Gesellschaft für Informatik (GI)

publishes this series in order to make available to a broad public recent findings in informatics (i.e. computer science and information systems), to document conferences that are organized in cooperation with GI and to publish the annual GI Award dissertation.

Broken down into the fields of
- Seminars
- Proceedings
- Dissertations
- Thematics

current topics are dealt with from the fields of research and development, teaching and further training in theory and practice. The Editorial Committee uses an intensive review process in order to ensure the high level of the contributions.

The volumes are published in German or English.

Information: http://www.gi-ev.de/service/publikationen/lni/

ISSN 1617-5468
ISBN 978-3-88579-255-0

The contributions within this volume reveal a broad range of up to date research results in the field of modeling, e.g.: (domain specific) modeling languages and their usability, process modeling, model versioning and transformation, meta structures and model quality assurance. „Modellierung 2010“ is the 10th instance of the correspondent conference series which emphasizes on a mutual exchange of knowledge and experiences between academia and industry.

G. Engels, D. Karagiannis, H.C. Mayr, (Hrsg.): Modellierung 2010

161

GI-Edition

Lecture Notes in Informatics

Gregor Engels
Dimitris Karagiannis
Heinrich C. Mayr (Hrsg.)

Modellierung 2010

24.–26. März 2010
Klagenfurt

Proceedings

Gregor Engels, Dimitris Karagiannis, Heinrich C. Mayr (Hrsg.)

Modellierung 2010

24. – 26. März 2010
Klagenfurt, Österreich

Gesellschaft für Informatik e.V. (GI)

Lecture Notes in Informatics (LNI) - Proceedings
Series of the Gesellschaft für Informatik (GI)

Volume P-161

ISBN 978-3-88579-255-0
ISSN 1617-5468

Volume Editors
Prof. Dr. Gregor Engels
Universität Paderborn, Institut für Informatik
Warburger Str. 100, 33098 Paderborn, Deutschland
Email: engels@upb.de
Prof. Dr. Dimitris Karagiannis
Universität Wien, Institut für Knowledge und Business Engineering,
Brünner Straße 72, 1210 Wien, Österreich
Email: dimitris.karagiannis@univie.ac.at
Prof. Dr.Dr.h.c. Heinrich C. Mayr
Alpen-Adria-Universität Klagenfurt, Institut für Angewandte Informatik
Universitätsstraße 65 – 67, 9020 Klagenfurt, Österreich
Email: heinrich.mayr@uni-klu.ac.at

printed by Köllen Druck+Verlag GmbH, Bonn

Vorwort

Die Fachtagung "Modellierung" ist eine gemeinsame Veranstaltung einer Reihe von Fachgruppen der Gesellschaft für Informatik (GI), die sich mit Aspekten der Modellierung aus den verschiedensten Blickwinkeln auseinandersetzen. Sie wird in geraden Jahren, also im Zweijahresrhythmus, durchgeführt, alternierend mit dem gleichnamigen Workshop einschlägiger Forschungsgruppen im deutschsprachigen Raum.

Markenzeichen der „Modellierung“ sind wissenschaftliche Beiträge hoher Qualität, lebendige Diskussionen und engagierte Rückmeldungen aus Wissenschaft und Praxis: denn sie ist eine Plattform für die Begegnung und Vernetzung von Forschung und Anwendung. Aktuelle Themen werden darüber hinaus in speziellen Workshops behandelt. Ein Praxisforum, einschlägige Tutorien und ein Doktorandensymposium runden das Programm ab: zu einer umfassenden Synopse des „Stands der Kunst“ in der deutschsprachigen Modellierungslandschaft.

Dieser Band enthält die 22 Beiträge des Hauptprogramms der Modellierung 2010, ausgewählt aufgrund von jeweils drei Gutachten aus insgesamt 49 eingereichten Arbeiten. Während des Auswahlprozesses hatten die Autorinnen und Autoren Gelegenheit, zu den ihre Arbeit betreffenden Gutachten Stellung zu nehmen. Basierend hierauf, auf den Gutachten und auf einer regen Diskussion der Gutachtenden untereinander wurden schließlich 19 Lang- und 3 Kurzbeiträge ausgewählt. Dies entspricht einer Annahmerate der Langbeiträge von ca. 38,7%. Sie behandeln neueste wissenschaftliche Ergebnisse zu Themen wie: *Benutzerfreundlichkeit und Gebrauchstauglichkeit von Modellierungssprachen*, *Modellieren und Agentensysteme*, *Prozessmodellierung*, *UML*, *Objektorientierung*, *Qualitätssicherung von Modellen*, *Modellversionierung*, *Modelltransformation*, *Metamodelle*, *Metastrukturen*, *Referenzmodelle*, *Domänenspezifische Sprachen*.

