitt der Realität beschrieben wird (Gua98).

[4] Man beachte den Gebrauch des unbestimmten Artikels; die Definition von Gruber schliesst das Vorhandensein mehrerer Ontologien nebeneinander mit ein.

[5] nebst dem Begriff Ontologie als ein philosophischer Zweig, der sich mit der Natur und dem Zusammenhalt der realen Welt beschäftigt

[6] Dies führt direkt zum Begriff des Semantic Web, welches sich zum Zwecke des Informationsgewinns mittels logischen Schliessens einer Taxonomie und einer Reihe von Inferenzregeln (Ontologie) bedient. Zur Beschreibung dieser Ontologie wird die Web Ontology Language (OWL) benutzt. Zum Semantic Web s. u. a. BL00, HKR09, Av08.

Auch wenn die vorhandenen Ontologien unterschiedlich sind, verfügen sie über viele Gemeinsamkeiten (s. CJB99):

- Es gibt *Objekte* auf der Welt.
- Objekte haben *Eigenschaften* oder *Attribute*, die *Werte* annehmen können.
- Objekte können *Teile* haben.
- Objekte können in verschiedenen *Beziehungen* untereinander vorkommen.
- Eigenschaften und Beziehungen können mit der *Zeit* ändern.
- Zu verschiedenen *Zeitpunkten* treten *Ereignisse* ein.
- Von Zeit zu Zeit gibt es *Prozesse*, an welchen Objekte teilhaben.
- Die Welt und ihre Objekte können sich in verschiedenen *Zuständen* befinden.
- Ereignisse können als *Effekt* weitere Ereignisse *auslösen* oder weitere Zustände herbeiführen.

Wand und Weber widmen sich der Frage nach der Abbildung eines konzeptuellen Modells aus der realen Welt in ein Informationssystem: „Given a user's or a group's conception of the real world, under what conditions would an information system provide a good or a poor representation of this conception?" (Web97, zitiert in Wys06).[7] Dazu haben Wand und Weber auf Basis der Arbeiten von Bunge (insbes. Bun77, Bun79, zitiert in GR02) zu Ontologien die *Bunge-Wand-Weber-Modelle* (*BWW-Modelle*) entworfen. Wand und Weber setzen voraus, dass eine Modelliersprache imstande sein muss, alle in der realen Welt existierenden Dinge, die für den Modellierer und den Anwender des Modells von Interesse sein könnten, abzubilden (GR02). Als Beitrag zur konzeptuellen Modellierung schlagen Wand und Weber (zitiert in DGMR05) ein

[7] Nicht von Interesse sind für Wand und Weber das Verhältnis der realen Welt zum konzeptuellen Modell der realen Welt. Dieser Umstand wird kontrovers diskutiert, da sich die Ontologie nach philosophischem Verständnis um die Beschaffenheit der realen Welt kümmert, nicht um deren Konzeptualisierung, wie es Wand und Weber tun.

Rahmenwerk vor, bestehend aus den 4 Elementen *Grammatik*, *Methode*, *Skript* und *Kontext*: Die Methode beschreibt, wie in einem bestimmten Kontext die von der Grammatik zur Verfügung gestellten Konstrukte und Regeln zu deren Herleitung zu einem Skript – einem konkreten Modell – zusammengefügt werden.

Mittels Ontologien lassen sich formal Modelliersprachen auf verschiedene Eigenschaften hin untersuchen (RRIG09). Unterstellt wird dabei, dass die der Evaluation zugrunde gelegte Ontologie geeignet ist, die in der realen Welt vorkommenden relevanten Elemente und deren Beziehung untereinander möglichst gut zu beschreiben. Dazu wird versucht, eine bidirektionale Abbildung zwischen den Konstrukten der Ontologie und den Konstrukten der zu untersuchenden Modellsprache herzustellen. Ist eine direkte 1:1-Beziehung nicht möglich, führt dies zu einem Repräsentationsdefizit, welches zu einer Konfusion beim Anwender führen kann (vgl. Web97, zitiert in RRIG06, GR01, GR02, FL03, RRIG06, Rec07).

GR05 machen darauf aufmerksam, dass ein Nichtübereinstimmen nicht notwendigerweise auf Mängel der Modelliersprache zurückzuführen ist, sondern ein Indiz dafür sein kann, dass die gewählte Ontologie mangelhaft spezifiziert ist. Damit relativieren GR05 den Anspruch der Allgemeingültigkeit einer Ontologie und schlagen vor, die zugrunde gelegte Ontologie den eigenen Bedürfnissen anzupassen, indem einzelne Konstrukte vertieft spezifiziert, weggelassen oder umbenannt werden.

Die häufig zur Beschreibung von Ontologien zur Anwendung gelangenden formalen Sprachen, wie semantische Netzwerke, konzeptuelle Graphen, Austauschformate, logische Sprachen u. a. m. eignen sich gut für automatisierbare Prozesse wie z. B. die Deduktion von Aussagen. Ungeeignet sind sie für die Kommunikation innerhalb der Modellierung, da sie ein gutes mathematisches Verständnis voraussetzen und deren Interpretation sich nicht ohne weiteres den im Modellierprozess beteiligten Personen erschliesst. DGMR05, GR05 schlagen daher die Überführung der Ontologie in ein Metamodell vor. Als Metasprache bedienen sie sich des auf Chen (Che76) zurückgehenden Entity-Relationship-Modells, erweitert um die Möglichkeit der Generalisierung, und zeigen an Hand konkreter Beispiele die Anwendung zum Vergleich zweier Ontologien (DGMR05) und mehrerer Modelliersprachen (GR05).

Ausgehend von einer Ontologie, erlaubt es die ontologiebasierte Analyse, Aussagen zur ontologischen Vollständigkeit und zur ontologischen Klarheit

zu machen. Abhängig vom Modellzweck, kann eine bestehende Ontologie an die Bedürfnisse der Untersuchung angepasst werden (GR05), um zum vornherein bekannte Diskrepanzen ausschliessen zu können. Die ontologiebasierte Analyse gestattet dank der zugrunde gelegten Ontologie als Bezugspunkt den durchgängigen, fundierten Vergleich von verschiedenen Modellsprachen, Modellen und Ontologien bzgl. ihrer Abbildungsmächtigkeit untereinander und wird auch als Repräsentationsanalyse bezeichnet (vgl. AM06, RRGI06, RM07).

2.2.1.3 Semiotisches Rahmenwerk

Das ursprüngliche semiotische Rahmenwerk von LSS94 setzt die vier Modellieraspekte *Sprache*, *Domäne*, *Modell* und *Interpretation durch die Zielgruppe* der Syntaktik, der Semantik und der Pragmatik zueinander in Beziehung. Die Begründung auf der Semiotik und der Linguistik ist aus der Überlegung entstanden, dass ein konzeptuelles Modell als eine Menge von Aussagen in einer Sprache aufgefasst werden kann (ebenda). Das Rahmenwerk wurde mittlerweilen in *SEQUAL Framework* benannt und wird laufend erweitert (KSJ06). Hinzugekommen sind u. a. weitere Qualitätsebenen, die auf der semiotischen Leiter („semiotic ladder“) von Sta96 beruhen (KSJ06). Abb. 2.1 zeigt das Rahmenwerk in seiner Form von 2006 mit den folgenden Entitäten:

- G: Die (normalerweise organisatorisch festgelegten) Ziele der Modellierung.
- L: Die Ausdrucksmächtigkeit der eingesetzten Modellsprachen.
- D: Die Domäne als die Menge aller möglichen Aussagen zum Untersuchungsgegenstand.
- M: Das externalisierte Modell als die Menge aller Aussagen des Modellierers, entstanden aus dessen Wahrnehmung.
- K_S: Das explizite Wissen der an der Modellentstehung interessierten Instanzen (Personen und Modellierwerkzeuge).
- K_M: Das explizite Wissen des Modellierers.

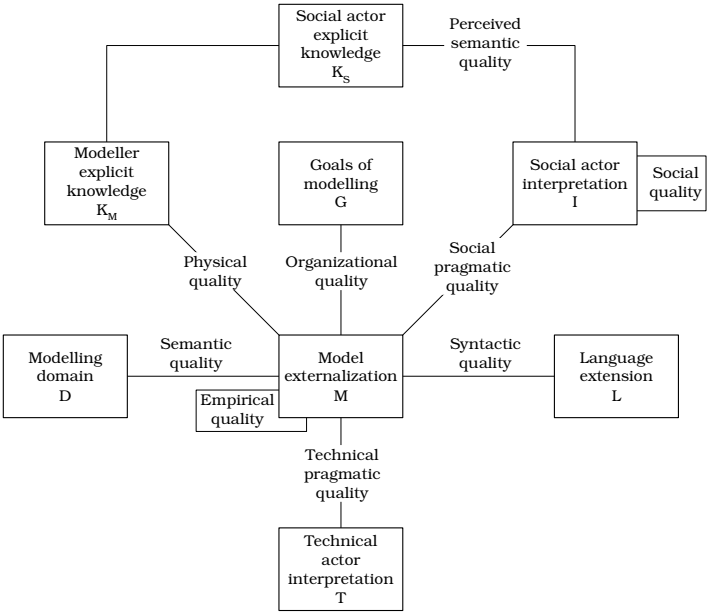


Abbildung 2.1: *Das Qualitätsrahmenwerk SEQUAL (nach KSJ06).*

- *I*: Die Interpretation von *M* zu einem bestimmten Zeitpunkt durch das Publikum.
- *T*: Die Interpretation durch Modellierwerkzeuge.

Die Entitäten sind durch Qualitätsanforderungen an das Modell auf den verschiedenen semiotischen Ebenen untereinander verknüpft:

- *Physical quality*: Die Beständigkeit und die Verfügbarkeit.
- *Empirical quality*: Die Verständlichkeit und Ästhetik.
- *Syntactic quality*: Die formale syntaktische Korrektheit.
- *Semantic quality*: Die (hinreichende) Gültigkeit und die (hinreichende) Vollständigkeit.

- *Perceived semantic quality*: Die wahrgenommene Gültigkeit und Vollständigkeit.
- *Pragmatic quality*: Die Verständlichkeit.
- *Social quality*: Die Übereinkunft bzgl. K_S, I und M.
- *Organisational quality*: Der Grad der Zielerreichung durch das Modell.

Hinzu kommt die *Language quality*, welche wie in Abb. 2.2 gezeigt die Modelliersprachen zu den Entitätsmengen setzt. (NK05) identifizieren und umschreiben die in Tab. 2.1 aufgeführten Bereiche, zusammen mit den Qualitätszielen, zu deren Erreichen sie in erster Linie einen Beitrag leisten.

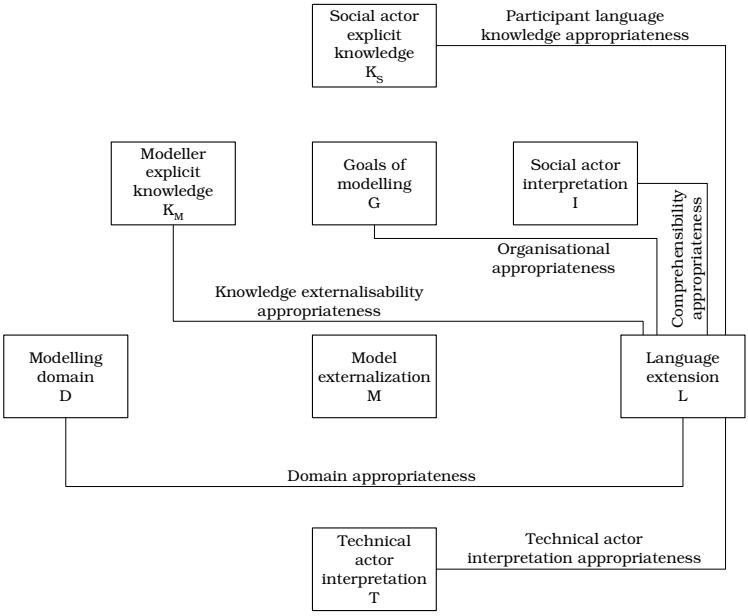


Abbildung 2.2: *Die Language quality des SEQUAL-Rahmenwerks (nach NK05).*

Bereich	Beitrag zu Qualitätsziel
Domain appropriateness	semantisch
Participant language knowledge appr.	physikalisch, pragmatisch
Knowledge externalisability appr.	physikalisch
Comprehensibility appropriateness	empirisch, pragmatisch
Technical actor interpretation appr.	syntaktisch, semantisch, pragm.
Organisational appropriateness	organisatorisch

Tabelle 2.1: *Modellsprachenqualitätsbereiche (links) und deren Beitrag an die Qualitätsziele (rechts).*

Das SEQUAL-Rahmenwerk bietet umfassende Hilfestellung bei der Beurteilung von Modellen und beim Erstellen dieser Modelle, indem es die verschiedenen Partizipanten zueinander in Beziehung setzt und darüber hinaus Hinweise gibt, wie die Qualität der Beziehungen beeinflusst wird. Insbesondere der Beitrag zur Modellsprachenqualität erlaubt Rückschlüsse über die Eignung von Modellsprachen in einem gegebenen Kontext. Eine empirische Studie dazu findet sich in NK05. Mit *QoMo* (vHPv07) existiert ferner eine auf SEQUAL basierende Erweiterung, die den Prozess des Modellierens stärker ins Zentrum rückt. Durch die Strukturierung von Modell- und Modellierzielen und die Ableitung von Strategien zu deren Erreichung wird der Dynamik, die einem Modelliervorhaben innewohnt, besser Rechnung getragen.

2.2.1.4 Eignungskriterien nach Zelewski

Zelewski stellt in Zel96 Kriterien auf, um die Eignung von Petri-Netzen zur Modellierung komplexer Realsysteme, insbesondere Produktions- und Logistiksysteme sowie Unternehmensverwaltungen, zu beurteilen. Er unterscheidet dabei zwischen dem Aspekt der *Modellierungsfähigkeit* und dem der *Modellierungsgüte*. Ersterer beleuchtet, welche Sachverhalte ein Modellierungskonzept grundsätzlich auszudrücken vermag. Zel96 unterscheidet dabei zwischen der allgemeinen und der speziellen, auf eine bestimmte Domäne beschränkten Modellierungsfähigkeit. Letzterer, der Aspekt der Modellierungsgüte, dagegen untersucht die qualitative Beschaffenheit der Repräsentation eines modellierten Sachverhalts.

Aus Sicht der Wirtschaftsinformatik kann die Modellierungsfähigkeit auf die Gebiete des Prozess-Managements und der informationstechnischen Implementierung eingeschränkt werden, Zel96 greift exemplarisch die Koordinierung von Produktionsprozessen in komplexen Produktionssystemen heraus und begründet die Wahl mit der grossen Bedeutung in der Praxis, der recht grossen Anzahl vorhandener Beiträge zur Petri-Netz-gestützten Modellierung der Prozesskoordinierung in komplexen Produktionssystemen und mit der Übertragbarkeit zahlreicher Aspekte der Prozesskoordinierung von Produktionssystemen auf andere betriebliche Realsysteme, wie etwa Unternehmensverwaltungen. Gemäss Zel96 erstrecken sich die speziellen Fähigkeiten, die von einem Modellierungskonzept für die Koordinierung von Prozessen in komplexen Produktionssystemen erwartet werden, u. a. auf die Repräsentation folgender Sachverhalte:

- grundlegende Strukturelemente zur Ablauflogik, wie sequentielle, nebenläufige Abfolgen, Entscheidungsalternativen, Synchronisierung;
- betriebswirtschaftliche Aspekte, wie Ressourcen-, Instandhaltungs- und Zeitbedarf, Zielsetzung;
- unvorhersehbare Ereignisse;
- bewegliche Objekte zur Modellierung des Werkstück- und Werkzeugflusses;
- individualisierte, eindeutig zuordbare Objekte;
- Priorisierungen;
- Integrationsfähigkeit des Modells.

Zur Beurteilung der Modellierungsgüte schlägt Zel96 nebst einer anwendungsgerechten Modellierfähigkeit als institutionellem Kriterium die instrumentellen Kriterien *Konstruktivität*, *Analysierbarkeit* und *Adaptivität* vor.

Unter der Konstruktivität versteht Zel96 den Unterstützungsumfang zur systematischen Konstruktion von Modellen mittels Entwurfshilfen im weitesten Sinne. Dies sind in erster Linie die Unterstützung der Modularität und der hierarchischen Modellverfeinerung. Die Modularität ermöglicht die Reduktion der Komplexität eines Objekts, indem dieses in einfachere, besser

überschaubare, in sich abgeschlossene Teilobjekte zerlegt wird, welche anschliessend systematisch untereinander verknüpft werden (Bottom-up-Prinzip). Die hierarchische Modellverfeinerung geht den umgekehrten Weg; mittels Top-down-Prinzip wird das Modell ausgehend von der Gesamtsicht laufend verfeinert.

Die Analysierbarkeit betrifft die Vielfalt der Auswertungsmöglichkeiten sowohl bzgl. der Modelleigenschaften wie der zur Verfügung stehenden Analysetechniken. Bei den Modelleigenschaften weist Zel96 darauf hin, dass nicht alle Eigenschaften, die untersucht werden können, von praktischem Interesse sind, und weist damit auf den relativen Charakter dieses Kriteriums hin. Eine grosse Anzahl zur Verfügung stehender Analysetechniken ist dann hilfreich, wenn sie sich bzgl. Erkenntnispotential und Ressourceneinsatz voneinander unterscheiden. Umgekehrt kann die Mächtigkeit einer Analysetechnik wichtiger sein, wenn möglichst viele Eigenschaften mit nur einer Technik untersucht werden sollen.

Die Adaptivität schliesslich beschreibt Zel96 als die Möglichkeit, ein Modell an unterschiedliche Modellierungsbedürfnisse anzupassen. Das betrifft die Option, Teile des Modells nachträglich zu verfeinern oder zu vergröbern (vertikale Achse) sowie Teile des Modells im Sinne der Modellierungsbedürfnisse wegzulassen oder neue Teile zum bestehenden Modell hinzuzufügen (horizontale Achse).

Weitere von Zel96 vorgeschlagene Aspekte im Sinne der Modellierungsgüte sind die Anwendungseffizienz bei der Konstruktion und bei der Analyse eines Modells, die Benutzerfreundlichkeit sowie die Implementierbarkeit im Sinne der interaktiven Unterstützung durch die Anwendung von Informationstechnologie.

2.2.1.5 Anforderungen nach Frank und van Laak

FB03 unterscheiden *formale*, *anwenderbezogene* und *anwendungsbezogene* Anforderungen an Modelliersprachen, die nicht immer voneinander unabhängig sind. Zu den formalen Anforderungen zählen sie die *Korrektheit und Vollständigkeit*, die *Einheitlichkeit*[8] *und Redundanzfreiheit* und die *Wiederverwendbarkeit und Wartbarkeit durch Abstraktion*. Als anwenderbezogene Anforderungen füh-

[8] Die Einheitlichkeit wird von den Autoren auch als Klarheit bezeichnet („Eine Sprachspezifikation sollte auch dem Prinzip der Einheitlichkeit bzw. Klarheit genügen.“).

ren die Autoren die *Einfachheit* und die *Verständlichkeit und Anschaulichkeit*, als anwendungsbezogene die *Mächtigkeit und Angemessenheit* und die *Operationalisierbarkeit* auf.

Im Hinblick auf die Modellierung von Geschäftsprozessen kommen gemäss FB03 weitere Anforderungen hinzu. Aufgeführt werden *Abstraktionsebenen*[9] zur Darstellung der verschiedenen Modellebenen vom Metamodell bis zur Instanzebene, *Flexibilität/Anpassbarkeit* bspw. in Form frei definierbarer Typisierungen und die *Unterstützung von Sichten*, um einzelne Komponenten wie Ressourcen, Kontrollstrukturen, Organisationseinheiten u. a. m. gesondert betrachten zu können.

Ferner gehen FB03 auf betriebswirtschaftliche und organisatorische Konzepte, namentlich den Prozess, das Ereignis, die Ziele von Geschäftsprozessen, die Ressourcen, den Prozessgegenstand, die Modellierung der Organisationsstruktur, die Konzepte zur Modellierung von Rollen, die Konzepte zur Modellierung unternehmensübergreifender Geschäftsprozesse sowie die Berücksichtigung weiterer betriebswirtschaftlicher Aspekte, ein und erwähnen als elementare Kontrollstrukturen die Sequenz, Bedingungen und Regeln, die alternative Ausführung, die Nebenläufigkeit und die Parallelität als Sonderfall der Nebenläufigkeit, die Transaktion, die Iteration, die Synchronisation und die unspezifizierte Reihenfolge. Ausnahmen sollen wie Ereignisse beschrieben werden, deren Behandlung in einem eigenen Prozessmodell abgebildet sein kann. Ausnahmen können unter anderem durch die Verletzung von Integritätsbedingungen, wie Kardinalitäten, Vor-, Nachbedingungen und Prozesstypinvarianten, ausgelöst werden.

Erwähnt wird in FB03 auch die Verbindung zwischen Geschäftsprozess und Implementierung; insbes. werden Fragen zur Persistenz der Daten, zur Bildung von Objekten und zur Umsetzung in ein Workflow-System aufgeworfen. Weitere Punkte betreffen die Möglichkeit, individuelle Anpassungen am Erscheinungsbild, weniger an der Syntax und Semantik vornehmen zu können, die an unterschiedliche Benutzergruppen angepasste Dokumentation sowie die exakte Spezifikation von Syntax und Semantik als Metamodell oder Grammatik. Abgeschlossen wird der Anforderungskatalog mit dem Hinweis auf die Notwendigkeit eines Vorgehensmodells und den Aufbau einer Projektinfrastruktur.

[9] Mit Abstraktion ist hier die Konkretisierung, nicht die Verkürzung gemeint

2.2.2 Sichtenspezifische Kriterien

2.2.2.1 Kriterien in der Datenmodellierung

Aus der Datenmodellierung stammen die Anforderungen von BCN91 und von MS94.

Anforderungen an Datenmodelle von Batini et al.: BCN91 schlagen als Kriterium an ein Datenmodell die acht Kriterien der *Vollständigkeit*, der *Richtigkeit*, der *Minimalität*, der *Ausdruckskraft*, der *Lesbarkeit*, der *Selbsterklärung*, der *Erweiterbarkeit* und der *Normalität* (Completeness, Correctness, Minimality, Expressiveness, Readability, Self-explanation, Extensibility, Normality) vor. Die *Vollständigkeit* besagt, dass die relevanten Merkmale im Modell enthalten sind. Die *Richtigkeit* teilt sich wie bei den GoM in eine syntaktische und eine semantische Komponente auf. Die syntaktische Richtigkeit fordert das Einhalten des Metamodells. Die semantische Richtigkeit verlangt, dass die zum Einsatz kommenden Modellkonstrukte gemäss ihrer Definition eingesetzt werden. Die *Minimalität* ist dann erfüllt, wenn das Modell beim Weglassen einer Komponente einen Informationsverlust erleidet. Die *Ausdruckskraft* ist dann erfüllt, wenn das Objektsystem auf natürliche Weise und möglichst kompakt modelliert ist. Die *Lesbarkeit* wird durch eine Anordnung der Symbole, die bspw. die Anzahl Kantenüberscneidungen minimiert, erreicht. Die *Selbsterklärung* verlangt die Beschreibung von Sachverhalten ohne die Hinzunahme von fremden Beschreibungstechniken. Die *Erweiterbarkeit* fordert die Möglichkeit der Zerlegung eines Modells, um es an geänderte Anforderungen anpassen zu können. Die *Normalität* schliesslich ist ein datenmodellspezifisches Kriterium, welches im relationalen Datenmodell unerwünschte Abhängigkeiten zu minimieren versucht.

Rahmenwerk zur Datenqualität von Moody und Shanks: MS94 stellen ein Rahmenwerk zur Qualitätssicherung vor, welches sich aus gewichteten Merkmalen zur Datenqualität zusammensetzt, an denen verschiedene Modellbeteiligte unterschiedlich stark interessiert sind. Hinzu kommen eine Metrik zur Messung und eine Strategie zur Verbesserung der Qualität. Das Rahmenwerk wurde mehrfach empirisch untersucht und überarbeitet. In MS03 sind acht Qualitätsmerkmale und vier Beteiligte aufgeführt: Der Akteur *Business*

User ist in erster Linie an den Merkmalen *Vollständigkeit*, *Integrität*, *Flexibilität* und *Verständlichkeit* interessiert. Für den *Data Analyst* primär wichtig sind die Merkmale *Korrektheit* und *Einfachheit*, für den *Data Administrator* steht das Merkmal *Integration* und für den *Application Developer* das Merkmal *Implementierbarkeit* im Vordergrund.

2.2.2.2 Workflow-Pattern

Zur Beurteilung der Ausdrucksmächtigkeit von Workflowmanagement-Systemen präsentieren vBtK00 in Anlehnung an die Design Pattern von GHJV95 die *Workflow Pattern*. Dabei handelt es sich um Kontrollstrukturbausteine zur Beschreibung der statischen Komponenten eines Workflow, wie sie in der Praxis beobachtet wurden. Zu den ursrprünglich 20 in vtKB03 beschriebenen Workflow Control Pattern sind weitere 23 hinzugekommen (RtvM06). Ebenfalls wurden in RtEv04a und RtEv04b weitere Pattern zu Ressourcen und Daten beschrieben. Mit Hilfe der Workflow Pattern sollen die beim Modellieren von Geschäftsprozessen stets wiederkehrenden Anforderungen beschrieben werden (RtvM06).

Die Darstellung der Workflow-Pattern folgt einem für alle Pattern gleichbleibenden Aufbau und umfasst nebst einer einleitenden Beschreibung Beispiele, Motivation, Kontext, Implementierung, mögliche Stolpersteine, Hinweise zu deren Vermeidung und Evaluationskriterien zur Unterstützung bei der Produktewahl. Die Kontrollfluss-Pattern sind als Coloured Petri Net dargestellt[10].

Anwendungen von Workflow Control Pattern finden sich in Untersuchungen von Workflow-Management-Systemen und Modelliersprachen (vtKB03, RtvM06, RvtW06, WvD$^+$06).

2.2.3 Bewertungsraster

Die Herleitung des für die Evaluation zugrunde gelegten Bewertungsrasters folgt diesem Vorgehen:

[10] Frühere Arbeiten zu Workflow Pattern benutzten zur Darstellung Petri-Netze mit informalen Erweiterungen. Im Ggs. dazu erlauben die Coloured Petri Net (s. 4.3) nicht nur eine einheitliche und eindeutige Aussage, sondern bieten direkte Unterstützung bei der Implementierung.

Kriterium	Anzahl Nennungen	GoM (2.2.1.1)	Ontologie (2.2.1.2)	SEQUAL (2.2.1.3)	Zelewski (2.2.1.4)	Frank, van Laak (2.2.1.5)	Batini et al. (2.2.2.1)	Moody, Shanks (2.2.2.1)	WF-Pattern (2.2.2.2)
Vollständigkeit	7		x	x	x	x	x	x	x 3)
Klarheit / Verständl.	5	x	x	x		x 1)		x	
Richtigk. / Korrektheit	5	x		x		x	x	x	
Abstraktionsfähigkeit	3	x			x 6)	x			
Angem. / Rel. / Zielerr.	3	x		x		x			
Implementierbarkeit	3				x	x		x	
Wirtschaftlichkeit	3	x			x 4)	x 5)			
Einfachheit	2					x		x	
Integrationsfähigkeit	2				x			x	
Verfügbark. / Persist.	2			x		x			
Vergleichbarkeit	2	x	x 2)						

Tabelle 2.2: *Meistgenannte Kriterien, alphabetisch geordnet nach Anzahl ihrer Nennungen. 1) Einheitlichkeit gemäss Autoren synonym zu Klarheit. 2) Die Vergleichbarkeit ist implizit im Ontologieansatz enthalten. 3) Die Vollständigkeit ist in der Forderung des Ansatzes, alle in der Praxis auftretenden Fälle abzudecken, implizit enthalten. 4) Die Anwendungseffizienz wird hier als Beitrag zur Wirtschaftlichkeit gedeutet. 5) Gemäss Autoren bedeutet die Vermeidung von Redundanz durch Sichten höhere Wirtschaftlichkeit. 6) Die Forderung des Autors nach Modularität und Verfeinerung wird als Forderung nach Abstraktionsfähigkeit gedeutet.*

1. Gleichlautende oder gleichbedeutende Kriterien aus 2.2.1.1 bis 2.2.2.2 mit mehr als einer Nennung werden in einem ersten Schritt tabellarisch angeordnet (s. Tab. 2.2). Die Reihenfolge entspricht ihrer Anzahl Nennungen; bei gleichvielen Nennungen erfolgt eine alphabetische Anordnung.

2. Kriterien mit nur einer Nennung, die mit den Kriterien von Tab. 2.2 eng verwandt sind, werden im zweiten Schritt diesen zugeordnet (s. Tab. 2.3). Die Anordnung folgt der Anordnung von Tab. 2.2.

3. In einem dritten Schritt werden die Kriterien aus Tab. 2.3 konsolidiert und mit einer neuen Benennung versehen (s. Tab. 2.4). Die Reihenfolge der umbenannten Kriterien erfolgt alphabetisch, um zum Ausdruck zu bringen, dass sie als gleichbedeutend gelten sollen.

Im Detail wurde folgendermassen verfahren:

Der Blick der Workflow-Pattern auf die Darstellungsmöglichkeit aller in der Praxis beobachteten Kontrollfluss-, Ressourcen- und Datenkonstrukte und die beispielhafte Aufzählung von Zelewski aller zu repräsentierenden Sachverhalte wird jeweils als eine Forderung der Vollständigkeit verstanden, und die paarweise auftretenden Kriterien von Frank und van Laak wurden getrennt behandelt.

Ferner wurden die Kriterien Klarheit und Verständlichkeit, Angemessenheit und Relevanz und Zielerreichung, Richtigkeit und Korrektheit, Verfügbarkeit und Persistenz auf Grund ihrer Ähnlichkeit jeweils zusammengefasst. Das Kriterium Abstraktionsfähigkeit subsumiert die Kriterien systematischer Aufbau, Sichten und Konstruktivität, welche alle dasselbe Ziel verfolgen, und beinhaltet damit die Dekomposition als Hilfsmittel.

Damit bleiben 18 Kriterien übrig, welche noch nicht zugeteilt wurden. Diese Kriterien sind mit den 11 Kriterien aus Tab. 2.2 eng verwandt und werden diesen wie folgt zugeordnet (s. Tab. 2.3):

Das Kriterium der Interpretation aus dem SEQUAL-Rahmenwerk (2.2.1.3), das Kriterium der Benutzerfreundlichkeit von Zelewski (2.2.1.4), die Kriterien der Lesbarkeit und der Selbsterklärung von Batini et al. (2.2.2.1) sind Voraussetzung für oder werden unterstützt durch das Kriterium der Klarheit/Verständlichkeit. Im SEQUAL-Rahmenwerk wird die Verständlichkeit bereits genannt und wird hier nicht doppelt gerechnet. Die beiden letztgenannten, Lesbarkeit und Selbsterklärung, werden von denselben Autoren genannt und liegen nahe beieinander, sie werden daher als ein Kriterium gezählt, so dass Klarheit/Verständlichkeit neu mit 7 Nennungen gezählt wird.

Das Kriterium der Integrität von Moody, Shanks (2.2.2.1) kann dem Kriterium Richtigkeit/Korrektheit hinzugerechnet werden, doch wird dieses bereits erwähnt.