Als eingeladener Sprecher konnte Wolfgang Reisig von der Humboldt-Universität in Berlin gewonnen werden. Sein Vortragstitel: „*50 Jahre Verhaltensmodellierung: Vom Modellieren mit Programmen zum Programmieren mit Modellen*“

Wir danken ihm und allen Beitragenden des Hauptprogramms, der Workshops, der Tutorien, des Praxisforums und des Doktorandensymposiums. Gleichermaßen gilt unser Dank den hierfür jeweils Verantwortlichen, sie haben sich mit hohem Engagement eingebracht. Besonderen Dank schulden wir dem bewährten Tagungs-Team der Forschungsgruppe „Application Engineering“ an der Universität Klagenfurt, allen voran dem unermüdlich wirkenden Christian Kop, sowie seinem Counterpart an der Uni Wien, Alexander Bergmayr. Dieser LNI-Band ist der erste mit einem etwas aufgefrischten Layout. Wir danken Judith Michael von der Universität Klagenfurt, dass Sie nicht nur die Vorlagen neu erstellt, sondern auch die eingereichten Papiere entsprechend angepasst hat.

Paderborn, Wien, Klagenfurt, im März 2010

Gregor Engels, Dimitris Karagiannis, Heinrich C. Mayr

Sponsoren

Wir danken den folgenden Unternehmen und Institutionen für die Unterstützung der Modellierung 2010.

BOC Information Technologies Consulting GmbH

Bundesministerium für Wissenschaft und Forschung

Förderverein Technische Fakultät an der Alpen-Adria Universität Klagenfurt

Industriellenvereinigung Kärnten

Alpen-Adria-Universität Klagenfurt

Querschnittsfachausschusses Modellierung

Die Plattform der GI zur Diskussion und zum Erfahrungsaustausch über aktueller und zukünftige Themen der Modellierungsforschung . Beteiligte GI-Gliederungen:

- EMISA, Entwicklungsmethoden für Informationssysteme und deren Anwendung
- FoMSESS Formale Methoden und Modellierung für SichereSysteme
- ILLS Intelligente Lehr- und Lernsysteme
- MMB Messung, Modellierung und Bewertung von Rechensystemen
- OOSE, Objektorientierte Software-Entwicklung
- PN Petrinetze
- RE Requirements Engineering
- ST Softwaretechnik
- SWA Softwarearchitektur
- WI-MobIS Informationssystem-Architektur: Modellierung betrieblicher Informationssysteme
- WI-VM Vorgehensmodelle für die Betriebliche Anwendungsentwicklung
- WM/KI Wissensmanagement

Tagungsleitung

Gesamtleitung:	Heinrich C. Mayr, Alpen-Adria-Universität Klagenfurt
Leitung des Programmkomitees:	Gregor Engels, Universität Paderborn
	Dimitris Karagiannis, Universität Wien
Workshops:	Wolfgang Hesse, Philipps-Universität Marburg
	Elmar Sinz, Otto-Friedrich-Universität Bamberg
Praxisforum:	Günther Müller-Luschnat, Pharmatechnik GmbH
DoktorandInnensymposium:	Ulrich Frank, Universität Duisburg-Essen
	Barbara Paech, Universität Heidelberg
Tutorien:	Jörg Desel, Katholische Universität Eichstätt
	Friederike Nickl, Swiss Life Deutschland