Kriterium	Anzahl Nennungen	GoM (2.2.1.1)	Ontologie (2.2.1.2)	SEQUAL (2.2.1.3)	Zelewski (2.2.1.4)	Frank, van Laak (2.2.1.5)	Batini et al. (2.2.2.1)	Moody, Shanks (2.2.2.1)	WF-Pattern (2.2.2.2)
Klarheit/Verständl.	7	x	x	x	x	x	x	x	
Vollständigkeit	7		x	x	x	x	x	x	x
Angem./Rel./Zielerr.	5	x		x	x	x	x		
Richtigkeit/Korrektheit	5	x		x		x	x	x	
Wirtschaftlichkeit	5	x			x	x	x	x	
Abstraktionsfähigkeit	3	x			x	x			
Implementierbarkeit	3				x	x		x	
Einfachheit	2					x		x	
Integrationsfähigkeit	2				x			x	
Verfügbark./Persistenz	2			x		x			
Vergleichbarkeit	2	x	x						

Tabelle 2.3: *Verbleibende Kriterien, die eines der 11 Kriterien aus 2.2 unterstützen oder einem davon ähnlich sind, werden diesen zugerechnet. Dies führt zur hier abgebildeten Tabelle.*

Das Kriterium der Adaptivität von Zelewski (2.2.1.4) ist ein Sonderfall des Kriteriums der Abstraktionsfähigkeit, indem es nebst der Vergröberung/Verfeinerung auch das Hinzufügen/Weglassen fordert, und wird der Abstraktionsfähigkeit zugerechnet.

Das Kriterium der Analysierbarkeit von Zelewski (2.2.1.4) sowie die beiden Kriterien Minimalität und Ausdruckskraft (2.2.2.1) von Batini et al., welche nahe beieinander liegen, werden zum Kriterium Angemessenheit/Relevanz/Zielerreichung zugerechnet, so dass dieses neu 5 statt 3 Nennungen aufweist.

Die Kriterien der Redundanzfreiheit und der Wiederverwendbarkeit und Wartbarkeit von Frank, van Laak (2.2.1.5) unterstützen zwar das Kriterium

der Wirtschaftlichkeit, doch wurde bereits deren Kriterium der Sichten, das ebenfalls Redundanz vermeiden hilft, berücksichtigt. Hingegen werden das Kriterium der Erweiterbarkeit von Batini et al. (2.2.2.1) sowie das Kriterium der Flexibilität von Moody, Shanks (2.2.2.1) der Wirtschaftlichkeit zugezählt, so dass dieses Kriterium nun 5 Nennungen aufweist.

Die Forderung der Normalität von Batini et al. (2.2.2.1) ist spezifisch auf die Datenmodellierung zugeschnitten und wird an dieser Stelle fallengelassen[11]. Ebenfalls nicht hinzugerechnet werden die Forderungen von Frank, van Laak (2.2.1.5) nach anwendergerechter Dokumentation, dieser Punkt wird als durch die von denselben Autoren angeführten Kriterien der Klarheit und Verständlichkeit abgedeckt verstanden.

2.2.4 Zusammenfassung mit Bewertungshinweisen

Am häufigsten als Kriterien genannt werden die Klarheit/Verständlichkeit und die Vollständigkeit; von einem Modell wird erwartet, dass die relevanten Aspekte abgebildet sind und vom Anwenderkreis interpretiert werden können. Ebenfalls häufig genannt werden die Angemessenheit/Relevanz/Zielerreichung, die Richtigkeit/Korrektheit und die Wirtschaftlichkeit; ihre Forderungen unterstützen das zielgerichtete Arbeiten. Die Abstraktionsfähigkeit richtet den Blick des Modellanwenders auf die relevanten Sachverhalte und unterstützt damit die Klarheit/Verständlichkeit und die Angemessenheit/Relevanz/Zielerreichung. Mit der Implementierbarkeit reduziert sich der Aufwand für die informationstechnische Umsetzung.

Das Kriterium der Einfachheit ist u. a. deshalb problematisch, weil es keine eindeutige Definition gibt. Einerseits scheint etwas dann als einfach bezeichnet zu werden, wenn es aus wenigen Elementen und Spielregeln besteht[12], andererseits kann Einfachheit als vom Grad der Erfahrung und Ausbildung abhängige Eigenschaft aufgefasst werden, womit sich die Frage stellt, inwieweit das Modell verantwortlich gemacht werden soll. Hinzu kommt, dass Einfach-

[11] wenngleich sich diese Forderung als Forderung nach Redundanzfreiheit interpretieren liesse

[12] Dies widerspiegelt sich in der folgenden, einzig auffindbaren Definition: „Einfachheit, oder auch Primitivität, ist ein Zustand, der sich dadurch auszeichnet, dass nur wenige Faktoren zu seinem Entstehen oder Bestehen beitragen, und dadurch, dass das Zusammenspiel dieser Faktoren durch nur wenige Regeln beschrieben werden kann. Damit ist Einfachheit das Gegenteil von Komplexität“ (Wikipedia).

heit im Widerspruch zu den gesteckten Modellzielen stehen kann. Daher wird von diesem Kriterium Abstand genommen, vielmehr soll die *mögliche* Einfachheit gefordert werden, im Sinne der Unterstützung des Kriteriums Klarheit/Verständlichkeit.

Die Integrationsfähigkeit und die Vergleichbarkeit beeinflussen die Effizienz des Prozess-Managements und die Akzeptanz bei den Anwendern, während das technisch/organisatorische Kriterium der Verfügbarkeit/Persistenz als selbstverständlich vorausgesetzt werden soll und vom Modell nur insofern beeinflusst werden kann, als dass dessen Implementierbarkeit gegeben ist; aus diesem Grund wird die Verfügbarkeit/Persistenz als Kriterium fallengelassen.

Damit verbleiben neun Kriterien, welche in Tab. 2.4 aufgeführt sind und teilweise neu benannt werden. Diese Kriterien gelten als gleich wichtig und werden im Rahmen des Design-Science-Ansatzes zur Bewertung des in Kap. 5 vorgestellten Artefakts herangezogen.

Als letzter Schritt verbleibt zu klären, welche Kriterien an eine Sprache zur Modellerstellung zu stellen sind. Um zu beurteilen, ob eine Modelliersprache das Erstellen eines den obigen Kriterien genügenden Modells unterstützt, wird im folgenden als Vorschlag zu jedem der neun Kriterien ein Hinweis gegeben, wie das jeweilige Kriterium erfüllt werden kann.

2.2.4.1 K 1 Abstraktionsfähigkeit

Das Kriterium der Abstraktionsfähigkeit soll dann als erfüllt gelten, wenn die Modellsprache die vertikale und die horizontale Gliederung eines Modells unterstützt. Mit der vertikalen Gliederung ist es möglich, Ausschnitte der Diskurswelt separat zu modellieren und später zu einem Ganzen zusammenzufügen. Beispiele für Ausschnitte in der Geschäftsprozessmodellierung sind einzelne Verrichtungen, Geschäftsfelder oder Abteilungen. Die horizontale Gliederung erlaubt die Verfeinerung von Prozessen über Subprozesse in Arbeitsschritte und umgekehrt.

2.2.4.2 K 2 Implementierbarkeit

Die Implementierbarkeit sei dann gegeben, wenn eine direkte Abbildung der mit der Modellsprache erzeugten Modelle in ein Informationssystem vorge-

nommen werden kann. Voraussetzung dazu ist eine unmissverständliche Beschreibung von Syntax und Semantik in Form einer Formalisierung.

2.2.4.3 K 3 Integrationsfähigkeit

Ein Modell soll dann als integrationsfähig gelten, wenn die Sprache Konstrukte bereitstellt, mit denen eine Verknüpfung mit weiteren Modellen hergestellt werden kann. Sind die anzubindenden Modelle nicht in derselben Sprache verfasst, und stellen diese ihrerseits Konstrukte zur Anbindung von anderssprachigen Modellen zur Verfügung, gilt die Integrationsfähigkeit dann als erfüllt, wenn die notwendigen Transformationsregeln zwischen den Sprachen definiert werden können.

2.2.4.4 K 4 Korrektheit

Damit die Korrektheit eines Modells erreicht werden kann, soll die Vollständigkeit vorausgesetzt werden, das Kriterium ist also nicht unabhängig. Nur wenn die Modellsprache sämtliche notwendigen Konstrukte zur Abbildung der Diskurswelt bereitstellt, kann ein korrektes Modell resultieren. Wenn die Vollständigkeit gegeben ist, liegt es am Modellierer, die formalen Regeln der Modellsprache zu beachten, um ein korrektes Modell herzuleiten.

2.2.4.5 K 5 Lesbarkeit

Die Lesbarkeit wird in aller Regel durch möglichst wenige, aussagekräftige Symbole erleichtert. Eine geschickte Plazierung der graphischen Elemente sowie das Bereitstellen von Techniken zur Abstraktion sind weitere Elemente, die die Lesbarkeit unterstützen können. Für die abschliessende Bewertung der Lesbarkeit eines Modells sind umfangreiche empirische Studien angezeigt.

2.2.4.6 K 6 Minimalität

Wenn es kein Modellsprachenelement gibt, das weggelassen werden kann, ohne die Vollständigkeit zu verletzen, gelte die Minimalität als erfüllt.

Abstraktionsfähigkeit	Abstraktionsfähigkeit	K 1
Implementierbarkeit	Implementierbarkeit	K 2
Integrationsfähigkeit	Integrationsfähigkeit	K 3
Richtigkeit/Korrektheit	Korrektheit	K 4
Klarheit/Verständlichkeit	Lesbarkeit	K 5
Angemessenheit/Relevanz/Zielerreichung	Minimalität	K 6
Vergleichbarkeit	Uniformität	K 7
Vollständigkeit	Vollständigkeit	K 8
Wirtschaftlichkeit	Wirtschaftlichkeit	K 9

Tabelle 2.4: *Die verbleibenden neun Kriterien (links) und deren Neubenennung (rechts) für die weitere Verwendung als Evaluationskriterien. Die Kriterien sollen als gleichwertig gelten, daher sind sie alphabetisch angeordnet.*

2.2.4.7 K 7 Uniformität

Das Kriterium der Uniformität soll dann als erfüllt angesehen werden, wenn Konstrukte mit gleicher Semantik gleiches Aussehen haben und direkt untereinander vergleichbar sind. Umgekehrt sollen Konstrukte unterschiedlicher Bedeutung ein unterschiedliches Aussehen haben.

2.2.4.8 K 8 Vollständigkeit

Wenn sich sämtliche für den Modellzweck relevanten Sachverhalte mit der Modellsprache abbilden lassen, gelte die Vollständigkeit als erfüllt. Die Abbildung relevanter Sachverhalte kann durch Erweiterungen, die die Modellsprache vorsieht, geschehen.

2.2.4.9 K 9 Wirtschaftlichkeit

Die Wirtschaftlichkeit ist ein schwierig einzuschätzendes Kriterium, da zwar der Aufwand leicht errechnet werden kann, nicht jedoch der Nutzen, dieser lässt sich in den meisten Fällen nur grob schätzen. Daher sei an Stelle der Wirtschaftlichkeit die Aufwandminimierung in Betracht zu ziehen. Diese kann dadurch unterstützt werden, dass die Kriterien Abstraktionsfähigkeit,

Implementierbarkeit, Integrationsfähigkeit, Korrektheit, Lesbarkeit und Uniformität erfüllt werden.

Kapitel 3

Geschäftsprozessmodellierung in der Praxis

Das Angebot an Notationen zur Geschäftsprozessmodellierung ist gross. Unterscheidungskriterien lassen sich auf Grund des zu modellierenden Gegenstandes, der Zielsetzung der Modellierung oder den Ansprüchen des Modellierers ableiten[1]. Die Auswahl der zwei im folgenden vorgestellten Geschäftsprozessmodelle erfolgte auf Grund ihrer Popularität (vgl. auch MN03) in deren jeweiligem primären Einsatzgebiet Betriebswirtschaft/Wirtschaftsinformatik und Informatik/Wirtschaftsinformatik.

Die Ereignisgesteuerten Prozessketten stehen als Vertreter für die betrieblich-organisatorische und wirtschaftsinformatische Sicht, während UML in erster Linie die Sicht des Software-Architekten vertritt.

3.1 Ereignisgesteuerte Prozessketten

Ereignisgesteuerte Prozessketten wurden am Institut für Wirtschaftsinformatik der Universität des Saarlandes entwickelt (KNS92). Sie setzen sich aus den

[1] Dies wirkt sich bei einigen Prozessmodellen durch die Bereitstellung vielfältiger Varianten aus: Alleine die Modellfamilie IDEF (s. http://www.idef.com/) umfasst 15 Methoden, jeweils auf ein spezifisches Teilgebiet ausgerichtet. Und die in 3.2 vorgestellte UML stellt aktuell 16 Diagramme zur Verfügung, mit denen Klassen, Objekte, Anwendungsfälle, Komponenten, Zustände, Aktivitäten, Sequenzen u. w. beschrieben werden können.

Grundelementen *Ereignis*, *Funktion*, *Kontrollflusskante* und *Verknüpfungsoperator* zusammen und sind Bestandteil der *Architektur integrierter Informationssysteme ARIS*.

3.1.1 Fachkonzepte

ARIS besteht aus den drei Sichten *Organisation*, *Daten* und *Funktionen*, untereinander mit der Sicht auf die *Steuerung* verbunden. Die jeweiligen Bestandteile sind in den Abstraktionsebenen *Fachkonzept*, *DV-Konzept* und *Implementierung* beschrieben (s. Abb. 3.1). Während im Fachkonzept die Sachverhalte aus einer betriebswirtschaftlichen Perspektive beschrieben werden, sind sie im DV-Konzept auf die Belange der Datenverarbeitung zugeschnitten. Die Konkretisierung erfolgt in der Implementierungsebene (vgl. Sch92).

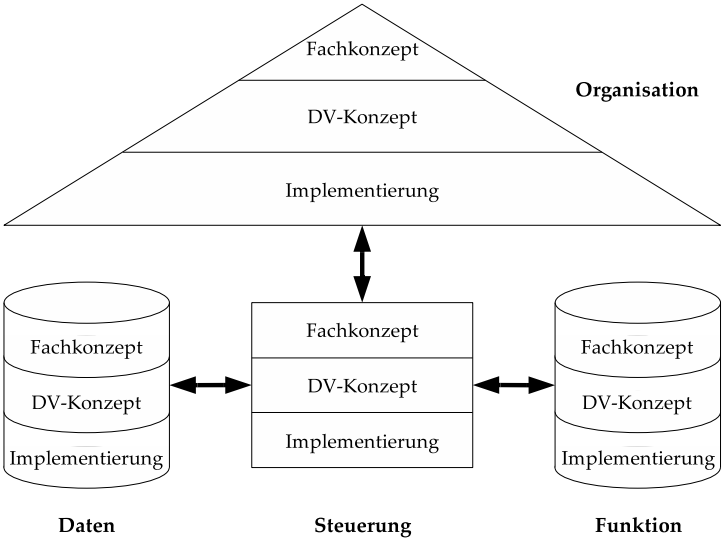


Abbildung 3.1: *Das ARIS-Konzept nach Sch98a.*

3.1.1.1 Steuerungssicht

Das Fachkonzept der Steuerungssicht in ARIS wird durch die Ereignisgesteuerten Prozessketten beschrieben. Im Zentrum von Prozessen steht die Funktion, die in den folgenden Ausführungen kurz vorgestellt wird, um sie anschliessend mit den Sichten der Organisation und der Daten in Verbindung zu setzen. Die Ausführungen beschränken sich auf die zum Verständnis der Ereignisgesteuerten Prozessketten notwendigen Elemente; eine weitergehende Beschreibung des ARIS-Hauses findet sich u. v. a. in Sch92, KNS92.

3.1.1.2 Funktionssicht

Eine Funktion in ARIS kennzeichnet einen Vorgang, der Objekte generiert oder verändert. Gezeichnet wird eine Funktion als ein mit der Funktionsbezeichnung beschriftetes Rechteck mit abgerundeten Ecken. Zur Reduktion der Komplexität lassen sich Funktionen in hierarchisch gegliederte Funktionsbäume zerlegen (vgl. Sch98a). Bäume als zweidimensional verkettete Strukturen sind leicht verständlich und finden vielerorten Anwendung. Abb. 3.2 zeigt als Beispiel die Zerlegung der Funktion *Auftragsabwicklung* in einen Funktionsbaum. Die Wurzel des Baumes, der Knoten ganz oben, entspricht der höchsten Abstraktionsebene, während die Blätter[2] den Funktionen, deren weitere Zerlegung betriebswirtschaftlich keinen Sinn mehr macht, entsprechen. Diese Funktionen, die i. d. R. aus nur einem Arbeitsschritt bestehen, werden Elementarfunktionen genannt.

Aus dem Funktionsbaum geht nicht hervor, in welcher zeitlich-logischen Abfolge die Funktionen ausgeführt werden. Durch Anordnungsbeziehungen kann diese fehlende Ablaufsicht mittels gerichteter Kanten dargestellt werden. Zusätzlich können durch eine Wahrscheinlichkeitsangabe Verzweigungen und durch Rücksprünge Schleifen gebildet werden. Ein Beispiel ist in Abb. 3.3 wiedergegeben (aus Sch98a).

3.1.1.3 Datensicht

Zu den Daten werden Informationsobjekte und Datenelemente gezählt. Beispiele sind Arbeitspläne (Informationsobjekte), aufgrund derer benötigte Res-

[2] auch Endknoten genannt

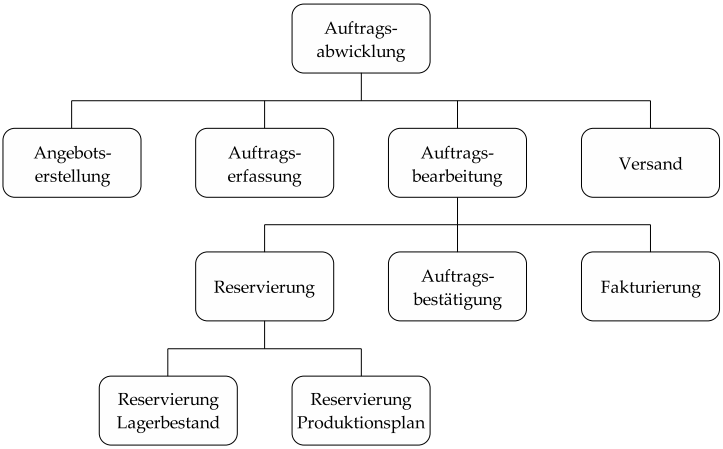
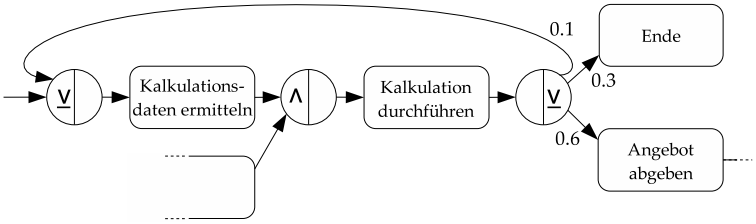


Abbildung 3.2: *Beispiel eines Funktionsbaumes nach Sch98a.*



Legende: ∧ „und“ ∨ „oder“ ⩒ „exklusiv oder“

Abbildung 3.3: *Beispiel einer Ablauffolge mit angegebenen Verzweigungswahrscheinlichkeiten nach Sch98a.*

sourcen (Datenelemente) berechnet werden können, oder berechnete Kosten und Lieferzeiten (Datenelemente), mit denen ein Angebot (Informationsobjekt) berechnet wird (vgl. Bro91).

Ereignisse, die nebst den Funktionen im Zentrum der Ereignisgesteuerten Prozessketten stehen, werden ebenfalls in der Datensicht erfasst: „Ereignisse lösen Funktionen aus und sind deren Ergebnis“ (Sch98a). Ereignisse verbrauchen im Gegensatz zu Funktionen keine Zeit, sondern treten unmittelbar ein. Ereignisse werden als mit der Ereignisbezeichnung beschriftete Sechsecke gezeichnet.

Treffen mehrere Ereignisse auf eine Funktion oder umgekehrt, werden diese durch logische Operatoren vom Typ *und*, *oder* und *exklusiv oder* verknüpft. Abb. 3.4 bis 3.10 nach Sch98a zeigen die Regeln.

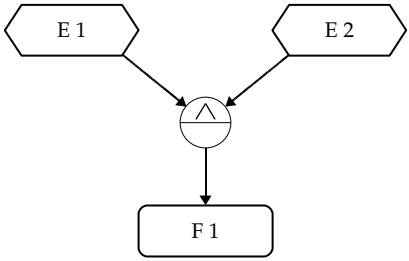


Abbildung 3.4: *„Wenn die Ereignisse E 1 und E 2 eingetreten sind, startet Funktion F 1“ (Sch98a).*

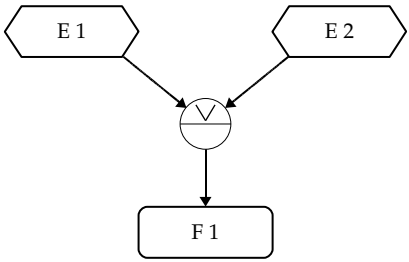


Abbildung 3.5: *„Wenn Ereignis E 1 oder E 2 eingetreten ist, startet Funktion F 1“ (Sch98a).*

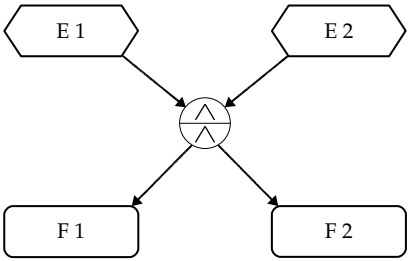


Abbildung 3.6: *„Wenn die Ereignisse E 1 und E 2 eingetreten sind, starten die Funktionen F 1 und F 2“ (Sch98a).*

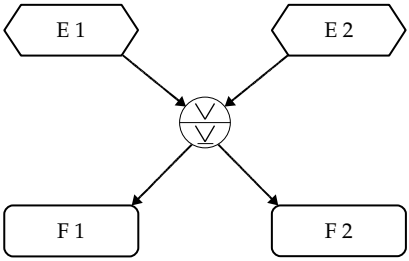


Abbildung 3.7: *„Wenn das Ereignis E 1 oder E 2 eingetreten ist, startet entweder Funktion F 1 oder F 2“ (Sch98a). (Auch wenn Scheer kein Beispiel mit einem nicht-ausschliessenden Oder aufführt, wäre ein solches regelkonform, was bei Verzweigungen zu Schwierigkeiten bei der Semantik und bei der Translation in ein anderes Modell führen kann.)*

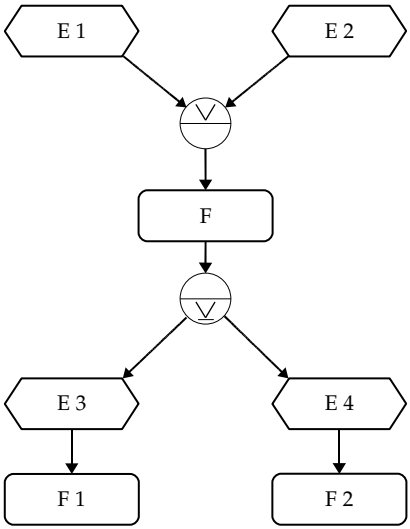


Abbildung 3.8: *„Wenn Ereignis E 1 oder E 2 eingetreten ist, startet die Entscheidungsfunktion F, in der entschieden wird, ob entweder Ereignis E 3 oder E 4 eintritt" (Sch98a).*

Die Operatoren ihrerseits unterscheiden zwischen Eingang und Ausgang. Unterschiedliche Operatoren können untereinander direkt verbunden werden (s. Abb. 3.9). Zusätzlich lassen sich bei komplexeren Beziehungen zwischen den abgeschlossenen und startenden Funktionen Entscheidungstabellen für Ein- und Ausgänge hinterlegen, darauf wird hier nicht näher eingegangen.

3.1.1.4 Organisationssicht

Beim Verbinden der Funktions- mit der Organisationssicht wird festgelegt, welche Organisationseinheit für welche Funktionsschritte einer Prozesskette zuständig ist. Zusätzlich kann in einer detaillierteren Betrachtung angegeben werden, ob die Funktionen interaktiv oder automatisch abgearbeitet werden. Dies führt zur ablauforganisatorischen Spezifizierung der Prozesskette.

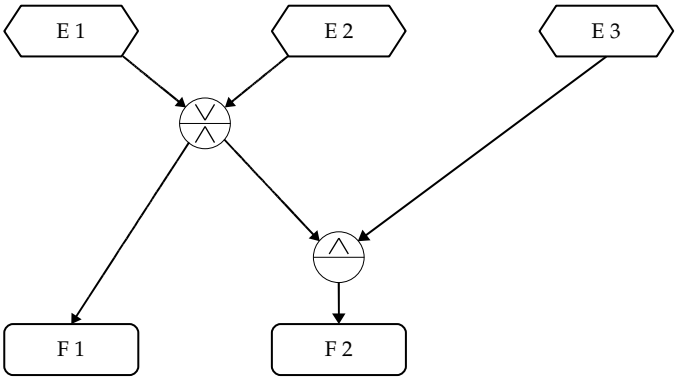


Abbildung 3.9: *„Wenn Ereignis E 1 oder E 2 eingetreten ist, startet die Funktion F 1; wenn Ereignis E 1 oder E 2 und Ereignis E 3 eingetreten sind, startet die Funktion F 2" (Sch98a). (Auch wenn die Beschreibung vermuten lässt, dass im zweiten Fall ausschliesslich F 2 startet, muss die Interpretation so lauten, dass dann zusätzlich zu F 1 die Funktion F 2 startet.)*

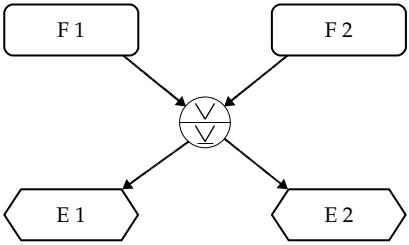


Abbildung 3.10: *„Wenn die Funktion F 1 oder F 2 abgeschlossen ist, tritt entweder Ereignis E 1 oder Ereignis E 2 ein" (Sch98a).*

Abb. 3.11 nach Sch98a zeigt einen Ausschnitt aus einem funktionsorientierten Organigramm einer prozessorientierten Gliederung.

Den Organisationseinheiten können nebst den Funktionen Daten zugeordnet werden. Bspw. kann festgelegt werden, welche Organisationseinheiten

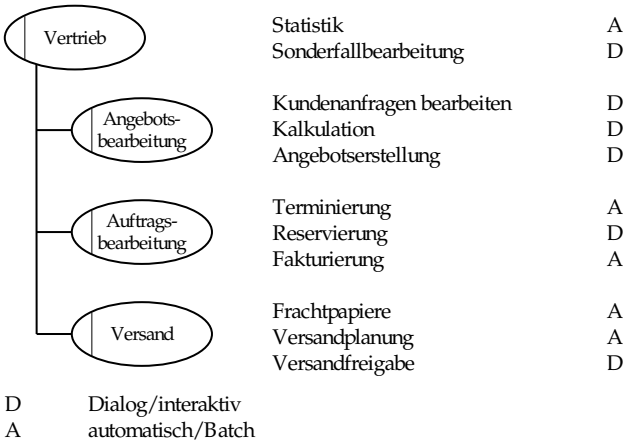


Abbildung 3.11: *Ausschnitt eines funktionsorientierten Organigramms nach Sch98a.*

schreibenden und welche lesenden Zugriff auf die Kundenstammdaten haben und wer für die Daten verantwortlich ist. Letzteres erlaubt es der Ereignissteuerung, an Hand der Daten zu erkennen, welche Organisationseinheit zuständig ist.

3.1.1.5 Vorgangskette als komprimierte Gesamtsicht

Werden Ereignisse, Funktionen, Datenelemente, Informationsobjekte, Organisationseinheit, Bearbeitungsform sowie die Zugriffsberechtigung in einer tabellarischen Graphik erfasst, ergibt sich das Vorgangskettendiagramm als komprimierte Gesamtsicht. Auch wenn einzelne Details fehlen können, ist die Prozesskette in detaillierter Form wiedergegeben. Einzelne Spalten können ausgeblendet werden, wodurch sich das Vorgangskettendiagramm den jeweiligen Bedürfnissen anpassen lässt. Die Abb. 3.12 stammt aus Sch98a, Bro91 und zeigt einen Ausschnitt aus einem Auftragsbearbeitungsprozess als Vorgangskette.

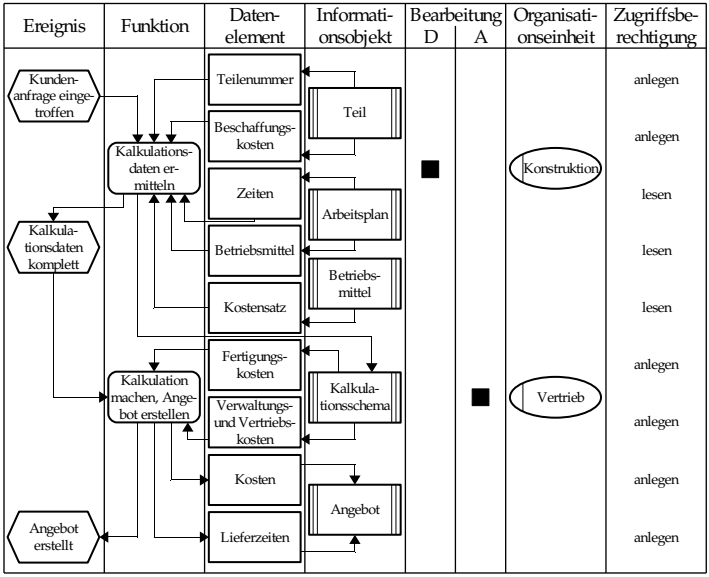


Abbildung 3.12: *Ausschnitt aus einer Auftragsbearbeitung als Vorgangskette, aus Sch98a, Bro91.*

3.1.2 Formalisierung

In der Vergangenheit wurden einige Versuche unternommen, die Semantik und die Syntax von Ereignisgesteuerten Prozessketten zu formalisieren. Nüttgens und Rump (NJ02) führen etliche Ansätze an, um anschliessend einen eigenen Entwurf für das Kontrollflusskonzept der Ereignisgesteuerten Prozessketten zu präsentieren. Dieser sieben Seiten umfassende Ansatz gliedert sich in die Definition einer Syntax und einer Semantik. Im selben Jahr wurde der Arbeitskreis *Geschäftsprozessmanagement mit Ereignisgesteuerten Prozessketten WI-EPK* in den Fachbereich Wirtschaftsinformatik der Gesellschaft für Informatik e. V. (GI) überführt.

3.1.3 Bewertung

Im folgenden werden die EPK gemäss dem in Abschnitt 1.5 vorgestellten Vorgehen an den in Kap. 2 aufgestellten Kriterien beurteilt.

K 1 Abstraktionsfähigkeit	•••
K 2 Implementierbarkeit	••[3]
K 3 Integrationsfähigkeit	••••
K 4 Korrektheit	••[3]
K 5 Lesbarkeit	••[3]
K 6 Minimalität	••••
K 7 Uniformität	••••
K 8 Vollständigkeit	••••
K 9 Wirtschaftlichkeit	••[4]

Tabelle 3.1: *Die qualitative Beurteilung der EPK gemäss den in Kap. 2 hergeleiteten Kriterien.* • *steht für minimal,* •••• *für maximal erfüllt.*

3.1.3.1 Abstraktionsfähigkeit

Mit dem Funktionsbaum, beschrieben in 3.1.1.2, sind die vertikale und die horizontale Gliederung möglich. Funktionsdarstellungen abstrahieren von Ereignissen und Kontrollelementen und erfahren damit Einschränkungen, welche durch Kombination mit Prozessketten aufgehoben werden können. Dies führt jedoch zu unterschiedlichen Abstraktionsebenen innerhalb einer Darstellung, welche sich auf die Lesbarkeit auswirken können.

3.1.3.2 Implementierbarkeit

Wegen der fehlenden Formalisierung, der semantischen Unklarheiten und der syntaktischen Unsicherheiten ist die Implementierbarkeit eingeschränkt gegeben. Die Erfüllung der Implementierbarkeit setzt voraus, dass die in Abschnitt 3.1.2 erwähnten Formalisierungsregeln eingehalten werden.

[3] unter Einhaltung der Formalregeln (s. Abs. 3.1.2) •••

[4] bei vorhandener aktiver Unterstützung durch Modellierwerkzeug ••••

3.1.3.3 Integrationsfähigkeit

Mittels Ereignissen oder Funktionen lassen sich Schnittstellen zu weiteren EPK- und Nicht-EPK-Modellen herstellen.

3.1.3.4 Korrektheit

Die Korrektheit kann nur bei Anwendung der Formalisierungsregeln erbracht werden.

3.1.3.5 Lesbarkeit

Die Lesbarkeit wird auf Grund der wenigen Symbole unterstützt, ist aber wegen der nicht eindeutigen Semantik eingeschränkt, solange nicht die Formalisierungsregeln angewandt werden. Zudem unterscheiden sich Symbole unterschiedlicher Semantik im Aussehen nur schwach. Eine weitere Einschränkung erfährt die Lesbarkeit durch die möglicherweise vorhandenen unterschiedlichen Abstraktionsebenen, wenn sowohl Funktionen wie Prozessketten kombiniert vorliegen.

3.1.3.6 Minimalität

Modellelemente können nicht untereinander substituiert werden, somit ist die Minimalität gegeben.

3.1.3.7 Uniformität

Semantisch vergleichbare Konstrukte sehen gleich aus, unterschiedlich aussehende Konstrukte haben unterschiedliche Semantik, damit ist die Uniformität erfüllt.

3.1.3.8 Vollständigkeit

Relevante Sachverhalte können die EPK alleine auf Grund ihrer Ausrichtung abbilden, weitere Sachverhalte lassen sich, bspw. als zusätzliche Spalten in einem Vorgangskettendiagramm, erfassen.

3.1.3.9 Wirtschaftlichkeit

Die Wirtschaftlichkeit setzt ein Hilfsmittel in Form eines Werkzeuges mit integrierter Formalitätsprüfung voraus, ansonsten der Aufwand für den Entwurf eines konsistenten Modells hoch ausfallen kann.

3.2 Prozessmodellierung mit der Unified Modeling Language

3.2.1 Übersicht

Die *Unified Modeling Language (UML)* ist eine graphische Beschreibungssprache, die auf das Software Engineering ausgerichtet ist und den Aufbau sowie das Verhalten von Informationssystemen aus einer objektorientierten Sichtweise beschreibt. Entstanden ist UML aus dem Zusammenschluss der Arbeiten von Booch, Rumbaugh, Jacobson und weiteren Autoren. Mit *Unified Method 0.8* stellten die drei 1995 einen ersten Entwurf vor, der ein Jahr später unter dem heutigen Namen in der Version 0.9 firmierte. Im Herbst 1997 brachte die *Object Management Group OMG*[5] eine Version 1.1 heraus. Mittlerweilen ist die Version 2.1 aktuell. Während in ursprünglichen Versionen die Vereinheitlichung der graphischen Notation im Vordergrund stand, wurde mit Erscheinen der Version 2.x das UML zugrunde liegende Metamodell überarbeitet. Weitere Diagrammtypen sind hinzugekommen, mittlerweilen sind es sechzehn[6].

UML ist nicht formal beschrieben, sondern mittels einem Metamodellansatz definiert[7]. Das Metamodell gliedert sich in die Ebenen *M0*, *M1*, *M2* und *M3* auf[8]. M3 ist die Top-Ebene und beherbergt die *Meta Object Facility (MOF)*.

[5] Die *Object Management Group, Inc. (OMG)* wurde 1989 gegründet; sie ist ein für alle offenes Konsortium, welches Spezifikationen zu Interoperabilität und Wiederverwendbarkeit von Unternehmensanwendungen in verteilten, heterogenen Umgebungen erarbeitet und unterhält. Weitere Informationen sind unter http://www.omg.org/ einsehbar.

[6] Es gibt keine offizielle Übersicht über die Diagramme; Diagramme in UML sind spezielle Sichten auf das Repository und lassen sich relativ frei definieren (Oes09).

[7] Eine spätere Formalisierung wird nicht ausgeschlossen: „It is important to note that the specification of UML as a metamodel does not preclude it from being specified via a mathematical formal language (e. g., Object-Z or VDM) at a later time“ (Obj07).

[8] Weitere Ebenen können laut OMG in Zukunft hinzukommen.

Die MOF stellt ein Rahmenwerk zur Verwaltung von Metadaten sowie eine Reihe von Metadaten-Services zur Verfügung, um die Entwicklung und Interoperabilität von modell- und metadatengetriebenen Systemen zu ermöglichen (s. Obj06). M2, eine Ebene tiefer, beinhaltet nebst weiteren Metamodellen das Metamodell UML. Auf der Ebene M1 findet sich das Benutzermodell, in dem die Diagramme von UML abgelegt sind. Auf der untersten Ebene, in M0, schliesslich sind die Objekte (Instanzen) beheimatet.

Die Diagrammtypen auf der Ebene M1 lassen sich in statische, die Struktur beschreibende, und in dynamische, das Verhalten beschreibende, unterteilen. Die verschiedenen Diagrammtypen eignen sich für die in den Projektphasen anfallenden Erfordernisse. Die unten vorgestellten Aktivitäten kommen bei der Analyse von dynamischen Vorgängen zum Zuge, indem sie die Abläufe resp. Geschäftsprozesse zeigen, während zur Spezifikation und Implementierung andere Diagrammtypen wie bspw. die Sequenz-, Kommunikations-, Klassen- oder Verteilungsdiagramme eingesetzt werden. Im Gegensatz zu früheren Versionen lassen sich in UML 2.x die Diagrammtypen als verschiedene Sichten auf ein Modell interpretieren, wobei zu beachten ist, dass die Reihenfolge ihrer Anwendung im Entwicklungsprozess i. a. vorgegeben ist und i. d. R. nicht alle Diagrammtypen zum Zuge kommen.

3.2.2 Prozessmodellierung mittels UML-Aktivitäten

In UML 2.x werden Ablaufdiagramme als Aktivitäten bezeichnet. Aktivitäten bestehen aus Start-, End- und Ablauf-Ende-Knoten[9], Aktions-, Objekt- und Kontrollknoten sowie Objekt- und Kontrollflüssen. Sie werden im Software-Entwicklungsprozess üblicherweise in der Anforderungsanalyse zur Abbildung der vorhandenen oder geplanten Arbeitsabläufe eingesetzt.

3.2.3 Die einzelnen Elemente

3.2.3.1 Aktivitätsparameter, Start- und Endknoten, Ablauf-Ende

Eine Aktivität hat mindestens einen Start- und mindestens einen Endknoten. An ihre Stelle können Eingangs- und Ausgangsparameter treten, die der Ob-

[9] Um Missverständnissen vorzubeugen, wird hier im Ggs. zu Oes09 die Schreibweise mit Bindestrich präferiert.

jektübernahme und -übergabe dienen (s. dazu 3.2.3.5). Ein Startknoten, dargestellt als ausgefüllter Kreis, hat einen oder mehrere Ausgänge und ist ein Startpunkt in einem Ablauf innerhalb einer Aktivität. Ein Endknoten, dargestellt als eingerahmter, ausgefüllter Kreis, hat einen oder mehrere Eingänge und ist ein Endpunkt in einem Ablauf innerhalb einer Aktivität. Sobald ein Endknoten in einem Ablauf erreicht wird, werden sämtliche Abläufe in einer Aktivität gestoppt. Ist dieses Verhalten nicht gewünscht, tritt ein Ablauf-Ende, ein als mit einem Kreuz versehener Kreis, an dessen Stelle.

3.2.3.2 Vor- und Nachbedingung

Auf der Aktivität lassen sich Vor- und Nachbedingungen mittels der Schlüsselwörter *„precondition“* und *„postcondition“* angeben.

3.2.3.3 Entscheidung und Zusammenführung, Teilung und Synchronisation

Entscheidungs- und Zusammenführungsknoten werden mittels Rautensymbol dargestellt. Im Falle, dass nur ein Eingang, aber mehrere Ausgänge vorliegen, wird von einer Entscheidung gesprochen. Zugunsten welchem Ausgang entschieden wird, kann mittels in eckige Klammern gesetzte Bedingungen festgelegt werden. Wenn mehrere Eingänge, aber nur ein Ausgang vorliegen, handelt es sich um eine Zusammenführung; sobald ein Eingang aktiv ist, wird an den Ausgang weitergeleitet. Es ist möglich, dass mehrere Eingänge und mehrere Ausgänge vorhanden sind, dann wird von einer Entscheidung und Zusammenführung gesprochen.

Teilungs- und Synchronisationsknoten werden mittels Balkensymbol dargestellt. Wenn nur ein Eingang, aber mehrere Ausgänge vorliegen, wird von einer Teilung gesprochen. Die Teilung leitet den Fluss an alle Ausgänge weiter. Wenn mehrere Eingänge, aber nur ein Ausgang vorliegen, handelt es sich um eine Synchronisation; sobald alle Eingänge aktiv sind, wird an den Ausgang weitergeleitet. Es ist möglich, dass mehrere Eingänge und mehrere Ausgänge vorhanden sind, dann wird von einer Teilung und Synchronisation gesprochen.

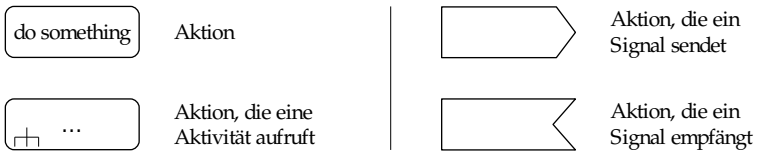


Abbildung 3.13: *Aktionen in UML 2.*

3.2.3.4 Aktionen

Die Aktion ist die Repräsentation des kleinsten Elementes, um ein Verhalten darzustellen. Graphisch dargestellt wird eine Aktion als ein abgerundetes Rechteck. Im Falle dass die Aktion an den Ein- und Ausgängen Übergabeparameter benötigt, bspw. weil sie Daten empfängt, bearbeitet oder generiert, kann dies mittels sog. Pins verdeutlicht werden (s. 3.2.3.5). Wenn eine Aktion sich so verhält, dass sie eine Aktivität aufruft, wird dies mit einem Gabelsymbol deutlich gemacht. Abweichende graphische Darstellungen treten auf, wenn die Aktion ein Signal sendet oder empfängt (s. Abb. 3.13).

3.2.3.5 Objektknoten

Der Objektknoten dient dazu, Objekte (dies können auch Daten sein) zwischenzuspeichern und sie von einer Aktion an die nachfolgende Aktion zu übergeben. Objektknoten werden als sog. Pins, kleine, an einer Aktion anliegende Rechtecke, oder als eigenständige, zwischen zwei Aktionen liegende Rechtecke dargestellt. Der Zustand der repräsentierten Objekte kann in rechteckigen Klammern angegeben werden. Einzelne Pins können zu sog. Parametergruppen zusammengefasst werden, ersichtlich an der Einrahmung; handelt es sich beim Objekt um eine Menge, lässt sich das durch vier aneinandergereihte Pins darstellen. Weitere Varianten sind der Ausnahmeparameter, durch ein Dreiecksymbol kenntlich gemacht, oder die Angabe, dass es sich beim Objekt um einen Datenstrom handelt, ausgezeichnet entweder durch Ausfüllen der Pins oder den Text *{stream}*. Eine Übersicht zu den Objektknoten gibt Abb. 3.14.

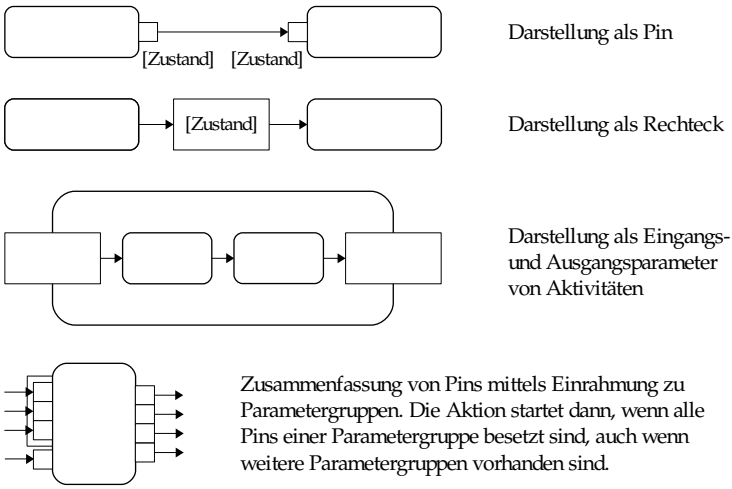


Abbildung 3.14: *Darstellungen von Objektknoten.*

3.2.3.6 Strukturierter Knoten

Um die Übersichtlichkeit zu wahren, lassen sich mehrere Aktionen, die der Fallunterscheidung, der wiederholten Ausführung oder der Bearbeitung von Massendaten dienen, mittels einer gestrichelten Linie zu einem strukturierten Knoten zusammenfassen (s. Abb. 3.15).

3.2.3.7 Kontroll- und Objektfluss

Der Fluss in einer Aktivität wird durch gerichtete Kanten dargestellt. Alternativ können gerichtete Kanten durch Konnektoren – ein zusammengehörendes Kreispaar mit eindeutigem Bezeichner – ersetzt werden, um bspw. die Übersichtlichkeit zu bewahren. Beim Fluss kann es sich sowohl um den Kontroll- wie den Objektfluss handeln. Den Objektfluss erkennt man daran, dass er Objektknoten mit Aktionsknoten resp. Pins mit Pins verbindet; ist er mit dem Kontrollfluss identisch, wird nur ein Fluss gezeichnet. Gesteuert werden der Kontroll- und der Objektfluss durch Aktions- und Kontrollknoten sowie durch Objektknoten. Die Kombination der einzelnen Elemente führt zu

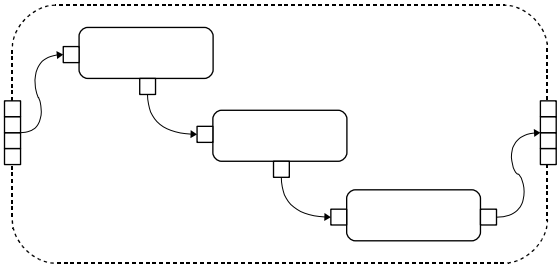


Abbildung 3.15: *Strukturierter Knoten.*

den in Abb. 3.16 gezeigten möglichen Kontrollstrukturen. Deren Semantik ist ebenfalls in der Abbildung angegeben.

3.2.4 Zuständigkeit

Zuständigkeiten für Aktionen lassen sich auf zwei Arten darstellen. Entweder wird für jede Zuständigkeit mittels senkrechter Linien eine Schwimmbahn angelegt, in der die entsprechenden Aktionen plaziert werden, oder die Zuständigkeit wird auf den Aktionen in runden Klammern angegeben.

3.2.5 Bewertung

3.2.5.1 Abstraktionsfähigkeit

Die vertikale Abstraktion ist mittels Aktionen, die Aktivitäten aufrufen (s. Abschnitt 3.2.3.4), realisiert, eine horizontale Abstraktion ist mit demselben Verfahren machbar, wobei der Modellierer darauf achtgeben muss, die von den Aktionen aufgerufenen Aktivitäten jeweils auf derselben Abstraktionsebene darzustellen.

3.2.5.2 Implementierbarkeit

Die Implementierbarkeit ist mit dem Metamodellansatz gewährleistet, allerdings sind bei einer Implementierung sehr viele Regeln zu berücksichtigen.

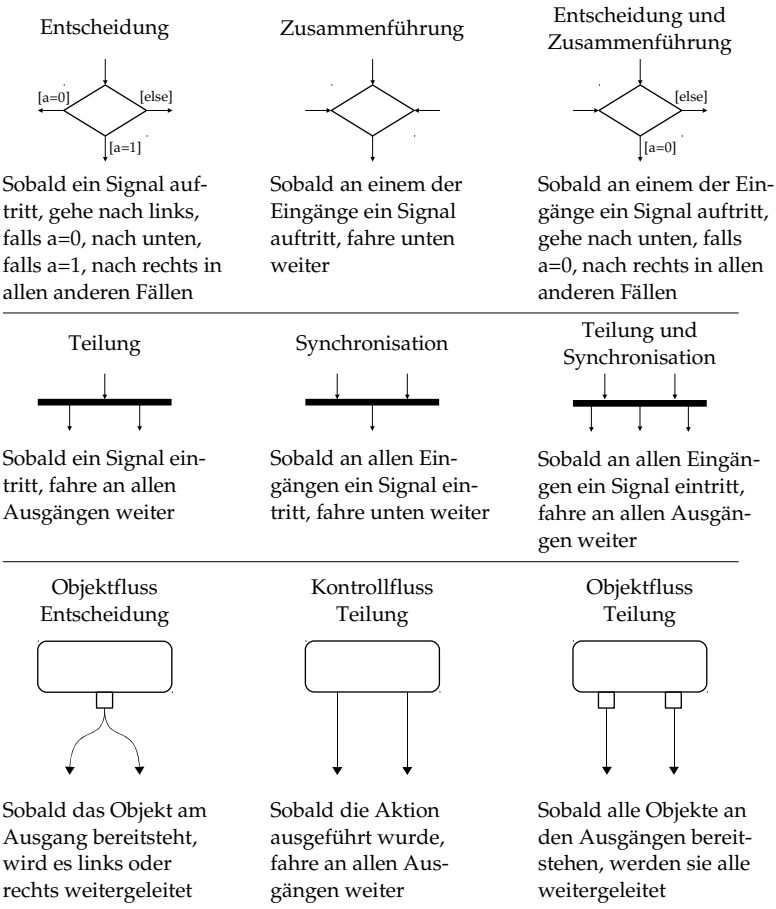


Abbildung 3.16: *Kontroll- und Objektflussstrukturen.*

3.2.5.3 Integrationsfähigkeit

Die Integrationsfähigkeit ist nur dann gegeben, wenn weitere Aktivitäten[10] eingebunden werden sollen; für eine Verknüpfung mit anderen UML-2-Diagrammen oder Nicht-UML-2-Diagrammen ist es notwendig, auf UML-2-Elemente ausserhalb der Aktivitäten zuzugreifen.

3.2.5.4 Korrektheit

Die Korrektheit kann nur dadurch erfüllt werden, dass weitere UML-2-Diagramme hinzugenommen werden, um die Vollständigkeit zu gewährleisten.

3.2.5.5 Lesbarkeit

Die Lesbarkeit ist auf Grund der vielen vorherrschenden, teils impliziten Regeln, der relativ hohen Anzahl an Symbolen und der Integration von Kontroll- und Objektfluss mit unterschiedlichen Regeln eingeschränkt.

K 1 Abstraktionsfähigkeit	•••
K 2 Implementierbarkeit	•••
K 3 Integrationsfähigkeit	•[11]
K 4 Korrektheit	••[11]
K 5 Lesbarkeit	••
K 6 Minimalität	•
K 7 Uniformität	•
K 8 Vollständigkeit	••[11]
K 9 Wirtschaftlichkeit	••

Tabelle 3.2: *Die qualitative Beurteilung von UML 2 unter ausschliesslicher Berücksichtigung der Aktivitätsdiagramme gemäss den in Kap. 2 hergeleiteten Kriterien.* • *steht für minimal,* •••• *für maximal erfüllt.*

[10] Aktivitäten hier im Sinne von UML 2

[11] unter Zuhilfenahme weiterer Diagramme und Elemente aus UML 2 ••••, jedoch zu Lasten anderer Kriterien, insbesondere der Minimalität und der Uniformität

3.2.5.6 Minimalität

Die Minimalität ist verletzt. Bspw. können die Pins zur Darstellung von Objekten durch Rechtecke ersetzt werden. Werden zum Erreichen der Vollständigkeit weitere UML-2-Diagramme hinzugezogen, kann die Minimalität wegen der dadurch möglicherweise entstehenden semantischen Überlappung zusätzlich verletzt werden.

3.2.5.7 Uniformität

Für den Kontrollfluss und den Objektfluss wird dasselbe Symbol verwendet. Umgekehrt können zur Darstellung von Objekten und Streams – kontinuierlichen Objektflüssen – unterschiedliche Symbole verwendet werden. Zur Darstellung der Oder-Verzweigung existiert im Falle des Kontrollflusses ein explizites Symbol, beim Objektfluss dagegen nicht.

3.2.5.8 Vollständigkeit

Die Aktivitäten von UML 2 beschränken sich weitgehend auf die Darstellung des Ablaufs. Zur Abbildung weiterer Sachverhalte stellt UML 2 zusätzliche Diagramme bereit, die sich aber hinsichtlich ihres Aussehens, ihrer Semantik und ihrer Syntax zum Teil stark unterscheiden, aber auch überlappen, so dass zum Erreichen der Vollständigkeit andere Kriterien, insbesondere die Minimalität und die Uniformität, beeinträchtigt werden.

3.2.5.9 Wirtschaftlichkeit

Der Aufwand zum Erstellen eines Modells wird davon abhängen, welche Aspekte abzubilden sind. Es kann davon ausgegangen werden, dass weitere Diagramme aus UML 2 hinzugezogen werden müssen, um ein vollständiges Modell zu erhalten. Dies wird sich in einem entsprechend hohen Aufwand niederschlagen.

Kapitel 4

Prozessmodellierung mit Petri-Netzen

4.1 Übersicht

Petri-Netze wurden 1962 durch Carl Adam Petri in seiner Dissertation *Kommunikation mit Automaten* (Pet62) vorgestellt. Ursprünglich aus der theoretischen Informatik hervorgegangen[1], finden Petri-Netze mittlerweilen in verschiedenen Gebieten Anwendung und werden dabei laufend variiert und erweitert. Mit Petri-Netzen lassen sich die Nebenläufigkeit und die Synchronizität von Prozessen sowie der Zustand eines Systems ausdrücken.

Petri-Netze bestehen aus einigen wenigen Elementen, die je nach Einsatzgebiet unterschiedlich interpretiert werden können. Die graphische Repräsentation macht sie leicht verständlich, die mathematische Repräsentation unterstützt die Analyse auf bestimmte Modelleigenschaften. Trotz ihrer Einfachheit und leichten Verständlichkeit haben Petri-Netze viele theoretische Fragestellungen hervorgebracht. In der folgenden Übersicht steht jedoch der Anwendungsbereich innerhalb der Geschäftsprozessmodellierung im Vordergrund. Weitergehende Literatur zu Petri-Netzen findet sich u. v. a. in Rei86, Mur89.

Petri-Netze werden im betriebswirtschaftlichen Umfeld eher zögerlich für die Geschäftsprozessmodellierung eingesetzt[2], obwohl sie sich aus mehreren

[1] Im Vorwort zu Rei10 schreibt Petri, dass er die Petri-Netze bereits 1939 dazu erfunden hat, chemische Prozesse anschaulich beschreiben zu können.

[2] Ausnahmen gibt es: Khn05 setzt höhere Petri-Netze zur Optimierung von Geschäftsprozessen in der Baubranche ein. AG03 führen mehrere Gründe für den Einsatz von Petri-Netzen im Workflow-

Gründen dazu eignen: Sie sind bestens untersucht, formal und doch einfach verständlich, lassen sich beliebig erweitern, können in andere Prozessmodelle umgewandelt werden[3], und mittels zahlreich vorhandener und meist frei verfügbarer Werkzeuge lassen sie sich auf verschiedenste Eigenschaften hin untersuchen.[4]

4.2 Grundlegende Petri-Netze

Das den Petri-Netzen zugrunde liegende Netz lässt sich als ein Tripel $\mathcal{N} = (S, T, F,)$ beschreiben, wobei
$S = \{s_1, s_2, \ldots, s_m\}$ eine Menge von Stellen,
$T = \{t_1, t_2, \ldots, t_n\}$ eine Menge von Transitionen,
$F \subseteq (S \times T) \cup (T \times S)$ eine Menge von Kanten (Fluss-Relation) ist und
$S \cap T = \emptyset$ und $S \cup T \neq \emptyset$ gilt.

Sind S und T endlich, wird $\mathcal{N}$ als endlich bezeichnet.

Abhängig von Variante oder Sprachgebrauch, werden Stellen auch als Plätze, Kanäle, Bedingungen oder Knoten und Transitionen als Instanzen, Ereignis oder Prädikat bezeichnet.

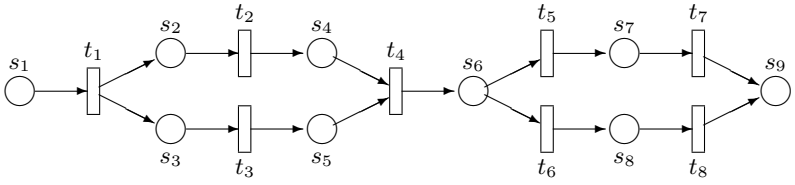


Abbildung 4.1: *Einfaches Netz.*

Management an. Und auch der Verfasser begegnet in seiner beraterischen Tätigkeit verschiedentlich Geschäftsprozessmodellen in Form von Petri-Netzen.

[3]Die Umkehrung gilt nur dann, wenn die durch die Umwandlung gewonnenen Prozessmodelle ebenfalls formal sind, dies ist oft nicht der Fall.

[4]Als weiteres Argument kann hinzugefügt werden, dass Petri-Netze, wenn sie um ein weiteres Element, den Inhibitor, welcher das Feuern einer Transition bei Überschreiten einer Mindestanzahl Marken verhindert, erweitert werden, Turing-mächtig sind.

Abb. 4.1 zeigt ein Beispiel eines einfachen Netzes, das mathematisch wie folgt beschrieben werden kann:
$S = \{s_1, s_2, \ldots, s_9\}$,
$T = \{t_1, t_2, \ldots, t_8\}$,
$F = \{(s_1, t_1), (t_1, s_2), (t_1, s_3), (s_2, t_2), (s_3, t_3), (t_2, s_4), (t_3, s_5), (s_4, t_4), \ldots\}$,
wobei die Elemente als Beispiel folgende Bedeutung haben können:
s_1: Bestellung eingegangen
t_1: Lager und Buchhaltung informieren
s_2: Lager ist informiert
s_3: Buchhaltung ist informiert
t_2: Rüstschein schreiben
s_4: Rüstschein ist erstellt
t_3: Debitoren buchen
s_5: Debitoren sind gebucht
$\vdots$

Dieses unvollständige Beispiel eines Bestellprozesses ist ein sogenanntes *Kanal-Instanz-Netz*. Die Stellen werden hier als Kanal bezeichnet; sie haben passiven Charakter und dienen der Beschreibung von möglichen Zuständen des Systems. Die Transitionen werden als Instanzen bezeichnet; sie sind aktive Elemente und stehen für Aktivitäten. Kanal-Instanz-Netze eignen sich zur Darstellung der Struktur und Funktionsweise eines Prozesses auf hoher Abstraktionsebene. Dessen aktueller Zustand kann damit nicht ausgedrückt werden.

Die Nachbarschaftsbeziehung der einzelnen Elemente in einem Netz kann mit folgender Notation erfolgen (s. Val03): Sei $x \in S \cup T$ ein Element, dann bezeichnet $\bullet x := \{y \in S \cup T \mid (y, x) \in F\}$ die Menge aller Eingangselemente von x und $x\bullet := \{y \in S \cup T \mid (x, y) \in F\}$ die Menge aller Ausgangselemente von x. Diese Definition kann auf eine Teilmenge $A \subseteq S \cup T$ ausgedehnt werden mit $\bullet A := \{y \mid \exists\, x \in A\,.\,(y, x) \in F\}$ und $A\bullet := \{y \mid \exists\, x \in A\,.\,(x, y) \in F\}$.

Um zu verstehen, wie der Kontrollfluss im Netz von Abb. 4.1 aussieht, können die *Bedingung-Ereignis-Netze* als die einfachste Variante von Petri-Netzen herangezogen werden. Die Stellen in einem Bedingung-Ereignis-Netz werden als Bedingung interpretiert, die Transitionen als Ereignisse. Zusätzlich kommen die Marken vom Datentyp *boolean* hinzu. Sie repräsentieren die Gültig-

keit einer Bedingung, d. h., wenn eine Stelle mit einer Marke belegt ist, gilt die Bedingung als erfüllt. Eine Stelle kann keine oder genau eine Marke tragen, was sich mit der Markierungsfunktion $M : S \rightarrow \{0, 1\}$ darstellen lässt. Damit nun ein Ereignis t eintreten kann, müssen sämtliche vorgelagerten Bedingungen $\bullet t$ mit einer Marke versehen sein, sämtliche nachfolgenden Bedingungen $t\bullet$ müssen leer sein. Dies wird auch als Aktivierung bezeichnet. Nach dem Eintreten des Ereignisses sind die Marken in $\bullet t$ entfernt und $t\bullet$ mit einer Marke versehen. Wenn in Abb. 4.1 das Ereignis t_1 eintritt, da s_1 als die einzige Bedingung mit einer Marke belegt ist, sind anschliessend s_2 und s_3 mit einer Marke belegt, während s_1 leer ist. Mit Vektoren ausgedrückt, sieht dies folgendermassen aus: $(1, 0, 0, 0, 0, 0, 0, 0, 0) \xrightarrow{t_1} (0, 1, 1, 0, 0, 0, 0, 0, 0)$.

Auf dieselbe Weise werden die Ereignisse t_2, t_3 und t_4 eintreten, so dass in s_6 eine Marke zu liegen kommt. Nun stehen die Ereignisse t_5 und t_6 in Konkurrenz zueinander, da beide zwar eine erfüllte Bedingung haben, die Marke in s_6 aber nur von einem der beiden Ereignisse konsumiert werden kann. Dies wird als *Verhaltenskonflikt* bezeichnet. (Solange s_6 keine Marke trägt, liegt lediglich ein Strukturkonflikt vor.[5])

Ein Ereignis muss nicht sofort eintreten, wenn es aktiviert ist, d. h. wenn alle Bedingungen erfüllt sind. Vielmehr ist das Eintreten eines Ereignisses nichtdeterministisch und muss von aussen aufgelöst werden. Zudem ist ein Ereignis zeitlos und atomar.