Programmkomitee

Colin Atkinson	Universität Mannheim
Thomas Baar	akquinet tech@spree GmbH, Berlin
Brigitte Bartsch-Spörl	BSR GmbH, München
Ruth Breu	Universität Innsbruck
Jörg Desel	Katholische Universität Eichstätt
Jürgen Ebert	Universität Koblenz-Landau
Ulrich Frank	Universität Duisburg-Essen
Martin Glinz	Universität Zürich, CH
Martin Gogolla	Universität Bremen
Andreas Heß	Capgemini sd&m AG, München
Wolfgang Hesse	Universität Marburg
Heinrich Hussmann	LMU München
Matthias Jarke	RWTH Aachen
Jan Jürjens	TU Dortmund und Fraunhofer ISST
Gerti Kappel	TU Wien
Roland Kaschek	Gymnasium Gerresheim, Düsseldorf
Ralf Kneuper	Darmstadt
Christian Kop	Alpen-Adria-Universität Klagenfurt
Thomas Kühne	Victoria University of Wellington,
Jochen Küster	IBM Research, Zürich, CH
Horst Lichter	RWTH Aachen
Peter Liggesmeyer	TU Kaiserslautern
Oliver Linssen	Liantis GmbH & Co. KG
Florian Matthes	TU München
Heinrich C. Mayr	Alpen-Adria-Universität Klagenfurt
Mark Minas	Universität der Bundeswehr München
Günther Müller-Luschnat	Pharmatechnik GmbH, München
Markus Nüttgens	Universität Hamburg
Andreas Oberweis	Universität Karlsruhe
Erich Ortner	Technische Universität Darmstadt
Barbara Paech	Universität Heidelberg

Programmkomitee (fortgesetzt)

Jan Philipps	validas AG, München
Udo Pletat	IBM Deutschland
Klaus Pohl	Universität Duisburg-Essen
Alexander Pretschner	TU Kaiserslautern und Fraunhofer IESE
Siegfried Reich	Salzburg Research
Ulrich Reimer	FH St. Gallen
Wolfgang Reisig	Humboldt-Universität zu Berlin
Werner Retschitzegger	Universität Linz,
Matthias Riebisch	Technische Universität Ilmenau
Bernhard Rumpe	RWTH Aachen
Bernhard Schätz	Technische Universität München
Peter Schmitt	Universität Karlsruhe
Andy Schürr	Technische Universität Darmstadt
Friedrich Steimann	Fernuniversität Hagen
Susanne Strahringer	TU Dresden
Michael von der Beeck	BMW AG
Gottfried Vossen	Universität Münster
Gerd Wagner	BTU Cottbus
Mathias Weske	HPI an der Universität Potsdam
Andreas Winter	Universität Oldenburg
Heinz Züllighoven	Universität Hamburg
Albert Zündorf	Universität Kassel

Organisationsteam

Xulian Benesch	Universität Wien
Alexander Bergmayr	Universität Wien
Günther Fliedl	Alpen-Adria-Universität Klagenfurt
Doris Gälle	Alpen-Adria-Universität Klagenfurt
Christian Kop	Alpen-Adria-Universität Klagenfurt
Alexander Kopper	Alpen-Adria-Universität Klagenfurt
Stefan Leitner	Alpen-Adria-Universität Klagenfurt
Judith Michael	Alpen-Adria-Universität Klagenfurt
Heidi Scherzer	Alpen-Adria-Universität Klagenfurt
Christine Seger	Alpen-Adria-Universität Klagenfurt
Claudia Steinberger	Alpen-Adria-Universität Klagenfurt
Jürgen Vöhringer	Alpen-Adria-Universität Klagenfurt

Inhalt

Objektorientierung und UML

Ansätze zur Qualitätssicherung (Kurzbeiträge)

Versionierung und Modelltransformation

Metamodelle, Metastrukturen, Referenzmodelle und Domänenspezifische Sprachen

50 Jahre Verhaltensmodellierung: Vom Modellieren mit Programmen zum Programmieren mit Modellen

Wolfgang Reisig[1]

Zusammenfassung

Software entwickeln bedeutet, die Kluft zwischen algorithmischen Ideen und implementierten Programmen zu überbrücken. Dabei helfen Modelle: Ein adäquates Modell einer algorithmische Idee repräsentiert die relevanten Aspekte der Idee verständlich und unmittelbar einsichtig. Ein gutes Modell vermeidet Zusätze oder Abstriche, die lediglich der verwendeten Modellierungstechnik geschuldet wären.

In diesem Beitrag konzentrieren wir uns auf Modelle zur Beschreibung des dynamischen Verhaltens von Systemen. Während „Datenmodellierung“ ein aus der Datenbanktheorie wohlbekanntes Konzept ist, gibt es für das Modellieren von Verhalten keinen gleichermaßen etablierten Begriff. Ein solcher Begriff würde weit mehr als den Kontroll- und Datenfluss umfassen und auch generelle Fragen an das Konzept des „Algorithmus“ stellen.