Damit sind, abhängig von der Anfangsmarkierung M_0, folgende 2 Abläufe möglich:

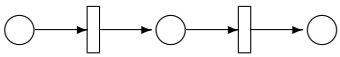
$$\begin{pmatrix} 1\\0\\0\\0\\0\\0\\0\\0\\0 \end{pmatrix} \xrightarrow{t_1} \begin{pmatrix} 0\\1\\1\\0\\0\\0\\0\\0\\0 \end{pmatrix} \xrightarrow{t_2, t_3} \begin{pmatrix} 0\\0\\0\\1\\1\\0\\0\\0\\0 \end{pmatrix} \xrightarrow{t_4} \begin{pmatrix} 0\\0\\0\\0\\0\\1\\0\\0\\0 \end{pmatrix} \xrightarrow{t_5} \begin{pmatrix} 0\\0\\0\\0\\0\\0\\1\\0\\0 \end{pmatrix} \xrightarrow{t_7} \begin{pmatrix} 0\\0\\0\\0\\0\\0\\0\\0\\1 \end{pmatrix}$$

und

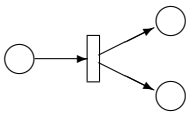
[5] Daneben gibt es Konstellationen, die nichtlokaler Natur sind und zu einer sogenannten *Konfusion* führen. Dies ist nicht Gegenstand der hiesigen Betrachtungen.

$$\begin{pmatrix}1\\0\\0\\0\\0\\0\\0\\0\\0\end{pmatrix} \xrightarrow{t_1} \begin{pmatrix}0\\1\\1\\0\\0\\0\\0\\0\\0\end{pmatrix} \xrightarrow{t_2,t_3} \begin{pmatrix}0\\0\\0\\1\\1\\0\\0\\0\\0\end{pmatrix} \xrightarrow{t_4} \begin{pmatrix}0\\0\\0\\0\\0\\1\\0\\0\\0\end{pmatrix} \xrightarrow{t_6} \begin{pmatrix}0\\0\\0\\0\\0\\0\\0\\1\\0\end{pmatrix} \xrightarrow{t_8} \begin{pmatrix}0\\0\\0\\0\\0\\0\\0\\0\\1\end{pmatrix}.$$

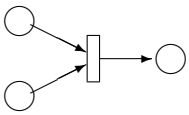
Die für die Modellierung eines Petri-Netzes notwendigen, in Abb. 4.2 gezeigten Grundkonstrukte liegen damit wie folgt vor:



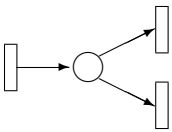
1) Sequenz



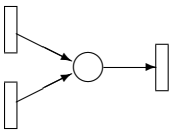
2) AND-Verzweigung



3) AND-Zusammenführung



4) XOR-Verzweigung



5) XOR-Zusammenführung

Abbildung 4.2: *Grundkonstrukte.*

1. *Sequenz*. Eine lineare Folge von Transitionen, die Marken in vorgelagerten Stellen entfernen und in nachgelagerten Stellen einfügen.

2. *AND-Verzweigung*. Eine Transition mit mehreren nachfolgenden Stellen, die eine Nebenläufigkeit einleitet.

3. *AND-Zusammenführung*. Eine Transition mit mehreren vorgängigen Stellen, die synchronisierenden Charakter hat.

4. *XOR-Verzweigung*. Mehrere Transitionen, die sich eine vorgängige Stelle teilen und somit in Wettbewerb zueinander stehen.

5. *XOR-Zusammenführung*. Mehrere Transitionen, die sich eine nachfolgende Stelle teilen und beim Eintreffen einer Marke feuern.

In *Stellen-Transition-Netzen* kann eine Stelle standardmässig beliebig viele Marken tragen, was durch eine Kapazitätsangabe K eingeschränkt werden kann. Die Marken sind dabei nicht zwingend vom Typ *boolean*, sie können von einem beliebigen Datentyp sein. Zusätzlich können die Kanten mit einer Gewichtsangabe W versehen sein, die angibt, wieviele Marken entfernt resp. eingetragen werden. Ohne Angabe eines Gewichts gilt $W = 1$. Es ergeben sich die 2 zusätzlichen Funktionen $W : F \rightarrow \mathbf{N}$ für die Kantengewichte und $K : S \rightarrow \mathbf{N}$ für die Stellenkapazitäten. Eine Transition kann nur dann schalten, wenn in $\bullet t$ mindestens soviele Marken vorhanden sind, wie im Kantengewicht angegeben ist, und die Kapazität von $t\bullet$ nach dem Schalten nicht überschritten wird. Es gilt also $\forall\, s \in \bullet t : M(s) \geq W^{\bullet t}(s,t)$ und $\forall\, s \in t\bullet : M(s) \leq C(p) - W^{\bullet t}(t,s)$. Das Vorhandensein mehrerer Marken kann als Zahl in der Stelle angegeben sein; bei nur wenigen Marken können diese auch einzeln gezeichnet werden. Kantengewichte werden als Zahl angegeben. Das in Abb. 4.3 links abgebildete Stellen-Transition-Netz sieht nach dem Schalten der Transition t_1 wie in Abb. 4.3 rechts aus. In diesem Beispiel lässt t_1 die Anzahl Marken unangetastet.

Abbildung 4.3: *Stellen-Transition-Netz.*

4.3 Höhere Petri-Netze

Petri-Netze wurden und werden laufend erweitert und variiert. Bspw. lassen sich Petri-Netzen temporale Aspekte, um Aussagen zum Zeitbedarf einer Transition zu machen, oder stochastische Aspekte, die bei der Simulation hilfreich sind, hinzufügen. Eine Variation der Petri-Netze ist die Individualisierung der Marken. Dies erlaubt es, eine Marke in Abhängigkeit ihrer Beschaffenheit unterschiedlich zu behandeln. Bekannte Vertreter sind die Prädikat-Transition-Netze (GL79, GL81) und die Coloured Petri Net (Jen81, Jen98); beide sind eng miteinander verwandt und zeichnen sich dadurch aus, dass auf den Stellen und Transitionen zusätzliche Bedingungen (Prädikate) formuliert werden können[6].

Abb. 4.4 zeigt ein Beispiel eines höheren Petri-Netzes aus Rei85 am Beispiel einer Tankstelle. Auf den Kanten sind die Variablen aufgetragen, die als Eingang resp. Ausgang einer Transition benötigt werden. In den Stellen sind die aktuell zur Verfügung stehenden Objekte als individuelle Marken angegeben. Im Beispiel ist soeben ein Dieselfahrzeug (A,D) in die leere Tankstelle eingefahren und wird, da es freie Plätze hat, als nächstes zur Zapfsäule fahren und damit einen der vier Standplätze C, E, F und G beanspruchen. Nach dem Auftanken an der Zapfsäule für Diesel D und der Bezahlung beim verfügbaren Tankwart T oder U wird einerseits die Zapfsäule wieder freigegeben und andererseits der vollgetankte Wagen von der Zapfsäule wegfahren und damit den Standplatz wieder freigeben, bevor er die Tankstelle über die Tankstellenausfahrt verlässt.

Ein anderes Beispiel zeigt Abb. 4.5. Auf den Kanten sind die Eingangs- und Ausgangsvariabeln vermerkt und in den Transitionen die Vor- und Nachbedingungen. Gezeigt ist das Petri-Netz vor dem Schalten (oben) und nach dem Schalten (unten). Ist die im Beispiel oben dargestellte Vorbedingung $3\,a = b$ erfüllt, kann die Transition schalten, was zur Nachbedingung $n = 2\,a + b$ führt. Die Vorbedingung wird von den Variablen $a = 2$ und $b = 6$ erfüllt, da $b = 3\,a$ gilt. Nach dem Schalten wird die Nachbedingung erfüllt, es gilt $n = 10 = 2\,a + b$.

[6] Für eine Abgrenzung von Prädikat-Transition-Netzen und Coloured Petri Nets wird auf die Fachliteratur verwiesen, z. B. Gru88.

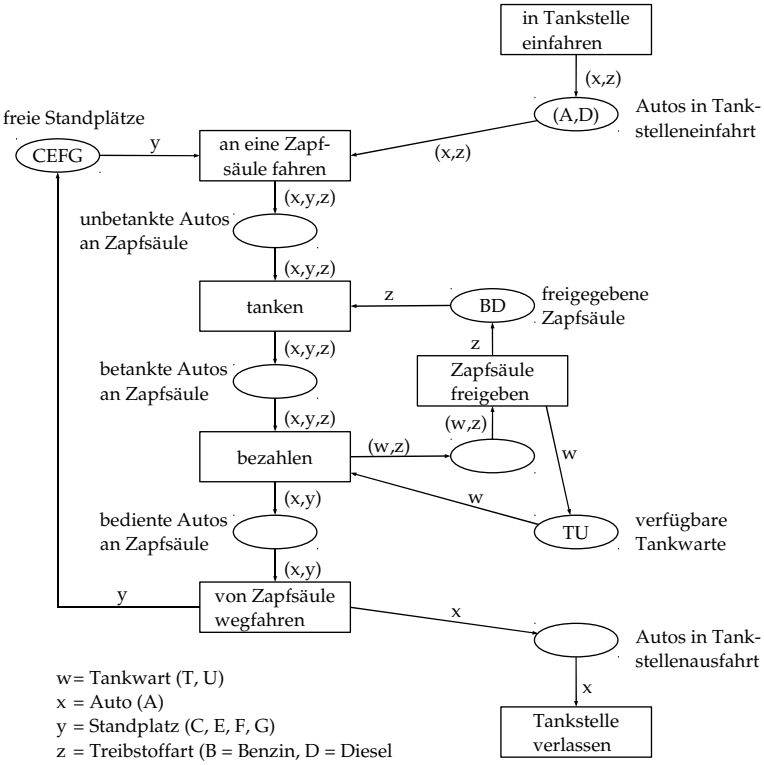


Abbildung 4.4: *Beispiel eines höheren Petri-Netzes mit individuellen Marken und Prädikaten an Hand einer Tankstelle. Zum Ablauf s. Text. Aus Rei85.*

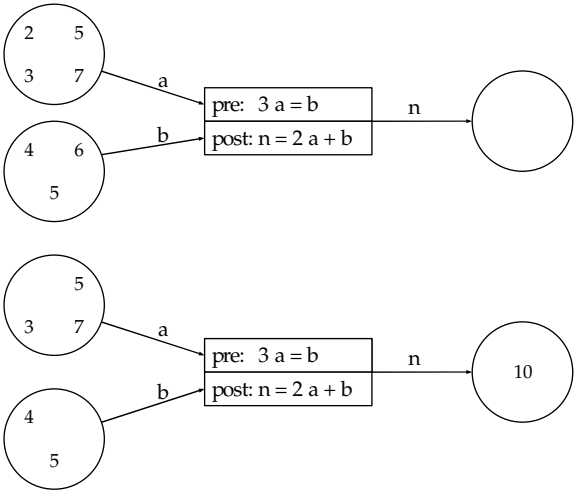


Abbildung 4.5: *Höheres Petri-Netz an einem einfachen Beispiel. Für Erläuterungen s. Text.*

4.4 Bewertung

4.4.1 Abstraktionsfähigkeit

Die einfachen Petri-Netze erfüllen die Abstraktionsfähigkeit nicht, können jedoch entsprechend erweitert werden, so dass die vertikale und die horizontale Aufgliederung eines Modells möglich wird.

4.4.2 Implementierbarkeit

Dank ihrer Formalisierung, der Möglichkeit, sie mathematisch zu beschreiben, und der wenigen Regeln können Petri-Netze direkt und mit wenig Aufwand implementiert werden.

K 1 Abstraktionsfähigkeit	•[7]
K 2 Implementierbarkeit	••••
K 3 Integrationsfähigkeit	••[7]
K 4 Korrektheit	••[7]
K 5 Lesbarkeit	•••
K 6 Minimalität	••••
K 7 Uniformität	••••
K 8 Vollständigkeit	••[7]
K 9 Wirtschaftlichkeit	••

Tabelle 4.1: *Die qualitative Beurteilung von einfachen Petri-Netzen gemäss den in Kap. 2 hergeleiteten Kriterien.* • *steht für minimal,* •••• *für maximal erfüllt.*

4.4.3 Integrationsfähigkeit

Einfachen Petri-Netzen fehlt eine Möglichkeit, sie direkt mit einem weiteren Modell (Zielmodell) zu verknüpfen; stattdessen sind sie mit dem Zielmodell zu verschmelzen. Ist das Zielmodell kein Petri-Netz, sind sie daher in die Modellsprache, mit der das Zielmodell entstanden ist, zu übersetzen, was auf Grund ihrer Formalisierung möglich ist (solange das Zielmodell gewissen Regeln folgt).

Werden die Petri-Netze um Schnittstellen erweitert, ist die Integrationsfähigkeit gegeben.

4.4.4 Korrektheit

Auf Grund der schwach erfüllten Vollständigkeit einfacher Petri-Netze ist ein korrektes Modell kaum erstellbar. Erweiterte Petri-Netze, welche die Vollständigkeit erfüllen, erfüllen ebenfalls die Korrektheit, da die Regeln klar sind und sich die Eigenschaften eines Modells gut untersuchen lassen.

[7] in einfachen Petri-Netzen nicht explizit vorgesehen, aber mittels Erweiterung möglich und damit auch mit ••••wertbar

4.4.5 Lesbarkeit

Die geringe Anzahl an Symbolen und Regeln unterstützt die Lesbarkeit von Petri-Netzen, wobei zu erwähnen ist, dass sich die Funktionsweise von Kontrollstrukturen dem weniger geübten Leser nicht sofort erschliesst. Mit einfachen Petri-Netzen erstellte Modelle nehmen i. d. R. sehr schnell grossen Raum ein, was die Lesbarkeit beeinträchtigt, so dass auf höhere Petri-Netze ausgewichen werden muss.

4.4.6 Minimalität

Die Minimalität ist erfüllt, es kann kein Sprachelement durch ein anderes Sprachelement ersetzt werden.

4.4.7 Uniformität

Konstrukte mit gleichem Aussehen haben die gleiche Semantik; die Uniformität ist erfüllt.

4.4.8 Vollständigkeit

Da mit einfachen Petri-Netzen lediglich die Struktur und die Zustandswechsel eines Systems abgebildet werden können, ist die Vollständigkeit i. d. R. nicht gegeben. Werden Petri-Netze so erweitert, dass die Abbildung relevanter Sachverhalte möglich wird, ist die Vollständigkeit erfüllt.

4.4.9 Wirtschaftlichkeit

Mit einfachen Petri-Netzen erstellte Modelle werden rasch umfangreich, so dass auf höhere Petri-Netze ausgewichen werden muss. Dies kann zu einem erhöhten Abstimmungsbedarf in der Arbeitsgruppe, zu Anpassungen an den eingesetzten Werkzeugen und zu zusätzlichem Schulungsaufwand führen.

Kapitel 5

Authentic Petri Net: Modellieren mit Netzen

5.1 Konzeption

Die in diesem Kapitel dargelegte Konzeption von APN folgt der Semiotik und teilt sich in die Syntaktik, die Semantik und die Pragmatik.

5.2 Syntaktik

Der erste der beiden folgenden Abschnitte führt in die grundlegenden Elemente von APN ein. Dies geschieht an Hand einer informalen Einführung mit Vorstellung der graphischen Repräsentation. Der zweite Abschnitt geht auf die Möglichkeiten der Abstraktion und Dekomposition ein.

5.2.1 Aufbau eines APN

Ein Element in APN ist entweder ein Knoten, eine Aktivität oder eine gerichtete Kante. Ein Knoten ist entweder ein Anfangsknoten oder ein Endknoten. Eine Aktivität ist entweder eine einfache oder eine doppelte Aktivität. Eine doppelte Aktivität ist entweder eine AND-XOR-, eine XOR-AND-, eine AND-AND- oder eine XOR-XOR-Aktivität.

Symbol	Funktion	Etikette	Symbol	Funktion	Etikette
	einfache Aktivität	a		AND/XOR	ax
	Fluss	f		XOR/AND	xa
	Eingangsknoten	i		AND/AND	aa
	Ausgangsknoten	o		XOR/XOR	xx

Abbildung 5.1: *Symbole von APN.*

Ein APN ist aus einem oder mehreren Anfangsknoten, einem oder mehreren Endknoten, einer oder mehreren einfachen oder doppelten Aktivitäten und einer Flussrelation, die die einzelnen Elemente verbindet und zu einem Prozess vervollständigt, aufgebaut.

Ein Subnetz von APN ist ein APN ohne Anfangs- und Endknoten. Subnetze spielen bei der Abstraktion, Dekomposition und Aggregation eine Rolle.

Abb. 5.1 zeigt die graphische Repräsentation der Elemente (vgl. Fra03).

Eine Aktivität lässt sich in einen Eingangs- und in einen Ausgangsbereich unterteilen, wobei der Eingangsbereich per Definition links, der Ausgangsbereich rechts liegt, wenn der Kontrollfluss von links nach rechts verläuft. Eine Aktivität hat mindestens einen Eingang und mindestens einen Ausgang, eine einfache Aktivität hat genau einen Eingang und genau einen Ausgang. Ein Anfangsknoten hat keinen Eingang und genau einen Ausgang, ein Endknoten hat genau einen Eingang und keinen Ausgang.

Das Beispiel in Abb. 5.2 kann als ein Tupel $\langle A, AX, XA, AA, XX, I, O, F\rangle$ beschrieben werden, wobei gilt:

A ist die Menge der einfachen Aktivitäten, $A = \{a_1, a_2, a_3, a_4\}$.
AX ist die Menge der AND-XOR-Aktivitäten, $AX = \{ax_1\}$.
XA ist die Menge der XOR-AND-Aktivitäten, $XA = \{xa_1, xa_2\}$.
AA ist die Menge der AND-AND-Aktivitäten, $AA = \{aa_1\}$.
XX ist die Menge der XOR-XOR-Aktivitäten ist, $XX = \{xx_1\}$.
I ist die Menge der Eingangsknoten, $I = \{i\}$.
O ist die Menge der Ausgangsknoten, $O = \{o\}$.

$F = \{(i, xa_1), (xa_1, a_1), (xa_1, aa_1), (a_1, aa_1), \ldots, (xa_2, o)\}$ ist die Flussrelation *F*.

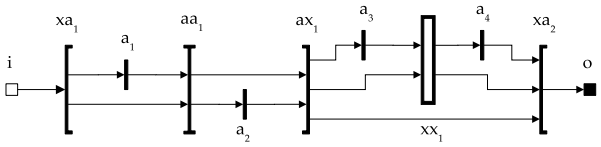


Abbildung 5.2: *Beispiel eines APN.*

Wie in der Abbildung ersichtlich, werden Anfangsknoten als schwarz umrandete weisse Rechtecke, Endknoten als schwarze Rechtecke und Aktivitäten als schwarze Balken dargestellt. XOR-AND- sowie AND-XOR-Aktivitäten haben zusätzlich Fahnen, XOR-XOR- sowie AND-AND-Aktivitäten sind Zusammensetzungen von XOR-AND- und AND-XOR- resp. AND-XOR- und XOR-AND-Aktivitäten. Die gerichteten Kanten definieren die Flussrelation.

Wird eine Untermenge $rAPN = \langle A, AX, XA, I, O, F \rangle$ gebildet, und gilt die Restriktion, dass eine doppelte Aktivität mit mehreren Eingängen nur einen Ausgang, eine mit mehreren Ausgängen nur einen Eingang besitzt, handelt es sich um ein *reduziertes APN*. Abb. 5.3 ist das zu Abb. 5.2 gehörende reduzierte APN. Diese Variation ist zugunsten der Kriterien Minimalität und Uniformität (vgl. Abschnitt 5.6) und kann ein Modell leichter verständlich machen, geht jedoch zu Lasten von dessen Kompaktheit.

Die syntaktische Definition lässt sich wie folgt zusammenfassen:

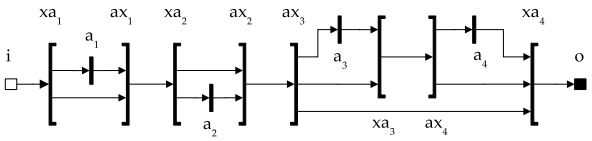


Abbildung 5.3: *Das Beispiel aus Abb. 5.2 als reduziertes APN.*

1. Die Elemente eines APN sind Anfangs- und Endknoten, Aktivitäten sowie gerichtete Kanten.
2. Eine gerichtete Kante manifestiert einen Ausgang eines Anfangsknotens oder einer Aktivität und einen Eingang eines Endknotens oder einer Aktivität.
3. Ein Anfangsknoten hat genau einen Ausgang und keinen Eingang.
4. Ein Endknoten hat genau einen Eingang und keinen Ausgang.
5. Eine Aktivität hat mindestens einen Eingang und mindestens einen Ausgang.
6. Eine Aktivität ist dann einfach, wenn sie genau einen Eingang und genau einen Ausgang hat.
7. Eine Aktivität mit mindestens einem Eingang und mehr als einem Ausgang oder mit mindestens einem Ausgang und mehr als einem Eingang ist doppelt.
8. Ein APN besteht aus mindestens einem Anfangsknoten, mindestens einem Endknoten und mindestens einer Aktivität[1].

Aus 2, 3, 4 und 5 folgt, dass jede Aktivität und jeder Anfangs- und Endknoten Teil der Flussrelation ist, d. h., es gibt keine Aktivitäten, deren Ein- und Ausgänge nicht mit einer gerichteten Kante versehen sind, wie es keine Anfangsknoten ohne gerichtete Kante beim Ausgang und keine Endknoten ohne gerichtete Kante beim Eingang gibt.

5.2.2 Abstraktion, Dekomposition

Im APN aus Abb. 5.2 lässt sich aus zusammenhängenden Aktivitäten ein Subnetz bilden, wie es in Abb. 5.4 a) dargestellt ist. Dieses Subnetz kann anschliessend durch eine Aktivität ersetzt (abstrahiert) werden, dies zeigt Abb. 5.4 b). Falls das Subnetz über je einen Eingang und einen Ausgang verfügt, kann es durch eine einfache Aktivität ersetzt werden. Wenn es mehrere Eingänge oder

[1] und folglich mindestens 2 gerichteten Kanten

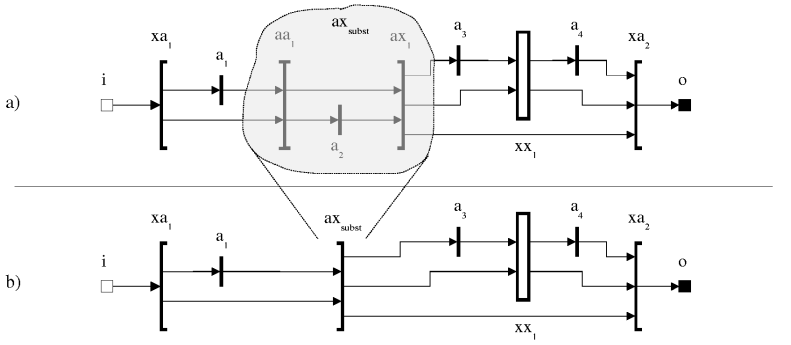


Abbildung 5.4: *Abstraktion und Dekomposition.*

Ausgänge besitzt, wird es durch eine doppelte Aktivität zu ersetzen sein. Werden alle Aktivitäten eines APN zu einer Aktivität zusammengefasst, entsteht das einfachstmögliche APN, bestehend aus einem oder mehreren Anfangs- und Endknoten sowie einer Aktivität.

Ein Subnetz muss derart gewählt werden, dass eine einzige Aktivität dessen Struktur nach aussen übernehmen kann. Für die in Abb. 5.5 a) und b) gezeigten Subnetze ist dies der Fall, für das Subnetz c) dagegen nicht. Ist ein Subnetz wohlgeformt (vgl. 5.4.1), wie das in Abb. 5.5 b) gezeigte, kann es in jedem Fall durch eine einfache Aktivität ersetzt werden.

Eine Aktivität lässt sich in weitere Aktivitäten aufspalten (dekomponieren). Wenn es sich dabei um eine abstrahierte Aktivität wie im Beispiel oben handelt, resultiert daraus das ursprüngliche Subnetz.

5.2.3 Aggregierung und Fragmentierung

Anfangsknoten eines APN lassen sich mit Endknoten eines weiteren APN zu einem neuen APN aggregieren, untereinander verbundene Anfangs- und Endknoten verschmelzen dabei. Umgekehrt kann ein bestehendes APN in zwei oder mehr kleinere APN fragmentiert werden, indem Kanten durchtrennt werden und diese mit Anfangs- und Endknoten versehen werden. Beides ist in Abb. 5.6 illustriert.

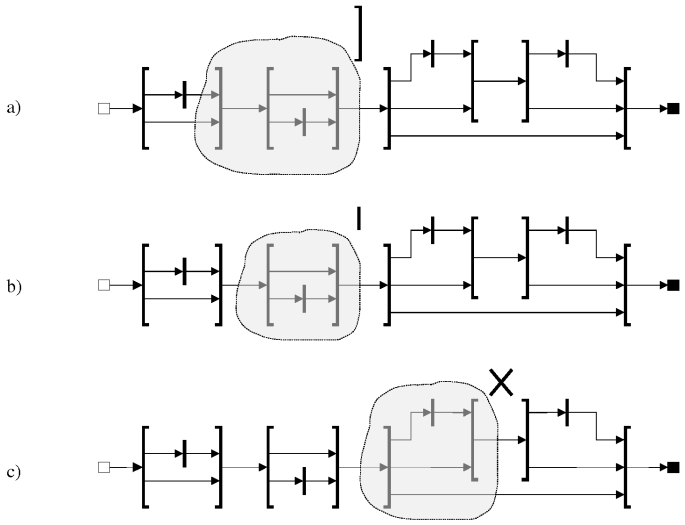


Abbildung 5.5: *Wahl gültiger Subnetze. Die Subnetze in a) und b) lassen sich durch die angedeutete Aktivität ersetzen, das in c) gezeigte Subnetz hingegen ist offen und kann nicht abstrahiert werden.*

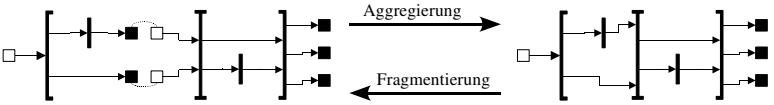


Abbildung 5.6: *Aggregierung und Fragmentierung.*

5.3 Semantik

APN haben keine Zustandsinformation, sie sind statisch. Damit jedoch das Verhalten der einzelnen Elemente und damit die Bedeutung eines APN verstanden werden kann, ist es notwendig, APN gedanklich so zu erweitern, dass sie auch Informationen über ihre Zustände beinhalten. Dazu wird im folgenden Abschnitt vorgängig ein Konzept eingeführt, welches APN um eine imaginäre Dynamik erweitert und die Zustandsänderungen unter Zuhilfenahme des Markenprinzips von Petri-Netzen ermöglicht. Dies geschieht im übrigen in Analogie zu UML 2.

5.3.1 Imaginäre Kennzeichnung der dynamischen Abläufe

In den in Kapitel 4 vorgestellten Petri-Netzen wird der Zustand eines Petri-Netzes mittels Marken beschrieben. Abhängig vom Typ, sind diese Marken anonym oder gefärbt, kann ihre Anzahl pro Stelle höchstens eins oder eine gegebene Konstante sein. Die Schaltregeln der Petri-Netze beruhen auf den möglichen Zustandsänderungen und lassen sich durch das „Markenspiel“ beschreiben. Dieses besagt im Falle der elementaren Petri-Netze, dass eine Aktivität dann freigeschaltet ist, wenn sämtliche vorgelagerten Stellen mit einer Marke belegt und sämtliche nachgelagerten Stellen frei sind. Eine Zustandsänderung ist dann erfolgt, wenn die Aktivität geschaltet hat. Dies wirkt sich dadurch aus, dass die Marken in den vorgelagerten Stellen entfernt und die nachgelagerten Stellen mit je einer Marke belegt sind.

Wird dieses Markenkonzept gedanklich auf APN übertragen, lassen sich Zustände und Zustandsänderungen eines APN beschreiben. Dies geschieht dadurch, dass auf mindestens eine gerichtete Kante eines APN eine imaginäre Marke, nachfolgend IM genannt, gelegt wird, wobei jede Kante höchstens eine Marke aufnehmen kann. Dieser Schritt bestimmt durch Anzahl und Position der Marken den Anfangszustand des APN und wird Initialisierung genannt. Die Kante entspricht der Stelle in Petri-Netzen. Alternativ wird bei vorgelagerten Kanten von Eingängen und bei nachgelagerten Kanten von Ausgängen gesprochen.

Es gibt einen wichtigen Unterschied zwischen den Petri-Netzen und APN: Während bei Petri-Netzen durch das Vorhandensein einer Marke die Aktivität schalten kann, aber nicht schalten muss, ist bei APN die IM der Auslöser für

ein unmittelbares Schalten. Das Schalten selbst dagegen kann eine gewisse Zeit in Anspruch nehmen.

Wie bei Petri-Netzen werden Marken auf vorgelagerten Stellen von den Aktivitäten beim Schalten konsumiert und auf nachfolgenden Stellen Marken produziert.

5.3.2 Konnektoren

Konnektoren dienen der Verbindung des Kontrollflusses an den Schnittstellen. Als Schnittstelle gelten nicht nur die Übergänge von APN nach fremden Systemen, sondern auch die Übergänge innerhalb eines APN, wie sie auf Kontrollflussebene dort enstehen, wo das Modell unvollständig ist. Diese Lücken entstehen dadurch, dass das Modell im Entstehen begriffen ist oder aus Gründen der Übersichtlichkeit nur Teile des Modells präsentiert werden. Definiert werden diese Schnittstellen mittels Anfangs- und Endknoten. Ihre alleinige Aufgabe ist die Kennzeichnung einer Verknüpfung untereinander, wobei ein Anfangs- und ein Endknoten mit selbem, eindeutigem Identifikator jeweils ein Paar bilden.

Schnittstellen im klassischen Sinne, d. h. Übergänge von APN in fremde Systeme, werden ebenfalls mittels Anfangs- und Endknoten realisiert. Hier hängt die Art der Kennzeichnung und die Paarbildung von Eingangs- und Ausgangsknoten auch vom fremden System ab und kann von APN alleine nicht beantwortet werden.

5.3.3 Aktivitäten

Aktivitäten sind Elemente, die den Kontrollfluss steuern. Daneben können sie weitere Aufgaben übernehmen, wie sie im Abschnitt 5.4 weiter unten aufgeführt sind. Ihre Funktionsweise soll anschaulich an Hand der oben eingeführten IM erklärt werden.

Eine Aktivität in APN ist systemtheoretisch betrachtet ein Verbund von zwei Funktionen. Dies drückt sich in der zweigeteilten Bezeichnung von Aktivitäten aus. So bezieht sich das *AND* in der Bezeichnung der AND-XOR-Aktivität auf die Eingangsfunktion, die einem logischen UND entspricht, und das *XOR* bezeichnet die Ausgangsfunktion, die einem logischen Exklusiv-Oder entspricht, indem sie eine Entscheidung für genau einen Ausgang trifft.

Die Eingangsfunktion überwacht alle Eingänge auf das Vorhandensein von IM und benachrichtigt in Abhängigkeit ihrer Logik die Ausgangsfunktion, welche ihrerseits das Plazieren von IM auf die Ausgänge übernimmt (vgl. das Beispiel in Abb. 5.7). Die IM fungieren also als Eingangsgrösse und als Ausgangsgrösse. Da sie rein gedanklicher Natur sind, wird keine eigentliche Ausgabe erzeugt.

Damit lassen sich die Verhaltensweisen der verschiedenen Aktivitäten folgendermassen beschreiben:

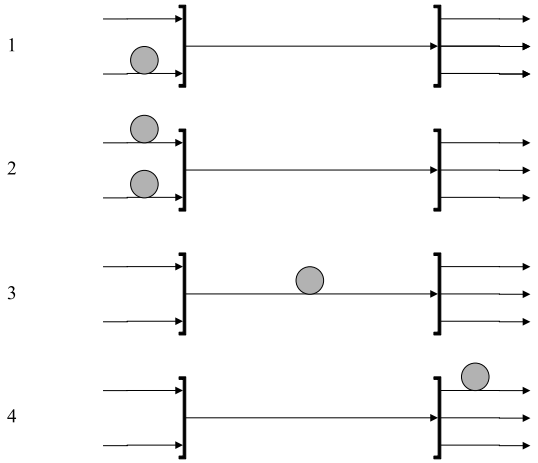


Abbildung 5.7: *Ablauf bei Eintreffen der imaginären Marken (IM), dargestellt als ausgefüllte Kreise, am Beispiel der in die Eingangs- und Ausgangsfunktion aufgetrennten AND-XOR-Aktivität: In 1) ist die AND-Bedingung der Eingangsfunktion nicht erfüllt, da nicht alle Eingänge mit einer IM belegt sind. In 2) ist die AND-Bedingung erfüllt, und die Eingangsfunktion der Aktivität entfernt die beiden IM und benachrichtigt die Ausgangsfunktion der Aktivität, dargestellt durch die dazwischenliegende IM (3). In 4) legt die Ausgangsfunktion der Aktivität eine IM auf genau einen Ausgang und entfernt die IM zwischen Eingangs- und Ausgangsfunktion der Aktivität.*

- Eine einfache Aktivität entfernt eine IM auf ihrem Eingang und belegt ihren Ausgang mit einer IM, wenn ihr Eingang mit einer IM belegt war.
- Eine Aktivität vom Typ *AND-XOR* entfernt sämtliche IM auf den Eingängen und belegt genau einen Ausgang mit einer IM, wenn sämtliche Eingänge mit einer IM belegt waren.
- Eine Aktivität vom Typ *XOR-AND* entfernt die auf einem der Eingänge liegende IM und belegt alle Ausgänge mit je einer IM, wenn einer der Eingänge mit einer IM belegt war.
- Eine Aktivität vom Typ *AND-AND* entfernt sämtliche IM auf den Eingängen und belegt alle Ausgänge mit je einer IM, wenn sämtliche Eingänge mit einer IM belegt waren.
- Eine Aktivität vom Typ *XOR-XOR* entfernt die auf einem der Eingänge liegende IM und belegt genau einen Ausgang mit einer IM, wenn einer der Eingänge mit einer IM belegt war.

Aktivitäten mit nur einem Eingang oder Ausgang sind ein Spezialfall der obigen Aktivitäten. Dies gilt insbesondere für die einfache Aktivität, die nur einen Eingang und einen Ausgang besitzt. Um die graphische Darstellung einfach zu halten, wird diese als kurzer Balken und ohne Fahnen dargestellt.

5.3.4 Kontrollfluss

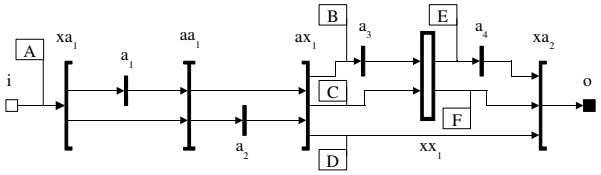
Die gerichteten Kanten legen die möglichen Routen des Kontrollflusses fest. Dieser kann unter Zuhilfenahme der IM sichtbar gemacht werden.

Zusätzlich gibt es die Möglichkeit, gerichtete Kanten zu beschriften, um auszudrücken, wie es zur Wahl einer bestimmten Route gekommen ist.

5.4 Pragmatik

Mit den Elementen von APN kann der Modellierer Prozesse beliebiger Grösse bilden. Diese Prozesse lassen sich zur Unterstützung des Modelliervorganges in Teilprozesse aufteilen und wieder zusammenführen (s. Abschnitt 5.2.3). Prozesse sind in APN strikt synonym zu Aktivitäten. Ein Prozess in seiner

höchsten Abstraktion ist eine einzelne Aktivität mit vorangehenden Anfangsknoten und nachfolgenden Endknoten. So kann bspw. eine Unternehmung als eine einzelne Aktivität betrachtet werden, die eine oder mehrere Eingaben zu einer oder mehreren Ausgaben transformiert.



Bez.	Beschreibung	verantwortlich	ausführend
A	Kundenbestellung liegt vor		
xa1	An Logistik und Buchhaltung leiten	Verkauf	Sachbearbeitung
a1	Adresse prüfen	Debibuchhaltung	Debibuchhaltung
aa1	An Transport und Warenlager leiten	Verkauf	Sachbearbeitung
a2	Rüstschein ausstellen	Warenlager	Warenlager
ax1	Verfügbarkeit prüfen	Warenlager	Warenlager
B	Teil muss erst produziert werden		
C	Teil muss nachbestellt werden		
D	Teil ist sofort lieferbar		
a3	Kunde über Lieferverzug informieren	Verkauf	Warenlager
xx1	Lieferung initiieren	Verkauf	Warenlager
E	Kunde wünscht Vorabinformation		
F	Kunde wünscht keine Vorabinformation		
a4	Kunde über Lieferung unterrichten	Verkauf	Transporteur
xa2	Teil liefern	Transporteur	Transporteur

Abbildung 5.8: *Das Beispiel aus Abb. 5.2 als modellierter Bestellvorfall.*

Ausgangspunkt einer Modellaufgabe kann das Erfassen eines (Teil-)Prozesses sein, den es in einzelne Aktivitäten aufzuspalten gilt, oder von Geschäftsvorfällen, die anschliessend zu einem (Teil-)Prozess zusammengeführt werden (vgl. Kap. 2). Für die weiteren Ausführungen wird in Abb. 5.8 an Hand eines einfachen Geschäftsvorfalls am Beispiel aus Abb. 5.2 gezeigt, wie die Modellierung eines Bestellvorgangs aussehen kann. Die Aktivitäten dieses Modells sind auf einer höheren Abstraktionsebene beschrieben; ginge es

darum, die einzelnen Arbeitsschritte abzubilden, wäre eine nachfolgende Verfeinerung im Sinne einer Dekomposition notwendig.

Der Bestellvorgang läuft wie folgt ab: An i liegt eine Kundenbestellung vor. Sodann wird die Bestellung einerseits an die Logistik durchgereicht, andererseits an die Debitorenbuchhaltung, die die Adresse überprüft und das Ergebnis anschliessend ebenfalls an die Logistik weitergibt. Diese leitet die Bestellung sowohl an den Transporteur wie an das Warenlager weiter, welches den Rüstschein ausstellt. Wenn das bestellte Teil an Lager ist, wird die Lieferung vollzogen, andernfalls wird es bestellt. Im Falle, dass das Teil nicht mehr lieferbar ist, wird der Kunde informiert, dass die Lieferung einige Tage verspätet stattfinden wird. Danach erfolgt entweder die direkte Lieferung, oder es findet vorgängig eine Information über die bevorstehende Lieferung an den Kunden statt. Damit ist der Vorgang beendet.

Je nach Zielsetzung, mit der ein Prozess modelliert wird, sind nebst der Struktur weitere Daten abzubilden. Dies geschieht in APN dadurch, dass jeder Aktivität, jeder Kante und jedem Anfangs- und Endknoten Attribute hinzugefügt werden können. Diese Attribute haben keinen bestimmten Datentyp und können beliebige Daten aufnehmen. Jedes dieser Attribute erweitert damit die Sicht auf den Prozess. In einem Geschäftsprozess können bspw. die Zuständigkeiten, die organisatorischen und geographischen Einheiten, die Durchlaufzeit, aber auch die Vorbedingungen, damit eine Aktivität erfolgreich agieren kann, und die Nachbedingungen, die eine Aktivität erfüllt, u. a. m. interessieren. Attribute lassen sich jederzeit hinzufügen, eine Taxonomie ist nicht vorgegeben, um dem Modellierer grösstmöglichen Freiraum zu geben.

5.4.1 Wohlgeformtheit

Um zu prüfen, ob die Struktur eines APN so beschaffen ist, dass der modellierte Prozess erfolgreich abläuft, bietet sich die Transformation in ein Petri-Netz mit anschliessender Analyse der Struktur und der Dynamik an. Es gibt jedoch eine einfach handhabbare Regel, deren Beachtung während der Modellierung die meisten Fallstricke vermeiden hilft: Die Wohlgeformtheit besagt, dass in einem reduzierten APN alle Ausgänge einer AND-XOR-Aktivität in eine XOR-AND-Aktivität münden resp. alle Ausgänge einer XOR-AND-Aktivität in eine AND-XOR-Aktivität münden, sofern es mehr als einen Ausgang gibt. Dazwischen kann ein beliebiges Subnetz liegen, welches wiederum

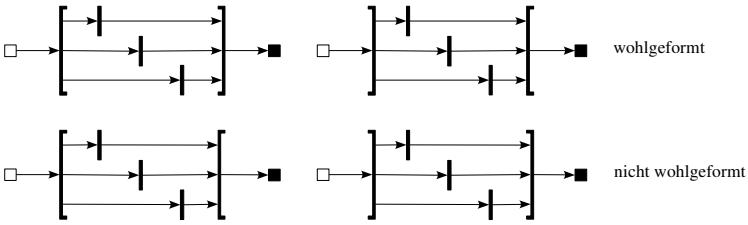


Abbildung 5.9: *Oben 2 Beispiele eines wohlgeformten APN, unten 2 Beispiele eines nicht wohlgeformten APN.*

wohlgeformt ist (s. Abb. 5.9). Folgt wie im Beispiel rechts unten der Abbildung auf eine AND-XOR-Aktivität eine weitere AND-XOR-Aktivität, würde der Prozess bei letzterer stillstehen, diese Konstellation führt zu einem Deadlock, da stets nur an einem Eingang eine IM zu liegen kommen kann.

5.5 Simulation eines APN mittels Transformation in ein Petri-Netz

Obwohl sich APN vom Aussehen von den Petri-Netzen unterscheiden, gibt es eine enge Verbindung. Ein beliebiges APN kann stets in ein Low-level-Petri-Netz[2] überführt werden, welches wiederum in ein APN rücktransformiert werden kann. Um ein beliebiges Low-level-Petri-Netz in ein APN zu transformieren, sind u. U. zusätzliche Blindtransitionen einzufügen.

APN beschreiben die Struktur, nicht aber das Verhalten eines Prozesses. Dieses Fehlen der Dynamik ist Motivation für die Transformation in ein Petri-Netz, namentlich wenn es um die Simulation eines Prozesses geht, um das Verhalten zu studieren und um Design-Fehler aufdecken zu können. Petri-Netze sind gut untersucht, und es gibt zahlreiche Algorithmen und Werkzeuge, die die Analyse eines Petri-Netzes unterstützen.

[2]Genaugenommen ist hier ein *Netz*, nicht ein Petri-Netz gemeint, da nur die Struktur, nicht aber der Zustand des Netzes überführt werden kann.

5.5.1 Vergleich von Grundkonstrukten von APN und Petri-Netzen

Um die enge Beziehung zwischen APN und Petri-Netzen zu zeigen, wird in einem ersten Schritt die AND-XOR-Aktivität von APN mit dem entsprechenden Petri-Netz verglichen. Abb. 5.10 a) zeigt die AND-XOR-Aktivität, bestehend aus zwei Eingängen und einem Ausgang, und ihre strukturelle Analogie als Petri-Netz. Das Konstrukt stellt sicher, dass der Prozess erst dann fortgeführt werden kann, wenn beide Eingänge aktiv sind[3], und kann – da nur eine Ausgangskante vorliegt – als eine synchronisierende *AND-Zusammenführung* interpretiert werden. Die beiden gerichteten Eingangskanten im APN entsprechen den zwei vorgelagerten Stellen und ihren Ausgangskanten im Petri-Netz, die Aktivität im APN entspricht der Transition, die gerichtete Ausgangskante im APN der gerichteten Ausgangskante und nachgelagerten Stelle im Petri-Netz. Abb. 5.10 b) zeigt ein weiteres Beispiel einer AND-XOR-Aktivität, diesmal mit einem Eingang und zwei Ausgängen, und ihr Pendant als Petri-Netz. Die zwei Transitionen im Petri-Netz entsprechen der AND-XOR-Aktivität im APN, die der Transition vorgelagerte Stelle und die beiden gerichteten Eingangskanten der beiden Transitionen im Petri-Netz entsprechen der gerichteten Eingangskante im APN, die beiden gerichteten Ausgangskanten der beiden Transitionen den beiden gerichteten Ausgangskanten der AND-XOR-Aktivität. Dieses Konstrukt erfordert eine Entscheidung von aussen[4], ob der Prozess auf der oberen oder der unteren Kante weiterzuführen ist, und kann – da nur eine Eingangskante vorliegt – als eine *XOR-Verzweigung* interpretiert werden.

In Abb. 5.10 c) werden die beiden Konstrukte aus a) und b) hintereinandergestellt. Werden nun in den APN nicht nur die Ausgangskante der AND-Zusammenführung mit der Eingangskante der XOR-Verzweigung, sondern die Aktivitäten der AND-Zusammenführung und der XOR-Verzweigung selber miteinander verschmolzen, entsteht die in Abb. 5.10 d) gezeigte AND-XOR-Aktivität mit mehreren Ein- und Ausgangskanten. Die Petri-Netze werden ebenfalls durch die Verschmelzung der Ausgangsstelle der AND-Zusammenführung mit der Eingangsstelle der XOR-Verzweigung zu einem neuen Petri-Netz verschmolzen.

[3] Ein Eingang ist dann aktiv, wenn die vorgeschaltete Aktivität beendet ist.

[4] Die Entscheidung muss von aussen erfolgen, es handelt sich hierbei um einen Konflikt, genauer um einen strukturellen Konflikt, da nichts über den Zustand des Systems bekannt ist.

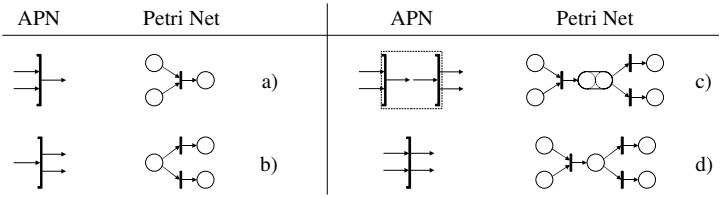


Abbildung 5.10: *Ableitung der AND-XOR-Aktivität und ihre Entsprechung als Petri-Netz.*

Abb. 5.11 gibt einen Überblick der APN-Konstrukte aus Abb. 5.1 zusammen mit deren Entsprechungen als Petri-Netz. Die AND-XOR-Aktivität aus Abb. 5.10 d) findet sich in g) wieder. Die Aktivitäten in h), j) und k) wurden auf dieselbe Weise gebildet[5].

5.5.2 Transformation eines APN in ein Petri-Netz

Abb. 5.12 a) zeigt erneut das APN aus Abb. 5.2, diesmal mit Einrahmung der einzelnen Elemente. Indem nun jedes eingerahmte Element durch das gemäss Abb. 5.11 entsprechende Petri-Netz-Konstrukt ersetzt wird, entsteht die Skizze von Abb. 5.12 b), zur besseren Sichtbarmachung der entsprechenden Konstrukte sind diese ebenfalls gerahmt. Werden nun die in einem Oval zu einem Paar zusammengefassten benachbarten Stellen zu einer Stelle verschmolzen, entsteht das in c) abgebildete Petri-Netz, welches dem APN in a) und in Abb. 5.2 entspricht. Dieses wurde zur Kennzeichnung des Status noch mit einer Marke in der Anfangstelle belegt.

5.5.3 Simulation des transformierten Geschäftsprozesses

Das Belegen einer Stelle mit einer Marke, wie im vorigen Abschnitt geschehen, kann als ein eingetroffener Geschäftsvorfall interpretiert werden. Dies kann bspw. eine Kundenanfrage oder die Beendung eines vorgelagerten Prozesses sein. Anschliessend beginnt die Simulation, indem mittels eines exter-

[5]Das Petri-Netz in j) ist zusätzlich komprimiert, indem das bei der Transformation entstehende Stellen-Transition-Paar in der Mitte fallengelassen wurde; die Semantik bleibt dieselbe.

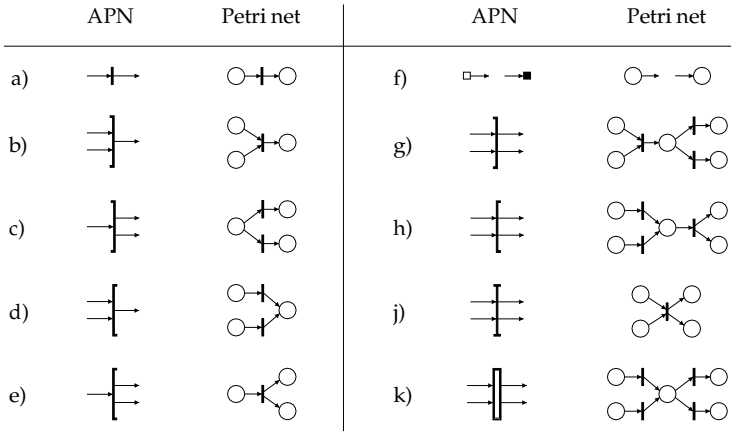


Abbildung 5.11: *Die einzelnen APN-Konstrukte und ihre Entsprechungen als Petri-Netze: a) einfache Aktivität, b) AND-Zusammenführung, c) XOR-Verzweigung, d) XOR-Zusammenführung, e) AND-Verzweigung, f) Anfang- und Endknoten, g) AND-XOR-Aktivität, h) XOR-AND-Aktivität, j) AND-AND-Aktivität, k) XOR-XOR-Aktivität.*

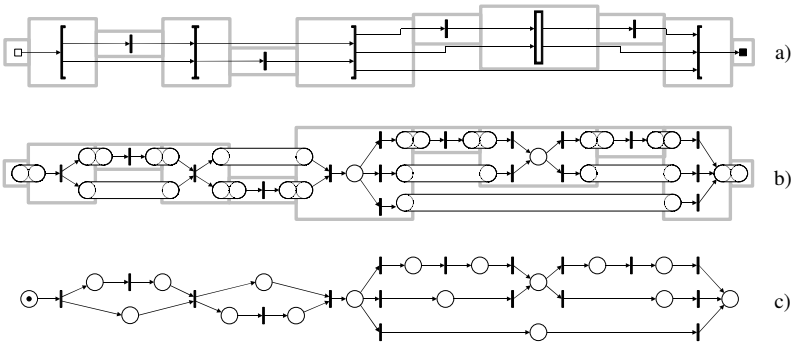


Abbildung 5.12: *Transformation eines APN in ein Petri-Netz.*

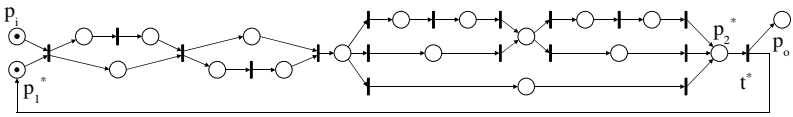


Abbildung 5.13: *Connected Petri net.*

nen Triggers die jeweils schaltbereite Transition zum Feuern veranlasst wird, bis die Stelle am Ende des Petri-Netzes mit einer Marke belegt ist, worauf der nächste Simulationslauf gestartet werden kann.

Damit die Simulationsergebnisse nicht verfälscht werden, darf erst nach Beendigung eines Simulationslaufes eine neue Simulation gestartet werden, andernfalls können „falsche“, zu einer anderen, gleichzeitig stattfindenden Simulation gehörende Marken den Ablauf verfälschen. Dies kann dadurch erreicht werden, dass Marken unterscheidbar gemacht werden, so dass sie einem bestimmten Simulationslauf zugeordnet werden können. Eine andere Möglichkeit ist die, das Petri-Netz so zu erweitern, dass die Simulationen nur sequentiell ablaufen können.

In Abb. 5.13 wurde das Petri-Netz aus Abb. 5.12 c) um die Transition t^* und die Stelle ${p_1}^*$ erweitert. ${p_2}^*$ ersetzt das ehemalige p_0. t^* ist die Rückkopplung zu ${p_1}^*$, welche gewährleistet, dass im Falle mehrerer Marken in p_i die erste Transition des ursprünglichen Petri-Netzes genau eine Marke aus p_i entfernt und somit genau ein Geschäftsvorfall gleichzeitig durchlaufen wird[6].

5.6 Bewertung

5.6.1 Abstraktionsfähigkeit

Die vertikale Abstraktionsfähigkeit ist mit der Möglichkeit der Abstraktion und Dekomposition, beschrieben in 5.2.2, gegeben. Ausschnitte der Diskurswelt lassen sich unabhängig voneinander modellieren und später mittels Kon-

[6]Das um die Rückkopplung ergänzte Petri-Netz ist ein sog. *connected* Petri net; ohne p_i und ohne p_o wäre es ein sog. *strongly connected* Petri net, in diesem Fall wegen dem Vorhandensein von ${p_1}^*$ und ${p_2}^*$ zudem ein *Workflow net*.

K 1 Abstraktionsfähigkeit	••••
K 2 Implementierbarkeit	••••
K 3 Integrationsfähigkeit	••••
K 4 Korrektheit	••••
K 5 Lesbarkeit	••[7]
K 6 Minimalität	•[8]
K 7 Uniformität	••••
K 8 Vollständigkeit	••••
K 9 Wirtschaftlichkeit	•••

Tabelle 5.1: *Die qualitative Beurteilung von APN gemäss den in Kap. 2 hergeleiteten Kriterien.* • *steht für minimal,* •••• *für maximal erfüllt.*

nektoren (s. Abschnitt 5.3.2) zusammenfügen, so dass die horizontale Abstraktionsfähigkeit ebenfalls gegeben ist.

5.6.2 Implementierbarkeit

Die Implementierbarkeit wird in Abschnitt 6.7 am Beispiel der Webshop-Anwendung esarine gezeigt. Prozesse können mit APN soweit verfeinert werden, dass für jeden Arbeitsschritt eine eigene Aktivität mit zugehörigen Attributen entsteht. Damit ist es möglich, den Kontrollfluss zusammen mit weiteren Angaben direkt vom Modell ausgehend zu implementieren.

5.6.3 Integrationsfähigkeit

Mittels Konnektoren lassen sich APN an weitere APN und an Prozessmodelle anderer Sprachen anbinden, sofern diese eine vergleichbare Schnittstelle zur Verfügung stellen. Bei Bedarf lassen sich mittels Attributierung Transformationsregeln definieren und so Schnittstellen zu fremden Modellen realisieren.

[7] nicht abschliessend geklärt, s. Text

[8] unter Anwendung zusätzlicher, einfacher Modellierungsvorschriften vollständig erreichbar

5.6.4 Korrektheit

Die einfache Formalisierung der den APN zugrundeliegenden Petri-Netzen und die eindeutige Semantik der wenigen Modellelemente von APN erleichtern dem Modellierer das Erstellen eines Modells, das den Absichten entspricht. Die für ein korrektes Modell vorausgesetzte Vollständigkeit (s. u.) ist gegeben.

5.6.5 Lesbarkeit

Die Lesbarkeit wird durch das Vorhandensein nur weniger graphischer Symbole, die jeweils eine eindeutige Semantik haben, unterstützt. Für die abschliessende Beurteilung der Lesbarkeit von APN-Modellen sind weitergehende Untersuchungen nötig.

5.6.6 Minimalität

Die Minimalität wird verletzt, eine AND-AND- oder eine XOR-XOR-Aktivität lässt sich durch Hintereinanderschalten einer AND-XOR- und einer XOR-AND- resp. einer XOR-AND- und einer AND-XOR-Aktivität substituieren. Diese Substitution führt zu einem reduzierten APN (vgl. Abb. 5.3), welches dann aber die Minimalität erfüllt.

5.6.7 Uniformität

Die Uniformität wird von einem APN-Modell erfüllt, wenn auch nicht offensichtlich. Dies sei am Beispiel der AND-XOR-Aktivität gezeigt: Im Falle eines einzelnen Eingangs schaltet die Aktivität dann, sobald der Eingang aktiviert ist. Im Falle mehrerer Eingänge schaltet die Aktivität erst dann, wenn alle Eingänge aktiviert sind. Logisch gesehen, prüft die Aktivität aber auch bei Vorhandensein von nur einem Eingang, ob alle Eingänge aktiviert sind.

5.6.8 Vollständigkeit

Mit der Möglichkeit, einer Aktivität beliebige Attribute hinzuzufügen (s. 5.4), lassen sich die für eine Aufgabenstellung relevanten Daten erfassen und auswerten. Dies sind im Falle von Geschäftsprozessen insbesondere Angaben

zu Rollen, Organisationseinheiten, Durchlaufzeiten, Tätigkeiten u. a. m. Damit lässt sich die Vollständigkeit eines APN-Modells erreichen.

5.6.9 Wirtschaftlichkeit

Die APN-Modelle erfüllen die für die Aufwandminimierung vorausgesetzten Kriterien. Die im Interesse der Flexibilität fehlende Taxonomie für die Attributierung der Elemente führt zu Beginn der Modellbildung zu einem Abstimmungsbedarf innerhalb des Modellierteams, womit die Wirtschaftlichkeit etwas beeinträchtigt ist.

Kapitel 6

Implementierung und Nutzung von APN im E-Commerce

Ursprüngliche Idee des World Wide Web (WWW) – die Übertragung von Information über das Internet unter Zuhilfenahme von Hypertext (HTML) (s. BL89) – war die Anfertigung und der Austausch von einfachen, statischen Dokumenten zwischen Forschern und Rechnern. Nach und nach wurden die kommerziellen Möglichkeiten des WWW entdeckt, und mit dem Aufkommen der ersten Auktionsplattformen konnten die Anwender eigene Inhalte ins Web stellen. Viele der heutigen Web-Anwendungen liefern Dokumente, deren Inhalte dynamisch generiert, zum Teil personalisiert und kontextabhängig sind[1].

Das Kapitel gibt eine knappe Einführung in den Electronic Commerce und in die Implementierung des ursprünglich am Lehrstuhl für Informationssysteme der Universität Freiburg i. Ue entstandenen Webshops *esarine*. Dies geschieht an Hand des Frameworks *Struts*, welches die Entwicklung von in Java geschriebenen Web-Anwendungen unterstützt. Zum besseren Verständnis wird dazu knapp auf einige grundlegende technische Aspekte – insbesondere HTTP und dessen Implikationen – eingegangen. Damit ist die Basis gelegt, um in Abschnitt 6.7 skizzenhaft zu zeigen, wie ein Modell, das die Interaktionen

[1] Je nach den gewählten Mitteln kann die personalisierte Gestaltung im Widerspruch zur Architektur des World Wide Web stehen und zu Schwierigkeiten primär auf der Anbieterseite führen; auf diesen Punkt wird in Abschnitt 6.2.1 am Rande eingegangen.

während eines Bestellvorgangs zwischen Kunde und Webshop abbildet, in die Ablaufsteuerungskomponente der Webshop-Anwendung überführt werden kann.

6.1 Electronic Commerce

6.1.1 Einordnung

Eine einheitliche Definition des Begriffs Electronic Commerce oder auch E-Commerce existiert nicht, dazu ist das Phänomen zu jung. Allgemein wird unter E-Commerce der über digitale, vielfach offene Kommunikationsnetze abgewickelte virtuelle Handel unter Unternehmen (B2B), von Unternehmen zu Privatpersonen (B2C) und unter Privatpersonen (C2C) verstanden. Electronic Commerce kann als Untermenge des Electronic Business verstanden werden; MS08 beschreiben Electronic Business als „[...] Anbahnung, Vereinbarung und Abwicklung elektronischer Geschäftsprozesse [...] zur Erzielung einer Wertschöpfung“. Electronic Commerce beschreiben sie als das Anbieten von Produkten und Dienstleistungen durch Unternehmen für Kunden (B2C) oder weitere Unternehmen (B2B).

Sind an einem elektronischen Geschäft nur ganz wenige Teilnehmer beteiligt, reicht eine einfache Kommunikation bspw. per E-Mail aus. Sollen dagegen viele Anbieter oder Nachfrager am Handel teilnehmen können, bedarf es nebst den erwähnten Kommunikationsnetzwerken einer technischen Einrichtung, die die Aufgaben eines Marktplatzes – Präsentation der Waren und Dienstleistungen, Preisfindung und Bezahlung – übernimmt. Im Idealfall ist dieser virtuelle Marktplatz neutral, d. h., er wird von einem Betreiber unterhalten, der die Interessen der Anbieter und der Nachfrager gleichermassen berücksichtigt. Beispiele neutraler virtueller Marktplätze sind Auktionsplattformen für Privatanwender (C2C) oder Börsen (B2B, B2C, C2C). Ein Beispiel eines nicht neutralen, von einem Nachfrager oder einer kleinen Gruppe von Nachfragern betriebenen virtuellen Marktplatzes (buy-side) ist die Einkaufsplattform; damit sich genügend viele Anbieter finden, die bereit sind, ihre Angebote einzureichen, braucht es jedoch eine gewisse Marktmacht seitens Nachfrager, so dass es in erster Linie grössere Konzerne, spezialisierte Abnehmer oder öffentliche Verwaltungen sind, die als Betreiber in Erscheinung

treten. Das Gegenstück dazu ist der von einem Anbieter oder einer kleinen Gruppe von Anbietern betriebene virtuelle Marktstand (sell-side).

TTL00 führen zur Charakterisierung der Erscheinungsformen von Arbeitsplattformen die *Business Webs* ein und nehmen eine Unterteilung in die fünf Typen *Agora*, *Aggregation*, *Value Chain*, *Alliance* und *Distributive Network* vor.

Der Typ Agora ist der offene, transparente Marktplatz, wo sich viele Anbieter und Nachfrager treffen. Er entspricht dem oben geschilderten neutralen virtuellen Marktplatz. Besonderes Merkmal ist die dynamische Preisfindung. Beispiele für den Typ Agora sind Wertschriftenbörsen oder Auktionen.

Beim Typ Aggregation handelt es sich um einen Händler, der die Vermittlungsfunktion zwischen Anbieter und Nachfrager übernimmt. Er bestimmt das Produkteangebot und legt die Preise fest. Beispiele sind Web-Shops oder Online-Dienstleister.

Ein Business Web vom Typ Value Chain als Integrator schafft Mehrwert für den Kunden, indem es in Eigenverantwortung die für eine massgeschneiderte Lösung erforderlichen Leistungserbringer koordiniert und dafür sorgt, dass der Abnehmer nur mit dem Integrator in Kontakt treten muss. Besonderes Merkmal ist die Bildung der Wertschöpfungskette durch den Integrator. Beispiele sind Generalunternehmer oder Systemanbieter.

Im Typ Alliance haben sich Gleichgesinnte zu einer Allianz zusammengeschlossen. Ziel ist es, auf gleichberechtigter Stufe gemeinsam eine Lösung zu finden. Besonderes Merkmal ist das gleichzeitige Liefern und Nachfragen der einzelnen Mitglieder. Beispiele sind Open-Source- oder Open-Content-Communitys.

Der Distributor als Vertreter des Typs Distributive Network ist ein Verteilkanal, der Anbieter und Nachfrager gleichermassen offensteht. Besonderes Merkmal ist die Zuweisung von Transportkapazität und Verteilung von Daten. Beispiele sind das Internet oder die Telekommunikationsanbieter.

6.1.2 Beweggründe

Motivation für den Einsatz von E-Commerce sind die Senkung der Transaktionskosten und die Erhöhung des Absatzmarktes. Ein weiterer Antrieb kann die Schaffung von Wettbewerbsvorteilen durch Innovation sein.

Der Betrieb und Unterhalt einer technischen Infrastruktur ist verglichen mit dem einer herkömmlichen Verkaufsfläche in aller Regel mit tieferen Ko-

sten verbunden: Der Verkaufsprozess von der Beratung bis zur Offerte läuft automatisiert und ohne Personalbedarf ab. Vom bescheidenen Platzbedarf der technischen Infrastruktur abgesehen, wird keine Fläche für die Warenauslage benötigt, und die durch den Kunden selber erfassten Kundendaten können ohne Umweg in das Bestellsystem übernommen und weiterverarbeitet werden. Diesem Kostenvorteil stehen die erforderlichen Investitionen in aus Sicht des Anbieters z. T. unbekannte und risikobehaftete Technologien gegenüber. Die Auswahl von Dienstleistern fällt gerade kleineren Anbietern besonders schwer, da sie oft nicht über das dazu erforderliche technische und organisatorische Wissen verfügen.