Eine Modellierungstechnik stellt Ausdrucksmittel bereit, um eine spezifische Art von Modellen zu charakterisieren. Eine zweckmäßige Modellierungstechnik bietet außerdem ausdrucksstarke Verfahren zur Analyse dieser Modelle, d.h. zum Nachweis spezieller Eigenschaften.

Wir konzentrieren uns hier auf Techniken zur Verhaltensmodellierung, die ausdrucksstarke Analyseverfahren bereitstellen. Wir streifen ihre historische Entwicklung, versuchen eine Klassifikation solcher Techniken gemäß signifikanter Kriterien, und fragen nach den wichtigsten Gestaltungsprinzipien zukünftiger Modellierungstechniken.

[1] Humboldt-Universität zu Berlin, Institut für Informatik, Unter den Linden 6, 10099 Berlin, reisig@informatik.hu-berlin.de

Ein generischer Ansatz zur Messung der Benutzerfreundlichkeit von Modellierungssprachen

Christian Schalles[1], Michael Rebstock[2] und John Creagh[3]

Abstract: Eine Ermittlung der Benutzerfreundlichkeit im Sinne der Usability von Modellierungssprachen war bisher nicht Zielsetzung empirischer Evaluationsstudien. In den meisten Usabilitystudien wurden und werden Applikationen, Webseiten und technische Produkte evaluiert. Ziel dieses Beitrags ist die Schaffung eines Rahmenkonzeptes zur Bewertung der Usability von Modellierungssprachen. Es ist als Beitrag zu verstehen, der die komplexen Zusammenhänge einer Usabilitystudie für Modellierungssprachen erarbeitet und eine Grundlage für daran anknüpfende empirische Untersuchungen schafft.

1 Einleitung

Die moderne Informationsgesellschaft ist ohne komplexe Anwendungssysteme nicht zu denken. Zur Unterstützung einer effizienten Entwicklung und Pflege sowie zum Management komplexer Systeme bieten die Methoden des Software Engineering eine geeignete Grundlage [SDJ07]. Eines der Grundkonzepte des Software Engineering stellt die Modellierung von Anwendungssystemen dar. Modelle können auf allen Gebieten und in allen Methoden des Software Engineering aufgefunden werden [Lu03]. Um eine vollständige und korrekte Interpretation eines Modells gewährleisten zu können ist es wichtig, dass das Modell das enthaltene Wissen strukturiert und geordnet darstellt. Eine korrekte Modellinterpretation kann nur sichergestellt werden, wenn potentielle Benutzer das Modell verstehen. Hierfür sollten Sie die Modellierungssprache, mit der das Modell erstellt wurde, beherrschen [MeSt08]. Aber auch das Modellieren an sich, also die Modellentwicklung, wird von der zu Grunde liegenden Modellierungssprache beeinflusst. Die Softwaremodellierung sieht eine Vielzahl an Methoden und Sprachen für das Erstellen von Modellen vor [Ga07].

Die Entscheidung für oder gegen den Einsatz einer bestimmten Modellierungssprache wird von einer Vielzahl von Kriterien, wie z.B. technologische, funktionale, ökonomische und usability-induzierte beeinflusst [MaEr05]. Vor allem usability-induzierte Kriterien rücken hier in den Vordergrund, da die Benutzerakzeptanz stark mit der Usability korreliert [Sc04]. Avison und Fitzgerald (1995) analysieren zwei Motive für die Bewertung von Modellierungssprachen: 1) der akademische-forschungsrelevante Ansatz unterstützt das Verständnis sowie darauf aufbauend die Weiterentwicklung von Modellierungssprachen; 2) der unternehmens-managementbezogene Ansatz umschreibt

[1] Department of Computing, Cork Institute of Technology, Cork, Ireland, christian.schalles@mycit.ie
[2] Faculty of Economics and Business Administration, Hochschule Darmstadt University of Applied Sciences, Haardtring 100, 64295 Darmstadt, michael.rebstock@h-da.de
[3] Department of Computing, Cork Institute of Technology, Cork, Ireland, john.creagh@cit.ie

die Unterstützung bei der Auswahl geeigneter Modellierungssprachen für einzelne Organisationsbereiche sowie für das gesamte Unternehmen [AvFi95].