Sowohl die geographische wie die zeitliche Ausdehnung des Marktes erlauben die Ausdehnung des Absatzes für versandfähige oder digital übertragbare Güter und über Distanz erbringbare Dienstleistungen. Voraussetzung dazu ist, dass es gelingt, die potentielle Käuferschaft auf das eigene Angebot aufmerksam zu machen. Zu beachten sind ferner das Delkredererisiko und der Umstand, dass in den Aufbau einer Versandabteilung investiert werden muss.

Dank den elektronisch vorliegenden, laufend nachgeführten Daten liefert das Berichtswesen stets aktuelle Kennzahlen, so dass auf Veränderungen im Markt rechtzeitig reagiert werden kann. Zudem lassen sich mit minimalem Zusatzaufwand neue Funktionalitäten implementieren, die bspw. das Cross-Selling[2] unterstützen, indem dem Kunden beim Abschluss des Kaufvertrages automatisiert ein Zusatzangebot gemacht wird.

Es bleibt anzumerken, dass die Kostenseite differenziert betrachtet werden muss. Kurz- und mittelfristig ist auf Grund hoher Marketing- und Logistikaufwendungen mit erheblichen Kosten zu rechnen. Amazon bspw. kam sieben Jahre nach seiner Gründung erstmals aus seiner Verlustzone[3].

6.1.3 Phasen des Electronic Commerce

Der Wertschöpfungsprozess des Electronic Commerce lässt sich in die Hauptphasen *Information*, *Vereinbarung* und *Abwicklung* aufteilen. Vor die Informa-

[2] Verkauf ergänzender Produkte oder Dienstleistungen

[3] S. Fra99. Die Geschäftsberichte der letzten 10 Jahre sind unter http://phx.corporate-ir.net/phoenix.zhtml?c=97664&p=irol-reportsAnnual abrufbar, letzter Zugriff am 23. August 2007.

tionsphase kann sich die Anregungsphase gesellen, die Abwicklungsphase kann von einer Kundenbetreuungsphase gefolgt werden (SW02).

6.1.3.1 Informationsphase

Die Informationsphase ist vom Wissensstand des potentiellen Kunden geprägt. Je nach Wissensstand bzgl. Produkt und Anbieter wird sich der Interessent verschiedener Techniken bedienen, bis er über genügend Informationen zu Produkt und Anbieter verfügt und im Quadranten I in Abb. 6.1 angelangt ist, um in die nachfolgende Vereinbarungsphase überzutreten – ohne sich für ein Produkt und einen Anbieter entschieden zu haben, ist keine Vereinbarung möglich.

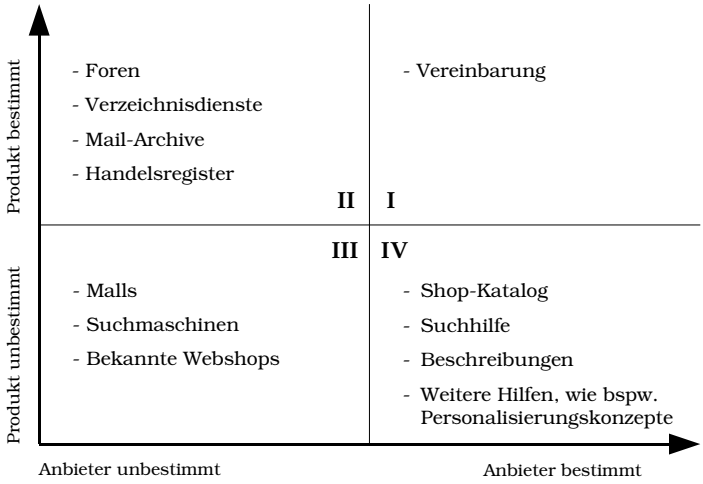


Abbildung 6.1: *Die verschiedenen Stadien der Informationsphase und die Werkzeuge, um den Informationsstand zu erhöhen. Um in die Vereinbarungsphase zu gelangen, muss Klarheit bzgl. Produkt und bzgl. Anbieter herrschen.*

Der Webshop-Betreiber (Anbieter) kann den Kaufentscheid des Interessenten unmittelbar im Quadranten IV beeinflussen. Mittel dazu sind vertrauensbildende Massnahmen, wie Angaben zum Unternehmen (Impressum) und zu

den Geschäftsbedingungen, ein attraktives Warenangebot, bestehend aus aktuellem Produktekatalog mit Angaben zur Verfügbarkeit, Abbildungen, Video und Audio, Suchhilfen, detaillierter Beschreibung, Cross-Selling-Funktionalität, Möglichkeiten der Interaktion via E-Mail oder Chat u. a. m., und eine einfach verständliche, intuitive Navigation. Den Quadranten III kann der Anbieter mittelfristig durch Steigern seiner Webpräsenz beeinflussen, bspw. durch die Einmiete in ein virtuelles Einkaufszentrum (Shopping Mall), den Eintrag in Webkataloge, um von Suchmaschinen prominenter gelistet zu werden, oder das Schalten von Online-Werbung. Der Quadrant II kann nur langfristig beeinflusst werden, indem bspw. auf eine gute Reputation geachtet wird.

6.1.3.2 Vereinbarungsphase

In der Vereinbarungsphase wird der Vertrag durch gegenseitige übereinstimmende Willenserklärung geschlossen. Durch das in der Informationsphase vom Anbieter vorgelegte Angebot ist ein grosser Teil der Willenserklärung seitens Anbieter bereits da, ausstehend sind Fragen zu Lieferort und Lieferart, Versandkosten, Mengenrabatte, Garantien u. a. m., die in der Vereinbarungsphase geklärt werden, nachdem der Kaufinteressent Angaben zur Menge der gewünschten Artikel und zu seiner Person gemacht hat. Abgeschlossen wird die Vereinbarungsphase im einfachsten Fall durch eine Auftragsbestätigung per E-Mail vom Anbieter an den Besteller, nachdem dieser seine Kaufabsicht per Knopfdruck bestätigt hat. Eine beglaubigte Kaufabsichtserklärung per elektronischer Signatur, die bspw. das schweizerische Recht seit Anfang 2005 anerkennt (Bun08, Art. 14, Abs. 2 bis und Art. 59a) wird zur Zeit mangels deren Verbreitung im Electronic Commerce nicht verlangt.

6.1.3.3 Abwicklungsphase

Mit der Informations- und der Vereinbarungsphase ist eine Obligation entstanden, die in der Abwicklungsphase erfüllt wird. Die Zahlung ist zu leisten, bei Sachgeschäften die Lieferung, bei Dienstleistungen die Tätigkeit vorzunehmen, bei Werkverträgen das Werk zu erbringen.

Während die Informations- und die Vereinbarungsphase i. d. R. elektronisch erfolgt, geschieht die Geschäftserfüllung meist noch auf herkömmlichem Weg: Die Bezahlung erfolgt per Kreditkarte, gegen Rechnung, Nach-

nahme usw., die Lieferung im Falle von physischen Gütern auf Grund deren Natur auch künftig per Kurier oder Bahn, bei digitalen Gütern dagegen oft elektronisch.

6.1.4 Aktuelle Entwicklungen

Aktuelle Entwicklungen gehen dahin, die Kunden vermehrt in einzelne Funktionsbereiche des Shops aktiv miteinzubeziehen. Es gibt Angebote, die von Kunden selber umschrieben und beworben werden. Andere Angebote erlauben es Besuchern, eigene Artikel oder selbstgestaltete Artikel wie bspw. T-Shirts im Shop einzustellen und gegen einen Erlösanteil über den Shop verkaufen zu lassen.

Mit dem Schlagwort Web 2.0 verlagert sich einerseits die Systemarchitektur von der monolithischen und zentral kontrollierten Applikation hin zu einer heterogenen Ansammlung vieler kleiner Teilapplikationen, deren Gesamtheit als eine Plattform betrachtet werden können (vgl. Tim05); andererseits wird die zentral vom Anbieter kontrollierte Verwaltung des Inhalts weitgehend durch eine aktive Mitgestaltung der Anwender abgelöst. Eine Übersicht zu aktuellen Trends findet sich in SF08.

6.2 Framework-gestützte Entwicklung von Web-Anwendungen

Web-Anwendungen unterscheiden sich von herkömmlichen Anwendungen dadurch, dass sich Geschäftslogik und Daten auf einem entfernten Rechner befinden und der Anwender über eine Internet-Verbindung lesenden und teilweise schreibenden Zugriff auf die Daten hat. Als Benutzerschnittstelle dient ein Webbrowser, und zum Transport der Daten und der Befehle zu deren Anzeige und Bearbeitung wird HTTP eingesetzt[4]. Dieser Aufbau beinhaltet einige Besonderheiten, auf die in den folgenden zwei Abschnitten kurz eingegangen wird.

[4] Auf Varianten, wie bspw. Teile der Geschäftslogik auf den Rechner des Anwenders zu verlagern, wird hier nur am Rande eingegangen.

6.2.1 HTTP

Das World Wide Web als einer von mehreren Diensten im Internet baut in seiner Urform auf den drei Elementen *Uniform Resource Identifier (URI)* als Identifikator, *Hypertext Markup Language (HTML)* als Seitenbeschreibung und *Hypertext Transfer Protocol (HTTP)* als Kommunikationsschema auf.

Das *Hypertext Transfer Protocol* regelt den Datenaustausch einer Ressource zwischen Server und Client. Sind die vom Client angeforderten Daten einer Ressource übertragen, wird die Verbindung zwischen Client und Server geschlossen. Im Falle einer Web-Seite, die aus vielen einzelnen Bildern zusammengesetzt ist, würden ebensoviele Verbindungen aufgebaut, was zu einer merklichen Verzögerung führen würde. Daher kennt die aktuelle Version 1.1 einen zustandsbehafteten Dateitransfer, der solange anhält, bis sämtliche Ressourcen einer Web-Seite übertragen sind. Die grundsätzliche Zustandslosigkeit des Protokolls ist eine bewusste Entscheidung, um den Server von der Last, viele u. U. gleichzeitige Verbindungen verwalten zu müssen, zu befreien, im Falle eines stark frequentierten Web-Servers eine grosse Entlastung und damit ein Beitrag an die Stabilität des Dienstes.

Zum Verständnis der Funktionsweise von Web-Anwendungen sei hier der HTTP-Request-Response-Zyklus kurz erläutert. Ein HTTP-Request setzt sich aus einer Methode, einem URI, einem Header und einem Body zusammen. Der Client setzt einen Request ab, typischerweise durch Eintippen eines URI im Adressfeld des Browsers. Dieser gelangt an den Web-Server, welcher ihn direkt bearbeitet oder zur Bearbeitung an weitere Rechner sendet, die ihrerseits wiederum eine Weiterleitung an andere Rechner vornehmen können. Nach der Bearbeitung schickt der Web-Server das Ergebnis als Response an den Client zurück.

HTTP in der aktuellen Version 1.1 kennt die Methoden *OPTIONS*, *GET*, *HEAD*, *POST*, *PUT*, *DELETE*, *TRACE* und *CONNECT*, wovon die wichtigsten vier im folgenden kurz beschrieben werden:

- GET: Liefert die Repräsentation der vom URI bezeichneten Ressource; falls der URI nicht auf eine Datei, sondern auf einen Prozess, der dynamisch eine Datei erzeugt, zeigt, wird die erzeugte Datei und nicht der Quell-Code des Prozesses zurückgeliefert.

- DELETE: Löscht die vom URI referenzierte Ressource.

- POST: Sendet Daten zu deren Weiterverarbeitung zum Web-Server; damit werden üblicherweise die ausgefüllten Felder in einem Formular übermittelt.
- PUT: Speichert oder modifiziert eine Ressource unter dem angegebenen URI.

Auf Basis obgenannter vier Methoden lassen sich vollwertige Web-Anwendungen erstellen. Die Zustandslosigkeit von HTTP kann im Falle von individuell ausgerichteten Web-Anwendungen zu gewissen Schwierigkeiten führen, die mit verschiedenen Techniken und damit einhergehend unterschiedlichen Vor- und Nachteilen lösbar sind. Dies sei am Beispiel des in einer Webshop-Anwendung typischerweise vorkommenden Warenkorbs illustriert, der nach und nach mit Waren befüllt und beim Bezahlvorgang geleert wird: Die Verwaltung des Warenkorbs kann zentral pro Kunde auf dem Applikations-Server geschehen (sog. Server-side state management), was bei einigen hundert gleichzeitig anwesenden Kunden zu einem erheblichen Ressourcenbedarf auf Server-Seite führt, oder die Verwaltung geschieht auf dem jeweiligen Rechner eines Kunden (sog. Client-side state management), so dass die Ressourcenbelastung aufgeteilt und der Server entlastet wird[5].

6.2.2 Aufbau und Funktionsweise von Web-Anwendungen

Die Kommunikation des Webbrowsers findet mit dem HTTP-Server (Web-Server) statt. Dieser verfügt über die in 6.2.1 aufgeführten Methoden[6]. Nebst der Auslieferung von HTML-Dokumenten und weiteren Ressourcen verfügt ein Web-Server über weitere Funktionalitäten, die es ihm erlauben, Script-Sprachen zu interpretieren und auszuführen oder weitere Server wie Applikations- und Datenbank-Server anzusprechen.

Abb. 6.2 zeigt schematisch den Daten- und Kontrollfluss vom Aufruf durch den Benutzer bis zur zurückgelieferten Antwort. Bei einer Benutzeranforderung an den Web-Server wird eine Programminstanz erzeugt, welche Daten

[5] zu Vor- und Nachteilen und zum Widerspruch des Server-side state management zum Web s. Fie00

[6] I. d. R. sind in einem produktiven Web-Server aus Sicherheitsüberlegungen standardmässig nicht alle Methoden aktiviert. OPTIONS, GET, HEAD und POST dürfen jedoch als unterstützt erwartet werden.

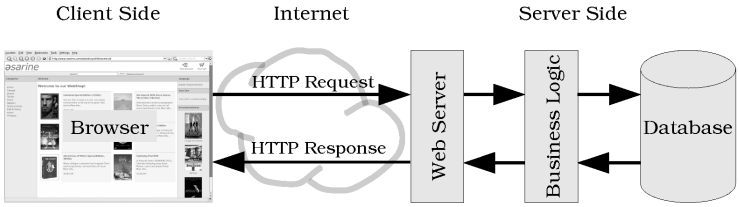


Abbildung 6.2: *Daten- und Kontrollfluss in einer Web-Anwendung.*

aus der Datenbank anfordert und anschliessend diese Daten in ein HTML-Dokument bettet, welches vom Web-Server ausgeliefert wird. Mit demselben Mechanismus können umgekehrt vom Benutzer mittels Web-Formular an den Web-Server übergebene Daten in die Datenbank geschrieben werden.

Das Erzeugen einer Programminstanz bei jedem Request zieht einen hohen Ressourcenbedarf nach sich, der sich in einer ausserordentlich langen Antwortzeit äussern kann. Dies hat dazu geführt, dass der Web-Server mit Hilfe von Erweiterungsmodulen selber um die Fähigkeit, Scriptsprachen zu interpretieren, erweitert wurde. Auch hier ist die Geschwindigkeit dadurch begrenzt, dass mit jeder Anfrage der Code neu interpretiert werden muss, was mit Cache-Mechanismen etwas verbessert werden kann.

Einen anderen Ansatz verfolgen die *Java Servlets*. Servlets sind kleine, in Java geschriebene Code-Stücke, die in einem sog. Servlet-Container ablaufen. Der Servlet-Container wiederum ist mit dem Web-Server über eine Kommunikationsleitung verbunden – meist über das TCP/IP-Protokoll. Dadurch, dass eine Instanz eines Servlets mehrere HTTP-Requests gleichzeitig abarbeiten kann, macht diese Technik sehr schnell. Java Servlets kommen im unten vorgestellten Framework Struts zum Zuge.

Damit eine Anwendung wartbar bleibt und damit sich die am Entwicklungsprozess beteiligten Personen auf ihre jeweiligen Kernkompetenzen konzentrieren und unabhängig voneinander die Implementierung vorantreiben können, braucht es ein Anwendungsschema, welches die Trennung in einzelne Teilbereiche unterstützt. Von Reenskaug stammt ein Vorschlag des ur-

sprünglich „Model-View-Editor“ (Ree79c) und später „Model-View-Controller“ (Ree79a) genannten Schemas, welches Eingang in die Klassenbibliothek von Smalltalk-80 fand (s. Ree79b).

Das *Model-View-Controller-Schema (MVC-Schema)* unterscheidet zwischen den drei Systemkomponenten *Model*, *View* und *Controller* (s. Abb. 6.3). Die Benutzereingabe, das Model in Form von Daten und die visuelle Rückmeldung an den Benutzer werden von drei Komponenten verwaltet (Bur92, GHJV95, Cav03).

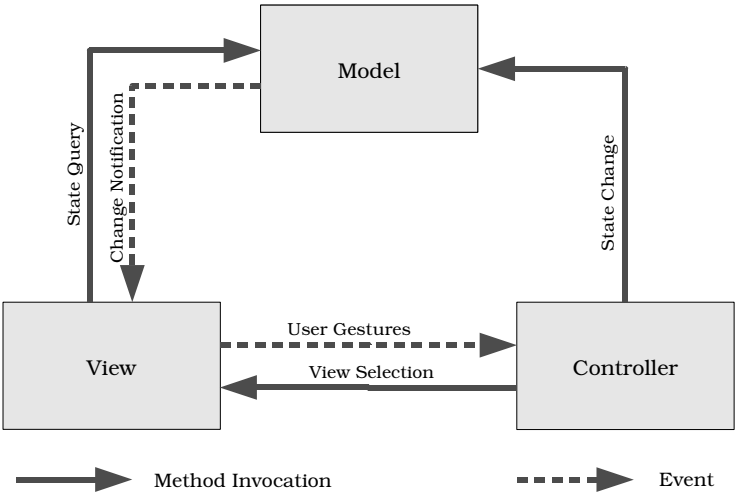


Abbildung 6.3: *Das Model-View-Controller-Schema.*

Die View ist für die benutzergerechte Darstellung des Model in Form von Text und Graphiken zuständig. Je nach Architektur kann sie sich beim Instantiieren beim Model anmelden, um laufend über Zustandsänderungen des Model informiert zu werden, alternativ kann der Controller diese Aufgabe übernehmen.

Der Controller ist für die Ablaufsteuerung zuständig, er reagiert auf die vom Benutzer kommenden Anforderungen und ruft die zuständigen Instanzen zur Aktualisierung des Model auf. Falls die View nicht automatisch vom Model über die Zustandsänderung informiert wird, stösst er zusätzlich die Anzeige des aktualisierten Model durch die View an.

Das Model schliesslich verfügt über das für die Pflege der Daten notwendige Geschäftswissen, gibt auf Anweisung über seinen Zustand Auskunft und reagiert auf angeforderte Zustandsänderungen, die normalerweise vom Controller kommen.

Die drei Komponenten lassen sich bei sauberer Implementierung einzeln auswechseln. Auch ist es möglich, einer Komponente Funktionen hinzuzufügen, ohne die beiden anderen signifikant anpassen zu müssen. Bspw. kann eine View, die Zahlen tabellarisch darstellt, um eine graphische Komponente erweitert werden, so dass die Zahlen zusätzlich als Diagramm angezeigt werden. Dazu ist weder am Model noch am Controller eine Änderung nötig.

6.3 Struts und dessen zentrale Steuerungsdatei struts-config

Struts ist ein Framework, das die Entwicklung von auf Servlets und JSP basierenden Web-Anwendungen beschleunigt, indem es wiederkehrende Funktionen vorimplementiert hat und ein Basisfundament für eigene Anwendungen bildet. Der Entwickler kann sich so auf die eigentlichen Geschäftsfunktionen konzentrieren. Struts ist ein Projekt der Apache Software Foundation, wird laufend weiterentwickelt und ist überaus populär, viele aktuelle Web-Anwendungen sind auf Basis von Struts erstellt.

Ein *Framework* besteht aus Klassen, die untereinander kooperieren und auf ein bestimmtes Anwendungsgebiet – im Falle von Struts auf Servlet-basierte Web-Anwendungen – ausgerichtet sind (vgl. Deu89, GHJV95, Cav03). Das Framework stellt abstrakte Klassen zur Verfügung, auf Basis derer der Entwickler die für sein Problem spezifischen konkreten Subklassen bildet und so die Basis für seine Anwendung legt. Das Framework legt die Architektur fest und ist für die Steuerung der Ablauflogik zuständig. Im Gegensatz zu einer Software-Bibliothek bestimmt das Framework die Architektur und die Steuerung der Ablauflogik, der Entwickler muss sich Gedanken zu den Zuständigkeiten der einzelnen Klassen machen, nicht aber zu deren Steuerung. Nicht die zu implementierende Anwendung steuert den Kontrollfluss, sondern die

im Framework vordefinierten Klassen[7]. Der Programmierer muss sich nicht um die Aufrufe kümmern, sondern dafür besorgt sein, dass bei einem Aufruf seitens Framework die verantwortliche Klasse bereit ist, den Aufruf korrekt zu verarbeiten.

Struts implementiert das in Abschnitt 6.2.2 vorgestellte Model-View-Controller-Schema und unterstützt damit die Aufteilung in die Präsentation, die Geschäftslogik und den Kontrollfluss. Der Controller, als Servlet implementiert, steuert den Anwendungsfluss. Das Model besteht aus JavaBeans oder EnterpriseJavaBeans, ist für die Geschäftslogik zuständig und kapselt den Anwendungszustand. Die View schliesslich nimmt das Ergebnis mittels JSPs entgegen und bereitet es zur Ausgabe im Browser auf. Im Detail sieht die Zusammenarbeit wie folgt aus:

Das *ActionServlet* als Teil des Controllers empfängt die Anfrage des Anwenders und übergibt sie dem *RequestProcessor*. Der RequestProcessor entscheidet an Hand des aufgerufenen URI und der Konfigurationsdatei *struts-config.xml*[8], welche *Action*-Klasse für die Aufgabe zuständig ist.[9] Die Action-Klasse ist das Bindeglied zum Model, sie führt die am Model auszuführenden Operationen mit den Parametern, die sie mittels einem *ActionForm* erhalten hat, aus. Zusätzlich übernimmt sie bei Bedarf weitere Aufgaben, wie die Authorisierung, die Protokollierung oder die Validierung der aktuellen Sitzung. Nach der Ausführung gibt die Action-Klasse ein *ActionForward*-Objekt zurück, welches dem Controller erlaubt, mittels Nachschlagen in der Datei struts-config.xml die an den Aufrufer zurückzuliefernde Seite zu bestimmen.

Abb. 6.4, entnommen aus Dav01, zeigt den Ablauf nach einem HTTP-Request in graphischer Form.

Damit ist die Funktionsweise einer auf dem Framework *Struts* aufbauenden Anwendung in groben Zügen gezeigt. Insbesondere ging es darum, die Rolle der zentralen Konfigurationsdatei *struts-config.xml* zu zeigen, welche es der Instanz *ActionServlet* erlaubt, die Abläufe zu steuern. Diese Abläufe sind

---

[7] Dies wird auch als die „Inversion of Control“ oder das „Hollywood Principle“ bezeichnet (s. Vli96), welches „Don’t call us; we’ll call you“ lautet.

[8] Die Datei kann ohne weiteres umbenannt werden, doch wird in dieser Arbeit stets unter diesem Namen auf die Datei verwiesen.

[9] Ursprünglich wurde die HTTP-Anforderung vom ActionServlet selber bearbeitet. Mit Struts Version 1.1 ist der RequestProcessor eingeführt worden, um dem Entwickler mehr Flexibilität zu geben, indem er den Verarbeitungsprozess mittels Subclassing bei Bedarf modifizieren kann.

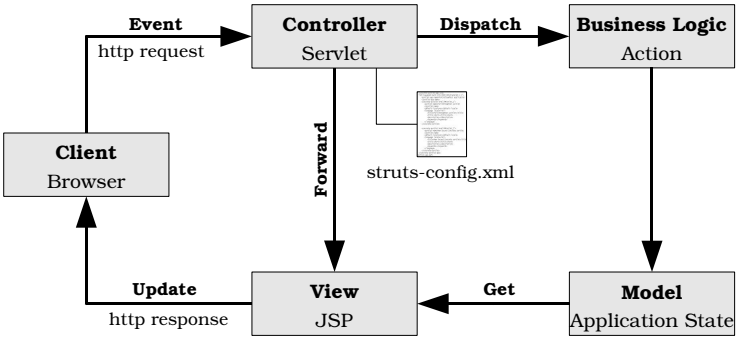


Abbildung 6.4: *Implementierung des MVC-Schemas durch Struts. Nach Dav01.*

sowohl menschen- wie maschinenlesbar in der Datei *struts-config.xml* festgehalten[10].

Im folgenden Abschnitt 6.4 werden die wichtigsten Funktionalitäten eines klassischen Webshops vorgestellt. Deren Implementierung zu einer lauffähigen Anwendung wird exemplarisch in den weiteren Abschnitten gezeigt: In Abschnitt 6.7 wird die Kundenbestellung als APN modelliert und in die Controller-Datei struts-config.xml überführt. Abschnitt 6.8 zeigt an Hand der Web-Anwendung *esarine*, wie eine konkrete Implementierung aussehen kann.

6.4 Grundfunktionalitäten einer E-Commerce-Applikation

Hauptaufgabe einer E-Commerce- oder Webshop-Applikation ist die Bereitstellung der für den Wertschöpfungsprozess notwendigen Funktionen, d. h. die Unterstützung der in Abschnitt 6.1.3 vorgestellten Informations-, Verein-

[10] Ab Version 1.1 ist es in Struts möglich, die gesamte Applikation in Module zu unterteilen, wobei jedes Modul eine eigene struts-config.xml hat. Dies ändert jedoch nichts am Konzept der Ablaufflusssteuerung von Struts, da die Gesamtheit aller Dateien *struts-config.xml* als eine zentrale Steuerungdatei betrachtet werden kann.

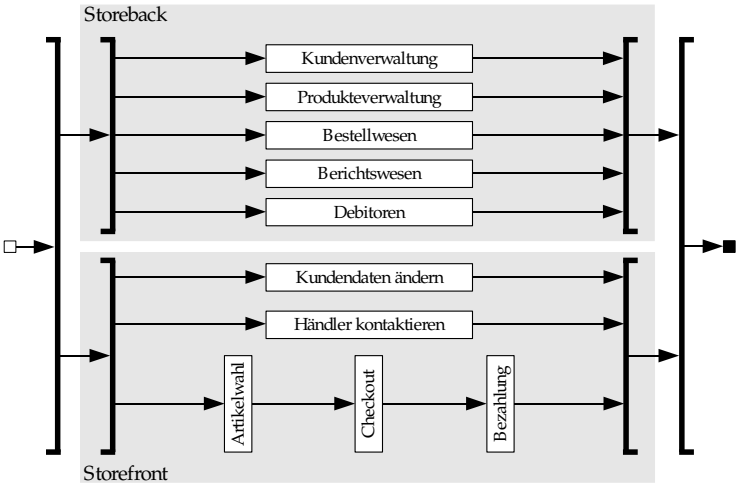


Abbildung 6.5: *Prozessmodell eines Webshops.*

barungs- und Abwicklungsphase. Abb. 6.5 zeigt das Prozessmodell eines typischen Webshops.

Die im oberen Teil von Abb. 6.5 aufgeführten Prozesse betreffen den Shop-Betreiber und laufen im für den Kunden nicht zugänglichen Storeback ab, während die im unteren Teil der Abbildung gezeigten Prozesse im öffentlichen Bereich des Storefront bereitstehen und von den Kunden initiiert werden.

6.4.1 Storeback

Welche Prozesse im Storeback ablaufen, hängt von der Integration des Webshops in das Informationssystem des Anbieters ab. Wenn der Webshop einer von mehreren Absatzkanälen ist, werden die aus den Prozessen anfallenden Daten aus Gründen der Wartbarkeit möglichst ausserhalb des Webshops verwaltet. Umgekehrt kann, wenn der Vertrieb einzig über den Webshop geschieht, die Datenverwaltung komplett im Webshop selber abgewickelt werden. Das würde heissen, dass zu den in Abb. 6.5 gezeigten Prozessen im Sto-

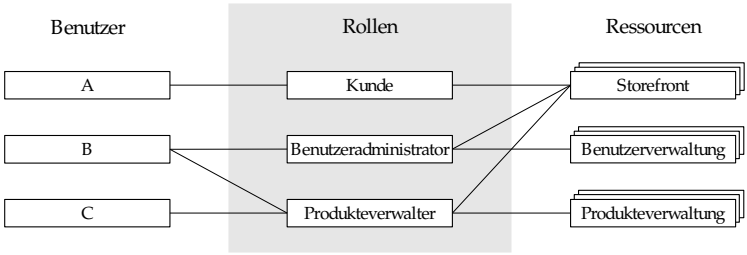


Abbildung 6.6: *Beispiel einer rollenbasierten Rechtevergabe.*

reback weitere Prozesse hinzukommen. Dies wären u. a. die Korrespondenz, die Kreditoren-, die Lohn- und die Finanzbuchhaltung.

6.4.1.1 Benutzerverwaltung

Die *Benutzerverwaltung* erlaubt dem Betreiber des Webshops einerseits das Nachführen personalisierter Angaben zu Kunden, wie Name, Rechnungs-, Lieferadresse, bevorzugte Sprache, Treueprämien, Konditionen u. a. m., andererseits die Berechtigungsvergabe derart, dass die Vertraulichkeit und die Integrität aller Daten sichergestellt ist.

Eine rollenbasierte Rechtevergabe ermöglicht hohe Flexibilität bei geringem Aufwand. Indem eine Anzahl Berechtigungen zu einer Rolle zusammengefasst und einem oder mehreren Benutzern zugeteilt werden können, verringert sich der Zeitbedarf für die Pflege der Berechtigungen beträchtlich. Auch ist es möglich, dass ein Benutzer verschiedene Rollen zugeteilt erhält. Ein Beispiel ist der Mitarbeiter, der gleichzeitig Kunde ist. In Abb. 6.6 ist ein Beispiel einer rollenbasierten Rechtevergabe aufgeführt.

Benutzer A ist ein Kunde und kann lediglich auf den Storefront zugreifen. B dagegen, für die Produkte und die Vergabe von Benutzerrechten zuständig, hat zusätzlich Zugriff auf die Produkte- und die Benutzerverwaltung. C führt laufend den Produktekatalog nach und kann auf die Produkteverwaltung und auf den Storefront zugreifen, nicht aber auf die Benutzerverwaltung.

6.4.1.2 Produkteverwaltung

Mit der Produkteverwaltung wird das Waren- oder Dienstleistungsangebot typisiert, kategorisiert, mit Abbildungen, Beschreibungen und einem Preis versehen und anschliessend publiziert.

Während ein Artikel im Normalfall über einen Bezeichner und eine Beschreibung verfügt, sind die weiteren Attribute von der Beschaffenheit des Artikels abhängig. Ein Buch bspw. trägt einen Autor, einen Verleger und eine ISBN-Nummer, ein Kleidungsstück Grösse, Material und Farbe. Einige Shops tragen diesem Umstand Rechnung und geben dem Produkteverwalter die Möglichkeit, unterschiedliche Artikelausprägungen zu definieren. Im Falle des Shops *esarine* (s. WSFM04) werden dazu individuelle Produkttypen angelegt und mit beliebig vielen Attributen versehen. Dabei kann es sich um statische Attribute handeln, die bei der Artikelbeschreibung frei belegt werden können (bspw. für eine Kurzbeschreibung), oder um Attribute, die mit Werten vorbelegt sind (wie bspw. für eine vorgegebene Auswahl an den Grössen S, M und L). Damit ist gewährleistet, dass in der Artikelverwaltung nur Attribute gepflegt werden müssen, die für ein bestimmtes Produkt relevant sind.

Zur einfacheren Auffindbarkeit werden Kategorien gebildet, welchen anschliessend die Artikel zugeordnet werden, wobei ein Artikel mehreren Kategorien zugehörig sein kann.

6.4.1.3 Bestellwesen

Im Bestellwesen sieht der Shop-Betreiber alle von Kunden getätigten Bestellungen und kann die Auslieferung sowie die Verbuchung anstossen. Hat ein Kunde Änderungswünsche, oder kommt eine Bestellung per Telephon oder E-Mail, kann der Shop-Betereiber dies ebenfalls im Bestellwesen bearbeiten.

6.4.1.4 Berichtswesen

Ein Berichtswesen auf operativer Ebene erlaubt dem Shop-Betreiber, den Erfolg von Werbemassnahmen und weiteren Aktionen sichten zu können. Weitergehende Techniken wie Datawarehousing und die Verdichtung zu Kennzahlen und Reports für die strategische Ebene sind in aller Regel ausserhalb des Webshops angesiedelt, da der Webshop üblicherweise nur eine von mehreren Datenquellen darstellt.

6.4.1.5 Debitoren

Die aus der Abarbeitung einer Kundenbestellung entstehenden Forderungen aus Lieferungen und Leistungen werden erfasst und in die Debitorenbuchhaltung transferiert, welche sich normalerweise ausserhalb des Webshops befindet und die Informationen für das Mahnwesen und die Liquiditätskontrolle bereitstellt.

6.4.2 Storefront

Während es für den Shop-Betreiber abhängig von der eingangs erwähnten Integration des Webshops unterschiedliche Zugänge zur Datenverwaltung geben kann, ist der Storefront die alleinige Schnittstelle für den Kunden. Entsprechend kommt dieser Schnittstelle eine grosse Bedeutung zu, indem sie sich der Zielgruppe entsprechend positioniert und dem Besucher die von ihm gewünschten Informationen in geeigneter Form und Detaillierung bereitstellt, ihm aber auch die Möglichkeit der Interaktion in die Hand gibt. Ferner lassen sich bestimmte Tätigkeiten auslagern und dem Kunden übertragen, wie bspw. die unten beschriebene Verwaltung der Kundendaten.

Die Prozesse *Kundendaten ändern* und *Händler kontaktieren* sind unabhängig, der Kaufprozess ist in die sequentiell ablaufenden Prozesse *Artikelwahl*, *Checkout* und *Bezahlung* unterteilt.

6.4.2.1 Kundendaten ändern

Indem man dem Kunden die Möglichkeit gibt, seine Daten wie Anschrift, Bezahlart, Lieferadresse usw. selber nachzuführen, wird die Stammdatenverwaltung auch für den Shop-Betreiber vereinfacht. Zusätzlich hat der Kunde die Möglichkeit der Personalisierung, indem er Einstellungen wie Sprache, Anzahl angezeigte Artikel pro Seite u. a. m. vornehmen kann.

6.4.2.2 Händler kontaktieren

Die Kontaktaufnahme ermöglicht es dem Kunden, seine Fragen und Anregungen an den Händler zu richten. Dem Händler gibt die Kontaktaufnahme die Gelegenheit, Bedürfnisse auf dem Markt zu erkennen.

6.4.2.3 Kaufprozess Artikelwahl, Checkout und Bezahlung

In der Artikelwahl kann sich der Kunde Waren- und Dienstleistungsangebote anzeigen und im virtuellen Einkaufskorb für einen möglichen Kauf deponieren lassen, so dass er jederzeit den Gesamtpreis samt Versandkosten und Rabatten angezeigt bekommt.

Im Checkout geht der Kunde zur virtuellen Kasse, wo er sich ggü. dem Shop-Betreiber identifiziert, die allgemeinen Geschäftsbedingungen bestätigt und allfällige weitere Angaben macht.

Nach dem Checkout wird ein vom Zahlungssystem abhängiger Zahlungsprozess ausgelöst. Dies kann eine Fakturierung, eine Abbuchung auf Kreditkarte oder auf ein anderes Zahlungsmittel sein. Danach kann der Kunde im Falle eines online verfügbaren Artikels diesen herunterladen, bei einem bestellten physischen Gut darf er mit dessen Lieferung rechnen.

6.5 Der Geschäftsvorfall aus organisatorisch-betriebswirtschaftlicher Sicht

Die im Kapitel 5 und in den Abschnitten 6.2 und 6.3 vorgestellten Konzepte sollen nun an einem konkreten Beispiel veranschaulicht werden. Dazu wird die kundenseitige Grundfunktion Offert- und Bestellwesen aus Abschnitt 6.4 in ein APN überführt. Es werden die betriebswirtschaftliche und die informationstechnische Sicht abgebildet, um als Ergebnis einen Entwurf der Web-Seiten und der Action Mappings der Datei struts-config.xml zu erhalten. Das Beispiel ist bewusst einfach gehalten, eine realistische Electronic-Commerce-Anwendung würde dem Besucher weit mehr Funktionen und Auswahlmöglichkeiten bieten.

Die Grundfunktion Offert- und Bestellwesen präsentiert sich aus organisatorischer Sicht so, dass ein Kunde in der Lage ist, aus einem ihm vorgelegten Waren- und Dienstleistungsangebot ein ihm zusagendes Produkt auszuwählen und seine Kaufabsicht kundzutun.

Der Ablauf von der Präsentation des Angebots bis zum Abschluss des Geschäfts kann auf unterschiedliche Weise erfolgen. Der Kunde kann sich von den Sonderangeboten leiten lassen, einen bestimmten Artikel suchen oder im gesamten Angebot stöbern. Er kann dabei von der Möglichkeit der Personalisierung Gebrauch machen und seine Präferenzen, wie bevorzugte Sprache,

Art der Darstellung am Bildschirm u. a. m., festlegen und sich registrieren, damit er beim nächsten Besuch sein individuelles Erscheinungsbild vorfindet. Weitere Punkte, die den Gang der Bestellung beeinflussen, sind der Lieferort oder die Verfügbarkeit der Artikel. Der Gestehungspreis kann von der gewählten Menge abhängig sein. Schliesslich kann es unerwartete Ereignisse geben, die einer Fehlerbehandlung bedürfen, bspw. weil sich der Kunde bei der Anmeldung nicht mehr an das Passwort erinnern kann oder weil ein technisches Problem auftritt.

Die graphische Modellierung dieser Sachverhalte sprengt das Fassungsvermögen einer A4-Seite und trägt nicht zur Veranschaulichung der folgenden Ausführungen bei. Daher gelten für das folgende Anwendungsbeispiel folgende Vereinfachungen: Der Kunde kennt die URI des Shops, hat sich bei einem früheren Besuch dort unter Angabe von Name, Adresse und Bezahlweise registriert und weiss im voraus, welchen Artikel er kaufen will. Damit liesse sich ein Bestellvorgang aus organisatorischer Sicht wie folgt formulieren:

- Der Kunde tippt im Browser die URI des Webshops ein, es erscheint eine Startseite mit verschiedenen Angeboten.

- Der Kunde sucht sich einen Artikel aus und legt ihn in den virtuellen Warenkorb, der Inhalt des virtuellen Warenkorbs wird angezeigt.

- Der Kunde geht zur virtuellen Kasse, es erscheint eine Seite mit der Möglichkeit, sich einzuloggen oder sich zu registrieren.

- Der Kunde loggt sich mit Benutzernamen und Passwort ein, er wird nach gewünschter Versandart gefragt.

- Der Kunde wählt die Versandart *express*, es wird ihm die Übersicht seiner Bestellung präsentiert.

- Der Kunde bestätigt die Bestellung, er wird mit einer Seite verabschiedet, die seinen Auftrag verdankt.

6.6 Der Geschäftsvorfall aus informationstechnischer Sicht

Damit der geschilderte Geschäftsvorfall ablaufen kann, sind in Struts folgende *Actions* und *Pages*[11] für die Bildschirmausgabe zu implementieren:

- Die Action *WelcomeAction* ermittelt die verfügbaren Artikel und reicht das Ergebnis an die View *welcome.jsp* weiter, der Kunde wählt aus.
- *AddToCartAction* setzt den Lagerbestand um die Anzahl ausgewählter Artikel zurück und gibt eine Repräsentation des Artikels an *myCart.jsp* zurück, der Kunde geht zur Kasse.
- *CheckoutAction* prüft, ob der Kunde bereits angemeldet ist. Falls dies nicht der Fall ist, wird ihm mittels *login.jsp* die Möglichkeit gegeben, sich einzuloggen oder sich als Neukunde zu registrieren, der Kunde loggt sich ein.
- *LoginAction* prüft das Login und lässt anschliessend den Kunden mittels *shipping.jsp* zwischen Versandart *normal* und *express* wählen, der Kunde wählt aus.
- *ChooseShippingAction* nimmt die gewünschte Versandart entgegen und gibt über *order.jsp* die Details der Bestellung zur Bestätigung aus, der Kunde bestätigt die Angaben.
- *ConfirmCheckoutAction* schliesslich löst den Bestellvorgang aus und verabschiedet über *goodbye.jsp* dankend den Kunden.

6.7 Der Geschäftsvorfall als APN und dessen Abbildung in *struts-config.xml*

Um den oben beschriebenen Sachverhalt in Struts abzubilden, wird ein APN modelliert und um Struts-spezifische Elemente erweitert. Dieses APN lässt sich anschliessend in die Datei *struts-config.xml* überführen, wo die Abschnitte *form-beans* und *action-mappings* generiert werden.

[11] Hier sind die Pages als Java Server Pages, erkennbar an der Dateiendung *.jsp* realisiert. Daneben gibt es weitere Techniken, erwähnt sei als Beispiel das Apache-Projekt *Velocity*, die hier nicht vorgestellt werden.

6.7.1 Fachliche Sicht

Abb. 6.7 zeigt den Geschäftsvorfall als APN, enthaltend die organisatorisch-betriebswirtschaftliche Sicht. Die Beschreibung, was der Kunde tut, erfolgt allein aus der fachlichen Sicht, während die Reaktion des Systems sowohl aus fachlicher wie aus technischer Sicht zu definieren ist: der Inhalt der Antwortseite ist fachlicher, die Umsetzung technischer Natur. Ebenfalls in die technische Verantwortung gehört das Festlegen der *Actions* und deren Verknüpfung mit der Geschäftslogik.

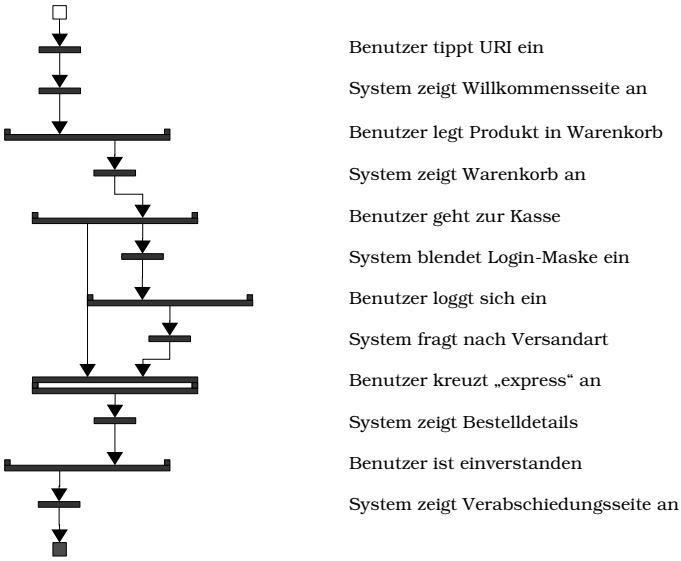


Abbildung 6.7: *Der Bestellvorgang als APN aus fachlicher Sicht.*

6.7.2 Technische Sicht

Damit aus dem vorliegenden APN die *action-mappings* der Steuerungsdatei *struts-config.xml* erzeugt werden können, wird das Modell wie in Abb. 6.8 um

Struts-spezifische Attribute ergänzt[12]. Die Attribute aus fachlicher Sicht von Abb. 6.7 sind der Übersichtlichkeit halber nicht wieder aufgeführt:

- Mit *path* wird der Pfad der vom Kunden übermittelten Anforderung bezeichnet, standardmässig ohne die Dateierweiterung (z. B. *.do*).
- Das Attribut *type* bezeichnet die *Action*-Klasse, die aufgerufen werden soll.
- Das Attribut *view* bezeichnet die View, wie sie der Kunde zu Gesicht bekommt.
- Das Attribut *case* beschreibt die fallweise Verzweigung, die die mit *view* angegebene Seite aufruft.
- Mit dem Attribut *form* wird ein Eingabeformular bestimmt, welches bei fehlenden oder ungültigen Benutzereingaben automatisch eine Fehlermeldung ausgibt.

Auf die weitere Untergliederung der Aktivitäten, um die Geschäftslogik zu modellieren, wird im Beispiel verzichtet. Bspw. liesse sich die Aktivität *Benutzer legt Produkt in Warenkorb* resp. die zum Zug kommende implementierte Aktivität *AddToCartAction* derart in Subaktivitäten unterteilen, dass die dahinter vermutete Lagerbestandesprüfung und -aktualisierung ersichtlich wird und die Detailspezifikation für den Programmierer damit vorhanden ist.

6.7.3 Ableiten der Abschnitte *form-beans* und *action-mappings* von *struts-config.xml*

Mit den Angaben in Abb. 6.8 lassen sich die Liste der notwendigen Eingabeformulare sowie die Flusssteuerung in der Datei *struts-config.xml* wie folgt generieren:

Jedes Attribut *form* wird in seine entsprechende Deklaration im Abschnitt *form-beans* überführt. Im Beispiel sind dies die für die Interaktionen erforderlichen Eingabeformulare für das Login und für die Wahl der Versandart:

[12]Die oben eingeführten Attribute sind nicht vollständig, für eine praxistaugliche Anwendung müssten weitere Attribute hinzugezogen werden, um eine voll funktionsfähige Datei *struts-config.xml* zu erzeugen. Dies ist nicht Gegenstand dieser Arbeit.

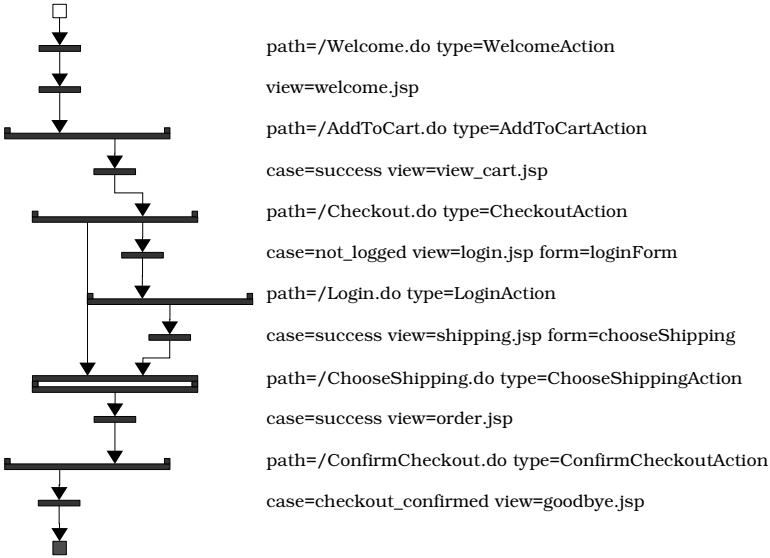


Abbildung 6.8: *Der Bestellvorgang als APN aus technischer Sicht.*

```
<form-beans>
 <form-bean
  name="loginForm"
  type="com.esarine.storefront.LoginForm"/>
 <form-bean
  name="chooseShipping"
  type="com.esarine.storefront.ChooseShippingForm"/>
</form-beans>
```

Jedes Attribut *path* findet seinen Eintrag im Abschnitt *action-mappings* als eigenes Element *action*, wobei das APN-Attribut *path* dem Attribut *path* von *struts-config.xml* entspricht. Mit den folgenden Abbildungsregeln können die Elemente *action* vervollständigt werden:

- Attribut *type* in APN entspricht dem *type* in *struts-config.xml*.
- Die APN-Attribute *case* und *view* ergeben in *struts-config.xml* ein Element *forward* als Attribut von *action*, wobei *case* dem *name* und *view* dem *path* entspricht.
- Das APN-Attribut *form* wird in das Attribut *name* überführt. Zusätzlich wird ein Attribut *validate* generiert und auf *true* gesetzt sowie ein Attribut *input* eingefügt, welches auf den Namen des Eingabeformulars lautet. Damit wird bei einer erfolglosen Validierung der Benutzereingaben eine Fehlermeldung im selben Formular erzeugt.
- Dort, wo nur in eine *view* verzweigt werden kann (im Beispiel die *view= welcome.jsp*), wird automatisch ein *forward* mit *name=success* gesetzt.

Damit resultieren die folgenden *action-mappings* von *struts-config.xml*:

```
<action-mappings>
 <action
  path="/Welcome"
  type="com.esarine.storefront.WelcomeAction">
  <forward name="success" path="welcome.jsp"/>
 </action>
 <action
  path="/AddToCart"
  type="com.esarine.storefront.AddToCartAction">
  <forward name="success" path="viewCart.jsp"/>
  <forward name="sold_out" path="soldOut.jsp"/>
 </action>
 <action
  path="/Checkout"
  type="com.esarine.storefront.CheckoutAction">
  <forward name="notLogged" path="login.jsp"/>
  <forward name="logged" path="chooseShipping.jsp"/>
 </action>
 <action
  path="/Login"
  type="com.esarine.storefront.LoginAction"
```

```
  name="loginForm"
  validate="true"
  input="login.jsp">
  <forward name="success" path="chooseShipping.jsp"/>
  <forward name="failure" path="login.jsp"/>
 </action>
 <action
  path="/ChooseShipping"
  type="com.esarine.storefront.ChooseShippingAction"
  name="chooseShipping"
  validate="true"
  input="chooseShipping.jsp">
  <forward name="success" path="confirmCheckout.jsp"/>
 </action>
 <action
  path="/ConfirmCheckout"
  type="com.esarine.storefront.ConfirmCheckoutAction">
  <forward name="success" path="goodbye.jsp"/>
  <forward name="edit" path="viewCart.jsp"/>
 </action>
</action-mappings>
```

Damit ist der Kontrollfluss aus der Abb. 6.8 definiert. Für die Implementierung im Struts-Framework sind die in der Datei struts-config.xml aufgeführten Elemente zu implementieren, die Business-Objekte, welche die Geschäftsdaten tragen, zu definieren und deren Persistenz sicherzustellen.

6.8 Architektur eines Webshops in Struts am Beispiel von *esarine*

Abb. 6.9 zeigt die Architektur des unter der GPLv2 freigegebenen Webshops *esarine*[13]. esarine verfügt über die sieben Module *admin*, *common*, *inventory*,

[13]Weitere Informationen zu esarine finden sich unter http://diuf.unifr.ch/is/esarine, wo esarine auch heruntergeladen werden kann. Eine Implementierung von esarine findet sich nebst anderen unter http://www.lehrmittelshop.so.ch/shlm/esarine/sf/Welcome.do.

ordering, *payment*, *security* und *shipping*, welche jeweils die View-, Controller- und Model-Elemente gemäss MVC-Schema beinhalten.

Drei Module sind in der Abbildung gezeigt, wobei das Modul Ordering die für den oben gezeigten Kundenbestellvorgang relevanten Teile explizit aufführt. Es sind dies für die View die ActionForms und die Anzeigeseiten für die Begrüssung, die Produkteansicht, die Suche, die Anmeldung und die Auswahl von Versandkanal und Bezahlart sowie die Verabschiedung. Für das Model sind die Business-Objekte zuständig. Sie sind die in den Prozessen betroffenen Entitäten, tragen die Daten und sind für deren dauerhafte Speicherung zuständig. Für die Änderung der Daten der Business-Objekte verantwortlich sind die Actions des Controllers; in ihnen steckt die Geschäftslogik. Die für die Steuerung sämtlicher Abläufe zwischen Besucher und Anwendung zuständige Instanz des ActionServlets ist in der Spalte ganz rechts abgebildet, zusammen mit der Konfigurationsdatei. Die Actions bilden zusammen mit dem Action-Servlet den Controller-Teil. Das relationale Datenbankmanagementsystem rechts unten schliesslich nimmt die vom DB Wrapper herstellerspezifisch übersetzten Befehle entgegen, welche von den Business-Objekten via den Connection-Pool erteilt werden.

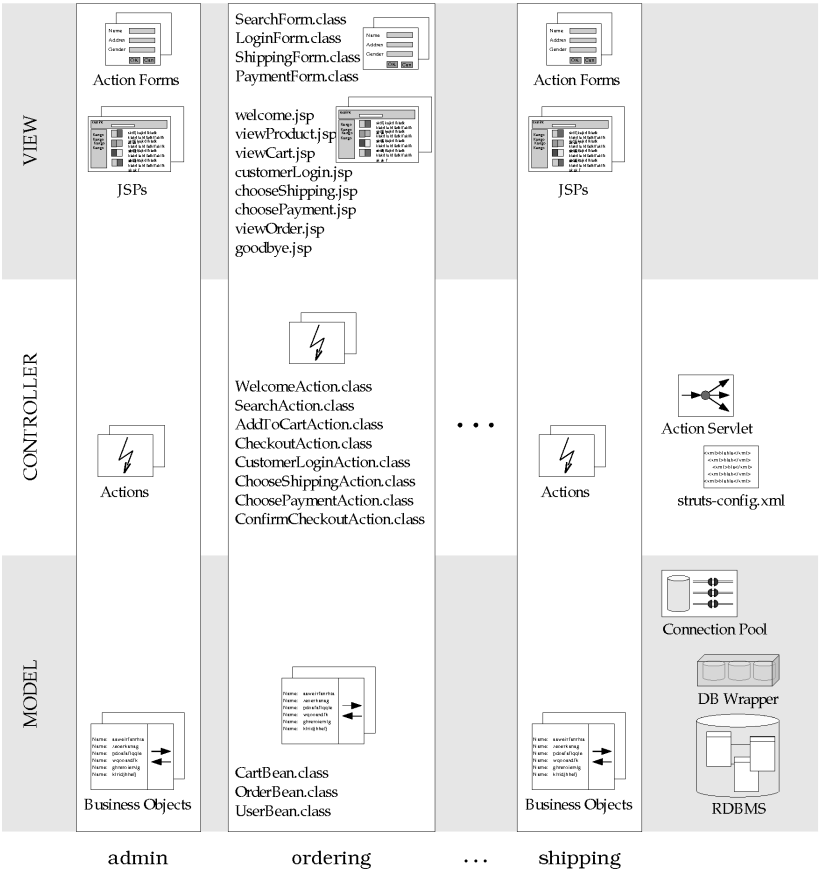


Abbildung 6.9: *Die Architektur von esarine. Dargestellt sind drei von insgesamt sieben nach Funktionen aufgeteilte Module und deren Inhalte auf den Ebenen View, Controller und Model sowie die übergreifenden Komponenten zur Ablaufsteuerung und Persistenz (rechts von der Komponente Versand dargestellt).*

Kapitel 7

Konklusion

In der folgenden Zusammenfassung werden die Forschungsfragen (s. Abschnitt 1.3) und die Richtlinien nach HMPR04 (s. Abschnitt 1.4) aufgegriffen und ein Résumé gezogen.

Im Zentrum des Geschäftsprozess-Managements steht die Geschäftsprozessmodellierung. Sie legt die gesamte Prozessorganisation fest und dient der Synthese, Implementierung, Analyse, Optimierung und Dokumentation der Geschäftsprozesse, die sich in Führungs-, Kern- und Unterstützungsprozesse aufteilen. Damit ist die Modellierung eine für die Unternehmung/Verwaltung kritische Komponente, und entsprechend sind die Anforderungen an das aus der Modellierung entstehende Modell zu gewichten. Die *Problem Relevance* (vgl. Abschnitt 1.4) ist gegeben, das Forschungsziel bekannt.

In Abschnitt 2.2 wurde eine Auswahl an Arbeiten zur Qualitätssicherung in der Modellierung untersucht und daraus eine Liste von neun Kriterien hergeleitet. Ein Modell, das diesen Kriterien entspricht, soll als den Anforderungen entsprechend gelten. Damit ist Forschungsfrage 1 beantwortet. Als Antwort auf Forschungsfrage 2 wurden in Abschnitt 2.2.4 Hinweise gegeben, über welche Eigenschaften eine Modelliersprache verfügen soll, um das Erstellen eines den neun Kriterien entsprechenden Modells zu unterstützen. Diese Kriterien liefern ebenfalls ihren Beitrag an die *Research Contributions* aus Abschnitt 1.4.

Die Untersuchung der zwei Modelliersprachen in Kap. 3 zu Handen Forschungsfrage 3 und als erste *Design Evaluation* hat einige Defizite bzgl. Lesbar-

keit, Wirtschaftlichkeit, Minimalität, Uniformität oder aber allgemeiner Akzeptanz aufgezeigt. Für die Modellierung von Abläufen auf detaillierter Ebene müssen bei eEPK zusätzliche Regeln aufgestellt werden, damit die Korrektheit gegeben ist. Die nicht immer explizite Bedeutung von bestimmten Konstrukten wie bspw. die Unterscheidung von Objekt- und Kontrollfluss und die Möglichkeit, gleiche Sachverhalte, z. B. die Objekte, unterschiedlich darstellen zu können, machen die Aktivitätsdiagramme von UML für weniger geübte Anwender schwer lesbar. Die Aktivitätsdiagramme und die weiteren von UML zur Verfügung gestellten Modelle legen den Fokus auf die informationstechnische Implementierung, weniger auf die betriebswirtschaftlichen Elemente.

Mit den in Kap. 4 vorgestellten Petri-Netzen steht eine weitere, theoretisch fundierte Modelliermethode zur Verfügung, die ausserhalb des akademischen Umfelds nur zögerlich eingesetzt wird. Dem weniger geübten Leser erschliesst sich die Bedeutung gewisser Konstrukte in Petri-Netzen nicht auf Anhieb, zudem werden mit grundlegenden Petri-Netzen geschaffene Geschäftsprozessmodelle sehr umfangreich, und die deutlich kompakteren Modelle, die mit den höheren Petri-Netzen erzielt werden, sind schwer lesbar. Die Darstellung der Zustände in Petri-Netzen ist für die Struktur eines Geschäftsprozesses nicht von Bedeutung, jedoch für die Semantik bei Verzweigungen notwendig, dies macht die Modelle zusätzlich gross und unübersichtlich. Die Möglichkeit, Petri-Netze zu erweitern, lässt Spielraum für die Angabe geschäftsprozessrelevanter Daten, so dass die Vollständigkeit mit Petri-Netzen erreicht werden kann.

Die Petri-Netze dienten ihrer Einfachheit und Formalisierung wegen bei der Herleitung (*Design as an Artifact*) der APN in Kap. 5 als Grundlage. Mit der Beurteilung der APN hinsichtlich ihrer Erfüllung der Anforderungen in Abschnitt 5.6 und dem Beispiel aus Kap. 6 zur Anwendung von APN in der Implementierung von E-Commerce ist Forschungsfrage 4 vorderhand beantwortet und die *Design Evaluation* durchgeführt. Das festgestellte Defizit bei der Minimalität dürfte in der Praxis keine grosse Rolle spielen. Um das Kriterium der Lesbarkeit der APN abschliessend zu beurteilen, wäre ggf. eine empirische Untersuchung angezeigt.

Die Einhaltung der Richtlinien *Research Rigor* und *Design as a Search Process* ist durch die vorliegende, auf den Forschungsfragen aus Abschnitt 1.3 beruhende Arbeit dokumentiert. Und die Kommunikation der Ergebnisse, Richt-

linie *Communication of Research*, erfolgt in Form der vorliegenden publizierten Arbeit.

Einige Fragen sind nur teilweise beantwortet. Zur Evaluation der verschiedenen Prozessmodelle bleibt anzumerken, dass die vorliegenden Kriterienkataloge aus der Literatur und der abgeleitete Kriterienkatalog grossteils auf pragmatischen Forderungen beruhen und nur teilweise empirisch erhärtet sind. Ebenfalls wäre die Orthogonalität der vorgeschlagenen Kriterien zueinander zu prüfen, damit dies entsprechend in der Bewertung eines Kriteriums berücksichtigt werden kann.

Die enge und immer noch zunehmende Kooperation und Kommunikation von Lieferanten und Abnehmern und damit von Unternehmen, Verwaltungen und Privatpersonen untereinander führt zu einem hohen Koordinationsdruck. Diesem kann mit dem Prozess-Management begegnet werdern, indem auf strategischer Ebene Überlegungen angestellt werden, welche Geschäftsfelder alleine, im Auftrag durch andere oder in Kooperation mit anderen abgedeckt werden sollen. Auf taktischer/operativer Ebene ergibt sich daraus die Aufgabe, die Kunden und Lieferanten derart in den Gesamtprozess einzubinden, dass im Sinne der hohen Effizienz und Kundenzufriedenheit möglichst wenige Schnittstellen/Medienbrüche entstehen. Gleichzeitig sind unter Beachtung der Verantwortlichkeiten Vorkehrungen zur Wahrung von Datensicherheit und Datenschutz zu treffen. Auf technischer Ebene sind die Schnittstellen zu gestalten, die für den Informationsfluss notwendigen Protokolle zu unterstützen und die Prozesse in Abhängigkeit von ihrem Wiederholungsgrad zu automatisieren. Es wäre interessant, zu erfahren, in welchem Umfang (sowohl vertikal wie horizontal) ein bestimmtes Werkzeug das Prozess-Management hierbei unterstützen kann und ob allenfalls technische, organisatorische oder politische Hürden vorhanden sind. Für APN wäre hiezu ein Modellier-, Analyse- und Entwicklungswerkzeug anzupassen/zu entwickeln, um weitere Erkenntnisse darüber zu gewinnen, wie gut die obigen Anforderungen in der Praxis durch den Modellansatz erfüllt werden.

Literaturverzeichnis

AG03 van der Aalst, W.; van de Graaf, M. Workflow Systems. In: Girault und Valk GV03, S. 507–540.

AK06 Ahlrichs, Frank; Knuppertz, Thilo. Controlling von Geschäftsprozessen. Schäffer-Poeschel, Stuttgart (2006).

AM06 Allen, Gove N. (A.); March, Salvatore T. A Critical Assessment of the Bunge-Wand-Weber Ontology for Conceptual Modeling. In: Proceedings of the Workshop on Information Technologies and Systems (WITS). Milwaukee (December 2006).

Av08 Antoniou, Grigoris; van Harmelen, Frank. A Semantic Web Primer. The MIT Press, 2. Aufl. (2008).

BCN91 Batini, Carlo; Ceri, Stefano; Navathe, Shamkant B. Conceptual Database Design – An Entity-Relationship Approach. Addison Wesley (1991).

BFN85 Batini, Carlo; Furlani, L.; Nardelli, Enrico. What is a Good Diagram? A Pragmatic Approach. In: Chen, Peter P. (Hrsg.), Entity-Relationship Approach: The Use of ER Concept in Knowledge Representation, Proceedings of the Fourth International Conference on Entity-Relationship Approach, S. 312–319. IEEE Computer Society and North-Holland, Chicago (October 1985).

BKR08 Becker, Jörg; Kugeler, Martin; Rosemann, Michael (Hrsg.). Prozessmanagement – Ein Leitfaden zur prozessorientierten Organi-

sationsgestaltung. Springer, Berlin, Heidelberg, New York, sechste, überarbeitete und erweiterte Aufl. (2008).

BL89 Berners-Lee, Tim. Information Management: A Proposal (März 1989). URL http://www.w3.org/History/1989/proposal.html. Letzter Abruf 9. Mai 2007.

BL00 Berners-Lee, Tim. Weaving the Web: The Original Design and Ultimate Destiny of the World Wide Web. HarperBusiness (2000).

BMW09 Becker, Jörg; Mathas, Christoph; Winkelmann, Axel. Geschäftsprozessmanagement. Springer, Berlin (2009).

Bro91 Brombacher, Reinhard. Effizientes Informationsmanagement – Die Herausforderung von Gegenwart und Zukunft. In: Schriften zur Unternehmensführung, Wiesbaden, Bd. 44, (1991), S. 111–134.

BRS95 Becker, Jörg; Rosemann, Michael; Schütte, Reinhard. Grundsätze ordnungsmäßiger Modellierung (GoM). In: Wirtschaftsinformatik, Bd. 5, (1995), S. 435–445.

BRU00 Becker, Jörg; Rosemann, Michael; von Uthmann, Christoph. Guidelines of Business Process Modeling. In: van der Aalst, Wil; Desel, Jörg; Oberweis, Andreas (Hrsg.), Business Process Management – Models, Techniques, and Empirical Studies, S. 30–49. Springer, Berlin, Heidelberg (2000).

Bun77 Bunge, Mario. Treatise On Basic Philosophy: Volume 3: Ontology I: The Furniture of the World. Reidel, Dordrecht (1977).

Bun79 Bunge, Mario. Treatise On Basic Philosophy: Volume 4: Ontology II: A World of Systems. Reidel, Dordrecht (1979).

Bun08 Bundeskanzlei. Obligationenrecht (2008).

Bur92 Burbeck, Steve. Applications Programming in Smalltalk-80: How to use Model-View-Controller (MVC) (1992). URL http://st-www.cs.uiuc.edu/users/smarch/st-docs/mvc.html. Letzter Abruf 25. August 2007.

Cav03 Cavaness, Chuck. Programming Jakarta Struts. O'Reilly, Sebastopol CA (2003).

Che76 Chen, Peter Pin-Shan. The Entity-Relationship Model – Toward a Unified View of Data. In: ACM Transactions on Database Systems, Bd. 1 (1), (1976), S. 9–36.

CJB99 Chandrasekaran, Balakrishnan; Josephson, John R.; Benjamins, V. Richard. What Are Ontologies, and Why Do We Need Them? In: IEEE Intelligent Systems, Bd. 14 (1), (1999), S. 20–26.

Dav01 Davis, Malcolm. Struts, an open-source mvc implementation (Februar 2001). URL `http://www.ibm.com/developerworks/library/j-struts/`. Letzter Abruf 17. Juli 2007.

Deu89 Deutsch, L. Peter. Design Reuse and Frameworks in the Smalltalk-80 System. In: Biggerstaff, Ted J.; Perlis, Alan J. (Hrsg.), Software Reusability, Volume II: Applications and Experience, S. 57–71. Addison-Wesley, Reading, MA (1989).

DGMR05 Davies, Islay; Green, Peter; Milton, Simon; Rosemann, Michael. Analyzing and Comparing Ontologies with Meta-Models. In: Krogstie, John; Halpin, Terry A.; Siau, Keng (Hrsg.), Information Modeling Methods and Methodologies, S. 1–16. Idea Group Publishing (2005).

DSZ03 Dittmann, Lars; Schütte, Reinhard; Zelewski, Stephan. Darstellende Untersuchung philosophischer Probleme mit Ontologien. In: Freyberg, Klaus; Petsche, Hans-Joachim; Klein, Bertin (Hrsg.), Knowledge Management and Philosophy 2003. Proceedings of the WM 2003 Workshop on Knowledge Management and Philosophy. CEUR Workshop Proceedings, Luzern (April 2003). URL `http://CEUR-WS.org/Vol-85/`. Letzter Abruf 13. Mai 2009.

FB03 Frank, Ulrich; Bodo, van Laak. Anforderungen an Sprachen zur Modellierung von Geschäftsprozessen (2003). URL `http://www.wi-inf.uni-due.de/FGFrank/documents/Arbeitsberichte_Koblenz/Nr34.pdf`. Letzter Abruf 10. August 2008.

FD08 Füermann, Timo; Dammasch, Carsten. Prozessmanagement. Carl Hanser, München, 3. Aufl. (2008).

Fie00 Fielding, Roy Thomas. Architectural Styles and the Design of Network-based Software Architectures. Dissertation, University of California, Irvine (2000). URL `http://www.ics.uci.edu/~fielding/pubs/dissertation/fielding_dissertation.pdf`. Letzter Abruf am 20. Juni 2007.

FL03 Fettke, Peter; Loos, Peter. Ontological Evaluation of Reference Models Using the Bunge-Wand-Weber Model. In: Proceedings of the 9th Americas Conference on Information Systems (AMCIS '03), S. 2944–2955. Tampa, Florida (August 2003).

Fon07 Fonseca, Frederico. The Double Role of Ontologies in Information Science Research. In: Journal of the American Society for Information Science and Technology, Bd. 58 (6), (2007), S. 786–793.

Fra98 Frank, Ulrich. Zur Anreicherung von Modellierungsmethoden mit domänenspezifischem Wissen: Chancen und Herausforderungen der Unternehmensmodellierung. In: Pohl, Klaus; Schürr, Andy; Vossen, Gottfried (Hrsg.), Proceedings der Tagung Modellierung 98, Bd. 9 von *CEUR Workshop Proceedings*, S. 31–35. CEUR-WS.org (März 1998).

Fra99 Frauchiger, Daniel. Chancen, Besonderheiten, Risiken und Perspektiven des Electronic Commerce (Juni 1999). Seminararbeit.

Fra00 Frank, Ulrich. Modelle als Evaluationsobjekt – Einführung und Grundlegung. In: Heinrich, Lutz J.; Häntschel, Irene; et al. (Hrsg.), Evaluation und Evaluationsforschung in der Wirtschaftsinformatik, S. 339–352. Oldenbourg (2000).

Fra01 Frauchiger, Daniel. Vorschlag eines Informationssystems zur Beurteilung von operationellen Risiken an Hand eines Prozessmodells als Hilfsmittel – Diplomarbeit. Diplomarbeit, Universität Freiburg, Schweiz (Juni 2001).

Fra03 Frauchiger, Daniel. APN – A Petri Net Based Business Process Model. In: Smari, Waleed W.; Chen, Shu-Ching; Melab, Nordine (Hrsg.), The 2003 International Conference on Information Systems and Engineering (ISE 2003), S. 139–148. The Society for Modeling and Simulation International, Montréal (July 2003).

FS08 Ferstl, Otto K.; Sinz, Elmar J. Grundlagen der Wirtschaftsinformatik. Oldenbourg, München, sechste, überarbeitete und erweiterte Aufl. (2008).

Gad02 Gadatsch, Andreas. Management von Geschäftsprozessen – Methoden und Werkzeuge für die IT-Praxis: Eine Einführung für Studenten und Praktiker. Vieweg, Braunschweig/Wiesbaden, 2., überarbeitete und erweiterte Aufl. (2002).

Gai83 Gaitanides, Michael. Prozeßorganisation – Entwicklung, Ansätze und Programme prozeßorientierter Organisationsgestaltung. Vahlen, München (1983).

GG95 Guarino, Nicola; Giaretta, Pierdaniele. Ontologies and Knowledge Bases. In: Mars, Nicolaas J. I. (Hrsg.), Towards Very Large Knowledge Bases, S. 25–32. IOS Press, Amsterdam (1995).

GHJV95 Gamma, Erich; Helm, Richard; Johnson, Ralph; Vlissides, John. Design Patterns – Elements of Reusable Object-Oriented Software. Addison-Wesley, Reading, MA (1995).

GL79 Genrich, Hartmann J.; Lautenbach, Kurt. The Analysis of Distributed Systems by Means of Predicate/Transition-Nets. In: Kahn, Gilles (Hrsg.), Semantics of Concurrent Computation, Proceedings of the International Symposium, Evian, Bd. 70 von *Lecture Notes in Computer Science*, S. 123–147. Springer (1979).

GL81 Genrich, Hartmann J.; Lautenbach, Kurt. System Modelling with High-Level Petri Nets. In: Theoretical Computer Science, Bd. 13, (1981), S. 109–136.

GR01 Green, Peter; Rosemann, Michael. Ontological Analysis of Integrated Process Models: Testing Hypotheses. In: Australasian Journal of Information Systems, Bd. 9 (1), (2001), S. 30–38.

GR02 Green, Peter; Rosemann, Michael. Perceived Ontological Weaknesses of Process Modeling Techniques: Further Evidence. In: Wrycza, Stanisław (Hrsg.), Proceedings of the 10th European Conference on Information Systems (ECIS '02), S. 312–321. Gdańsk (June 2002).

GR05 Green, Peter; Rosemann, Michael. Ontological Analysis of Business Systems Analysis Techniques: Experiences and Proposals for an Enhanced Methodology. In: Green, Peter; Rosemann, Michael (Hrsg.), Business Systems Analysis with Ontologies, S. 1–27. Idea Group Publishing, Hershey (2005).

Gru88 Grude, Ulrich. Petri-Netze – eine informelle EinfÃijhrung in die Grundideen (1988). URL `public.beuth-hochschule.de/~grude/Petrinetze.pdf`. Letzter Abruf 22. Februar 2010.

Gru93 Gruber, Thomas R. A Translation Approach to Portable Ontology Specifications. In: Knowledge Acquisition, Bd. 5 (2), (1993), S. 199–220.

Gru09 Gruber, Thomas R. Ontology. In: Özsu, M. Tamer; Liu, Ling (Hrsg.), Encyclopedia of Database Systems. Springer (2009). URL `http://tomgruber.org/writing/ontology-in-encyclopedia.htm`. Letzter Abruf 11. Mai 2009.

Gua98 Guarino, Nicola. Formal Ontology and Information Systems. In: Guarino, Nicola (Hrsg.), Proceedings of Formal Ontology in Information Systems (FOIS) 98, S. 3–15. Amsterdam, IOS Press, Trento (June 1998).

GV03 Girault, Claude; Valk, Rüdiger (Hrsg.). Petri Nets for Systems Engineering – A Guide to Modeling, Verification, and Applications. Springer, Berlin, Heidelberg (2003).

HA04 Holl, Alfred; Auerochs, Robert. Analogisches Denken als Erkenntnisstrategie zur Modellbildung in der Wirtschaftsinformatik. In: Frank, Ulrich (Hrsg.), Wissenschaftstheorie in Ökonomie und Wirtschaftsinformatik, S. 367–389. Gabler, Wiesbaden (2004).

Häs00 Hässig, Kurt. Prozessmanagement – erfolgreich durch effiziente Strukturen. Versus, Zürich (2000).

HC93 Hammer, Michael; Champy, James. Reengineering the Corporation – A Manifesto for Business Revolution. HarperBusiness, New York (1993).

HKR09 Hitzler, Pascal; Krötzsch, Markus; Rudolph, Sebastian. Foundations of Semantic Web Technologies. Chapman & Hall/CRC (2009).

HM03 Hevner, Alan R.; March, Salvatore T. The Information Systems Research Cycle. In: Computer, Bd. 36 (11), (2003), S. 111–113.

HMPR04 Hevner, Alan R.; March, Salvatore T.; Park, Jinsoo; Ram, Sudha. Design Science in Information Systems Research. In: MIS Quarterly, Bd. 28 (1), (2004), S. 75–105.

HS09 Heinrich, Lutz J.; Stelzer, Dirk. Informationsmanagement. Oldenbourg, München, Wien, 9., vollständig überarbeitete Aufl. (2009).

HSK05 Halpin, Terry; Siau, Keng; Krogstie, John (Hrsg.). Proceedings of the 10th International Workshop on Exploring Modeling Methods for Systems Analysis and Design (EMMSAD '05). Porto (June 2005).

Jen81 Jensen, Kurt. Coloured Petri Nets and the Invariant-Method. In: Theoretical Computer Science, Bd. 14, (1981), S. 317–336.

Jen98 Jensen, Kurt. An Introduction to the Practical Use of Coloured Petri Nets. In: Reisig, Wolfgang; Rozenberg, Grzegorz (Hrsg.), Lectures on Petri Nets II: Applications – Advances in Petri Nets, Bd. 2, S. 237–292. Springer, Berlin, Heidelberg, New York (1998).

KAPP08 Kohlas, Jürg; Altorfer, Karin; Pugin, Catherine; Pouly, Marc. 50 Jahre Informatik an der Universität Freiburg (2008).

Khn05 Khneisseh, Amira. Geschäftsprozessmodellierung mit Petri-Netzen. expert Verlag, Renningen (2005).

KK97 Kueng, Peter; Kawalek, Peter. Process Models: a Help or a Burden? In: Proceedings of the Americas Conference on Information Systems, AIS '97, S. 676–678. Indianapolis (August 1997).

KNS92 Keller, Gerhard; Nüttgens, Markus; Scheer, August-Wilhelm. Semantische Prozeßmodellierung auf der Grundlage „Ereignisgesteuerter Prozeßketten (EPK)“. In: Veröffentlichungen des Instituts für Wirtschaftsinformatik, Saarbrücken, Bd. 89. URL http://www.iwi.uni-sb.de/Download/iwihefte/heft89.pdf. Letzter Abruf 31. Juli 2008.

Kos62 Kosiol, Erich. Organisation der Unternehmung. Gabler, Wiesbaden (1962).

KSJ06 Krogstie, John; Sindre, Guttorm; Jørgensen, Håvard. Process Models representing Knowledge for Action: a Revised Quality Framework. In: European Journal of Information Systems, Bd. 15, (2006), S. 91–102.

Kül99 Küll, Roland. Petri-Netz-basierte Simulation von Geschäftsprozessen. Dissertation, Universität St. Gallen, St. Gallen (1999). Dissertation.

Lan09 Lano, Kevin. Model Driven Software Development using UML and Java. Cengage Learning Services (2009).

LSS94 Lindland, Odd Ivar; Sindre, Guttorm; Sølvberg, Arne. Understanding Quality in Conceptual Modeling. In: Software, IEEE, Bd. 11 (2), (1994), S. 42–49.

LWS08 Lehner, Franz; Wildner, Stephan; Scholz, Michael. Wirtschaftsinformatik. Eine Einführung. Carl Hanser, 2. Aufl. (2008).

MN03 Mendling, Jan; Nüttgens, Markus. XML-basierte Geschäftsprozessmodellierung. In: Uhr, Wolfgang; Esswein, Werner; Schoop, Eric (Hrsg.), Wirtschaftsinformatik 2003 / Band II – Medien, Märkte, Mobilität, S. 161–180. Physica-Verlag, Dresden (September 2003).

MPGP05 Maes, Ann; Poels, Geert; Gailly, Frederik; Paemeleire, Roland. Measuring User Beliefs and Attitudes towards Conceptual Models: A Factor and Structural Equation Model (June 2005). URL http://ideas.repec.org/p/rug/rugwps/05-311.html. Letzter Abruf 28. Januar 2009.

MS94 Moody, Daniel L.; Shanks, Graeme G. What Makes a Good Data Model? Evaluating the Quality of Entity Relationship Models. In: Loucopoulos, Pericles (Hrsg.), Proceedings of the 13th International Conference on the Entity-Relationship Approach, Manchester, S. 94–111. Springer, London (December 1994).

MS95 March, Salvatore T.; Smith, Gerald F. Design and natural Science Research on Information Technology. In: Decision Support Systems, (15), (1995), S. 251–266.

MS03 Moody, Daniel L.; Shanks, Graeme G. Improving the quality of data models: Empirical validation of a quality management framework. In: Information Systems, Bd. 28 (6), (2003), S. 619–650.

MS08 Meier, Andreas; Stormer, Henrik. eBusiness & eCommerce – Management der digitalen Wertschöpfungskette. Springer-Verlag, Berlin, Heidelberg, 2. Aufl. (2008).

Mur89 Murata, Tadao. Petri Nets: Properties, Analysis and Applications. In: Proceedings of the IEEE, Bd. 77, S. 541–580 (April 1989).

NJ02 Nüttgens, Markus; J. Rump, Frank. Syntax und Semantik Ereignisgesteuerter Prozessketten (EPK). In: Desel, Jörg; Weske, Mathias (Hrsg.), Prozessorientierte Methoden und Werkzeuge für die Entwicklung von Informationssystemen (Promise 2002), S. 64–77. Gesellschaft für Informatik, Bonn, Potsdam (Oktober 2002).

NK05 Nysetvold, Anna Gunhild; Krogstie, John. Assessing Business Processing Modeling Languages Using a Generic Quality Framework. In: Halpin et al. HSK05, S. 159–170.

Nor34 Nordsieck, Fritz. Grundlagen der Organisationslehre. Poeschel, Stuttgart (1934).

NPGP05 Nelson, H. James; Poels, Geert; Genero, Marcela; Piattini, Mario. Quality in Conceptual Modeling: Five Examples of the State of the Art. In: Data & Knowledge Engineering, Bd. 55 (3), (2005), S. 237–242.

NPW08 Neumann, Stefan; Probst, Christian; Wernsmann, Clemens. Kontinuierliches Prozessmanagement. In: Becker et al. BKR08, S. 299–328.

Obj06 Object Management Group, Inc. Meta Object Facility (MOF) Core Specification (Januar 2006). URL `http://www.omg.org/spec/MOF/2.0/PDF/`. Letzter Abruf 2. August 2008.

Obj07 Object Management Group, Inc. OMG Unified Modeling Language – Infrastructure, V2.1.2 (November 2007). URL `http://www.omg.org/spec/UML/2.1.2/Infrastructure/PDF/`. Letzter Abruf 1. August 2008.

Oes09 Oestereich, Bernd. Die UML 2.3 Kurzreferenz für die Praxis. Oldenbourg, München, 5., überarbeitete Aufl. (2009).

OF96 Osterloh, Margit; Frost, Jetta. Prozessmanagement als Kernkompetenz. Gabler, Wiesbaden, 1. Aufl. (1996).

OF03 Osterloh, Margit; Frost, Jetta. Prozessmanagement als Kernkompetenz. Gabler, Wiesbaden, 4. Aufl. (2003).

Pet62 Petri, Carl Adam. Kommunikation mit Automaten. Dissertation, Universität Bonn, Bonn (1962). Dissertation.

PM07 Pastor, Oscar; Molina, Juan Carlos. Model-Driven Architecture in Practice: A Software Production Environment Based on Conceptual Modeling. Springer, Berlin (2007).

Rec05 Recker, Jan. Conceptual Model Evaluation. Towards more Paradigmatic Rigor. In: Halpin et al. HSK05, S. 183–194.

Rec06 Recker, Jan. Towards an Understanding of Process Model Quality. Methodological Considerations. In: Ljungberg, Jan; Andersson, Magnus (Hrsg.), Proceedings of the 14th European Conference on Information Systems (ECIS '06). Göteborg (June 2006).

Rec07 Recker, Jan. A Study on the Decision to Continue Using a Modeling Grammar. In: Proceedings of the 13th Americas Conference

on Information Systems (AMCIS '07). Keystone, Colorado (August 2007).

Ree79a Reenskaug, Trygve. Models - Views - Controllers (December 1979). URL http://heim.ifi.uio.no/~trygver/1979/mvc-2/1979-12-MVC.pdf. Letzter Abruf 25. August 2007.

Ree79b Reenskaug, Trygve. The Original MVC Reports (February 1979). URL http://urn.nb.no/URN:NBN:no-14314. Letzter Abruf 25. August 2007.

Ree79c Reenskaug, Trygve. Thing-Model-View-Editor (May 1979). URL http://heim.ifi.uio.no/~trygver/1979/mvc-1/1979-05-MVC.pdf. Letzter Abruf 10. August 2007.

Rei85 Reisig, Wolfgang. Systementwurf mit Netzen. Springer, Berlin, Heidelberg, New York, Tokyo (1985).

Rei86 Reisig, Wolfgang. Petrinetze – Eine Einführung. Springer-Verlag, Berlin, Heidelberg, New York, Tokyo, 2., überarbeitete und erweiterte Aufl. (1986).

Rei10 Reisig, Wolfgang. Petrinetze – Modellierungstechnik, Analysemethoden, Fallstudien (2010). URL www2.informatik.hu-berlin.de/top/pnene_buch/pnene_buch.pdf. Letzter Abruf 22. Februar 2010.

RM07 Recker, Jan; Mendling, Jan. Adequacy in Process Modeling: A Review of Measures and a Proposed Research Agenda. In: Pernici, Barbara; Gulla, Jon Atle (Hrsg.), Proceedings of the 19th International Conference on Advanced Information Systems Engineering (CAiSE '07), S. 235–244. Trondheim (June 2007).

RRGI06 Recker, Jan; Rosemann, Michael; Green, Peter; Indulska, Marta. Extending the Scope of Representation Theory: A Review and a Proposed Research Model. In: Hart, Dennis N.; Gregor, Shirley D. (Hrsg.), Information Systems Foundations: Theory, Representation and Reality, S. 126–140. Canberra (September 2006).

RRIG06 Recker, Jan; Rosemann, Michael; Indulska, Marta; Green, Peter. Business Process Modeling: A Maturing Discipline? (2006). URL http://www.BPMCenter.org/. Letzter Abruf 12. Mai 2009.

RRIG09 Recker, Jan; Rosemann, Michael; Indulska, Marta; Green, Peter. Business Process Modeling - A Comparative Analysis. In: Journal of the Association for Information Systems JAIS, Bd. 10 (4), (2009), S. 333–363.

RSD08 Rosemann, Michael; Schwegmann, Ansgar; Delfmann, Patrick. Vorbereitung der Prozessmodellierung. In: Becker et al. BKR08, S. 45–103.

RtEv04a Russell, Nick; ter Hofstede, Arthur H. M.; Edmond, David; van der Aalst, Wil M. P. Workflow Data Patterns (Revised Version) (2004). URL http://www.bpm.fit.qut.edu.au/about/docs/DataPatternsRevised.pdf. Letzter Abruf 6. Juli 2009.

RtEv04b Russell, Nick; ter Hofstede, Arthur H. M.; Edmond, David; van der Aalst, Wil M. P. Workflow Resource Patterns (2004). URL http://is.tm.tue.nl/research/patterns/download/ResourcePatternsBETATR.pdf. Letzter Abruf 6. Juli 2009.

RtvM06 Russell, Nick; ter Hofstede, Arthur H. M.; van der Aalst, Wil M. P.; Mulyar, Nataliya. Workflow Control-Flow Patterns – A Revised View (2006). URL http://www.BPMCenter.org/. Letzter Abruf 6. Juli 2009.

Rüt90 Rütte, Richard. Prozessmanagement: Die Basis für ein effektives Projektmanagement. In: SAQ-Bulletin-ASPQ, Bd. 6, (1990), S. 18–22. Jg. 25.

RvtW06 Russell, Nick; van der Aalst, Wil M. P.; ter Hofstede, Arthur H. M.; Wohed, Petia. On the Suitability of UML 2.0 Activity Diagrams for Business Process Modelling. In: Stumptner, Markus; Hartmann, Sven; Kiyoki, Yasushi (Hrsg.), Proceedings of the 3rd Asia-Pacific Conference on Conceptual Modelling, Bd. 53 von *CRPIT*, S. 95–104. Australian Computer Society, Hobart (January 2006).

RW05 Rosemann, Michael; Wyssusek, Boris. Enhancing the Expressiveness of the Bunge-Wand-Weber Ontology. In: Proceedings of the Eleventh Americas Conference on Information Systems (AMCIS '05), S. 2803–2810. Omaha (August 2005).

Sch92 Scheer, August-Wilhelm. Architektur integrierter Informationssysteme – Grundlagen der Unternehmensmodellierung. Springer-Verlag, Berlin, Heidelberg, 2. Aufl. (1992).

Sch97 Schütte, Reinhard. Die neuen Grundsätze ordnungsmäßiger Modellierung (1997). URL http://www.pim.wiwi.uni-due.de/uploads/tx_itochairt3/publications/GoM_Forschungsforum.pdf. Letzter Abruf 21. Juni 2008.

Sch98a Scheer, August-Wilhelm. Wirtschaftsinformatik – Referenzmodelle für industrielle Geschäftsprozesse. Springer-Verlag, Berlin, Heidelberg, 2. Aufl. (1998).

Sch98b Schütte, Reinhard. Vergleich alternativer Ansätze zur Bewertung der Informationsmodellqualität (1998). URL http://www.wi-inf.uni-duisburg-essen.de/MobisPortal/pages/rundbrief/pdf/Schu98.pdf. Letzter Abruf 10. August 2008.

Sch99 Schütte, Reinhard. Literaturauffassungen zur Bewertung von Informationsmodellen. In: Kaschek, Roland (Hrsg.), Entwicklungsmethoden für Informationssysteme und deren Anwendung, EMISA 99, S. 175–195. Gesellschaft für Informatik, Teubner, Fischbachau (September 1999).

Sch07 Schauer, Carola. Rekonstruktion der historischen Entwicklung der Wirtschaftsinformatik (Mai 2007). URL http://www.icb.uni-due.de/fileadmin/ICB/research/research_reports/ICBReport18.pdf. Letzter Abruf 20. Februar 2010.

SF08 Stormer, Henrik; Frauchiger, Daniel. Aktuelle Entwicklungen elektronischer Shopsysteme. In: HMD – Praxis der Wirtschaftsinformatik, Bd. 261, (2008), S. 61–70.

Sim94 Simons, Gary F. Conceptual Modeling versus Visual Modeling: A Technological Key to Building Consensus. In: Consensus ex Machina, Joint International Conference of the Association for Literary and Linguistic Computing and the Association for Computing and the Humanities. Paris (April 1994).

Sim96 Simon, Herbert A. The Sciences of the Artificial. The MIT Press, third Aufl. (1996).

Smi98 Smith, Barry. An Introduction to Ontology. In: Peuquet, Donna; Smith, Barry; Brogaard, Berit (Hrsg.), The Ontology of Fields. Panel on Computational Implementations of Geographic Concepts, S. 9–14. National Center for Geographic Information and Analysis, Santa Barbara, Bar Harbor, Maine (June 1998).

Smi09 Smith, Barry. Ontology and Information Systems (2009). URL `http://ontology.buffalo.edu/ontology(PIC).pdf`. Letzter Abruf 13. Mai 2009.

Sta73 Stachowiak, Herbert. Allgemeine Modelltheorie. Springer-Verlag, Wien (1973).

Sta96 Stamper, Ronald. Signs, Information, Norms and Systems. In: Holmqvist, Berit; Andersen, Peter B.; Klein, Heinz; Posner, Roland (Hrsg.), Signs of Work: Semiosis and Information Processing in Organisations, S. 349–397. Walter de Gruyter & Co., Berlin (1996).

Stö09 Stöger, Roman. Prozessmanagement. Schäffer-Poeschel-Verlag, Stuttgart, 2., überarbeitete Aufl. (2009).

SVEH07 Stahl, Thomas; Völter, Markus; Efftinge, Sven; Haase, Arno. Modellgetriebene Softwareentwicklung. dpunkt, Heidelberg, 2., aktualisierte und erweiterte Aufl. (2007).

SW02 Schubert, Petra; Wölfle, Ralf (Hrsg.). E-Business erfolgreich planen und realisieren. Hanser Verlag, München (2002).

SZ99 Schütte, Reinhard; Zelewski, Stephan. Wissenschafts- und erkenntnistheoretische Probleme beim Umgang mit Ontologien. In: König,

Wolfgang; Wendt, Oliver (Hrsg.), Wirtschaftsinformatik und Wissenschaftstheorie 99 – Verteilte Theoriebildung, S. 1–19. Institut für Wirtschaftsinformatik, Frankfurt a. M. (Oktober 1999).

Tim05 Tim O´Reilly. What is Web 2.0 – Design Patterns and Business Models for the Next Generation of Software (2005). URL `http://www.oreillynet.com/pub/a/oreilly/tim/news/2005/09/30/what-is-web-20.html`. Letzter Abruf 27. April 2007.

TTL00 Tapscott, Don; Ticoll, David; Lowy, Alex. Digital Capital – Harnessing the Power of Business Webs. Harvard Business School Press (2000).

Val03 Valk, Rüdiger. Essential Features of Petri Nets. In: Girault und Valk GV03, S. 9–28.

vBtK00 van der Aalst, W. M. P.; Barros, A. P.; ter Hofstede, A. H. M.; Kiepuszewski, B. Advanced Workflow Patterns. In: Etzion, Opher; Scheuermann, Peter (Hrsg.), Proceedings of the 7th International Conference on Cooperative Information Systems (CoopIS 2000), Bd. 1901 von *Lecture Notes in Computer Science*, S. 18–29. Springer (September 2000).

vHPv07 van Bommel, Patrick; Hoppenbrouwers, Stijn J. B. A.; Proper, H. A. Erik; van der Weide, Theo P. QoMo: A Modelling Process Qualitiy Framework based on SEQUAL. In: Proper, Erik; Halpin, Terry; Krogstie, John (Hrsg.), Proceedings of the 12th International Workshop on Exploring Modeling Methods for Systems Analysis and Design (EMMSAD '07), S. 112–121. Trondheim (June 2007).

Vli96 Vlissides, John. Protection, Part I: The Hollywood Principle (Februar 1996). URL `http://researchweb.watson.ibm.com/designpatterns/pubs/ph-feb96.txt`. Letzter Abruf 24. August 2007.

vtKB03 van der Aalst, Wil M. P.; ter Hofstede, Arthur H. M.; Kiepuszewski, Bartek; Barros, Alistair P. Workflow patterns. In: Distributed and Parallel Databases, Bd. 14 (1), (2003), S. 5–51.

Web97 Weber, Ron. Ontological Foundations of Information Systems. In: Coopers & Lybrand Accounting Research Methodology Monograph, (4).

WSFM04 Werro, Nicolas; Stormer, Henrik; Frauchiger, Daniel; Meier, Andreas. eSarine – a Struts-based Webshop for Small and Medium-sized Enterprises. In: Proceedings of the EMISA Conference – Information Systems in E-Business and E-Government. Luxembourg (October 2004).

WvD+06 Wohed, Petia; van der Aalst, Wil M. P.; Dumas, Marlon; ter Hofstede, Arthur H. M.; Russell, Nick. On the Suitability of BPMN for Business Process Modelling. In: Dustdar, Schahram; Fiadeiro, José Luiz; Sheth, Amit (Hrsg.), Business Process Management. 4th International Conference, BPM 2006, Vienna, Proceedings, Bd. 4102 von *LNCS*, S. 161–176. Springer (September 2006).

Wys06 Wyssusek, Boris. On Ontological Foundations of Conceptual Modelling. In: Scandinavian Journal of Information Systems, Bd. 18 (1), (2006), S. 63–80.

Zel96 Zelewski, Stephan. Eignung von Petrinetzen für die Modellierung komplexer Realsysteme – Beurteilungskriterien. In: Wirtschaftsinformatik, Bd. 4, (1996), S. 369–381.