In diesem Beitrag setzen wir uns die Entwicklung eines Rahmenkonzeptes als Grundlage für eine darauf aufbauende Untersuchung der Benutzerfreundlichkeit von grafischen Modellierungssprachen als Schwerpunkt. Hierzu werden wir in Kapitel 2 eine für Modellierungssprachen allgemeingültige Definition des Begriffs „Usability“ methodisch herausarbeiten. Anschließend werden die verschiedenen Attribute der Benutzerfreundlichkeit extrahiert und an Modellierungssprachen angepasst. Des Weiteren werden Methoden, die eine Messung der verschiedenen Attribute unterstützen aufgezeigt. Im letzten Schritt werden die Ergebnisse zu einem Rahmenkonzept zusammengesetzt.

2 Benutzerfreundlichkeit und ihre heterogenen Definitionen

Die Benutzerfreundlichkeit oder Usability wird weder von Forschungsgruppen noch von Standardisierungsorganisationen wie beispielsweise der International Organization for Standardization (ISO) einheitlich definiert [Ni06]. Die hieraus resultierende Vielfalt an Usabilitydefinitionen und Messmodellen erschwert das Erstellen einer einheitlichen Usabilitydefinition für Modellierungssprachen. Eine Usabilitystudie, die nicht auf einer Standarddefinition basiert, würde nach Coursaris und Kim (2006) wenig Sinn ergeben [CoKi06]. Dies veranlasst uns verschiedene Standarddefinitionen zu analysieren und identische bzw. ähnliche Kriterien zu extrahieren und zu einer an Modellierungssprachen adaptierten einheitlichen Definition zusammenzusetzen.

Die ISO versteht in dem Standard ISO 9241-110 Usability als Leistung einer Software. Diese userbezogene Leistung bezieht sich auf Verständlichkeit, Erlernbarkeit und Nutzerzufriedenheit [ISO06]. Des weiteren existiert ein weiterer Standard ISO 9241-11, welcher den Nutzer und seine spezifische Zufriedenheit sowie die Möglichkeit effizient und effektiv mit dem zu evaluierenden Produkt zu arbeiten zentriert [ISO98]. Das Institute of Electrical and Electronics Engineers (IEEE) definiert einen Standard, worin Usability mit den Attributen Erlernbarkeit und Verständlichkeit umschrieben wird. Hiernach ist es ein wichtiges Kriterium für Usability, wie einfach eine Anwendung oder vergleichbare Artefakte erlernt und verstanden werden können [IEEE90]. Dumas and Redish (1999) beschreiben Usability anhand der Möglichkeit eines Benutzers eine vorgegebene Aufgabe schnell und einfach zu erfüllen.

Diese Definition basiert auf vier Annahmen [DuRe99]: 1. Usability zentriert den Benutzer, 2. Usability äußert sich in einer produktiven Aufgabenerfüllung, 3. Usability stützt sich auf eine einfache Nutzungsmöglichkeit, 4. Usability wird durch die Möglichkeit, effizient zu arbeiten, erzeugt. Shackel (1991) verbindet fünf Attribute einer benutzerfreundlichen Anwendung: Geschwindigkeit, Erlernbarkeit, Fehlertoleranz, Aufgabenerfüllungsgrad und die benutzerspezifische Einstellung zu einer Anwendung [Sh91]. Preece et al. (1994) fassen Effetivität und Effizienz unter Produktivität zusammen [Pr94]. Constantine und Lockwood (1999) und Nielsen (2006) erstellen eine

Sammlung an Usabilityattributen und entwickeln auf dieser Grundlage fünf domänenübergreifende Usabilityattribute: Erlernbarkeit, Einprägsamkeit, Effektivität, Effizienz und Benutzerzufriedenheit [CoLo99], [Ni06]. Die gezeigte Vielfalt an Usabilitydefinitionen und daraus resultierenden Usabilityattributen führt zu dem Gebrauch verschiedenster Ausdrücke für identische Uabilitycharakteristiken. Tabelle 1 gibt einen kurzen Überblick über die in der Literatur am meisten auftauchenden Usabilityattribute:

	[IEEE90]	[Sh91]	[Pr94]	[ISO98]	[CoLo99]	[DuRe99]	[ISO06]	[Ni06]
Effizienz		x		x	x	x		x
Erlernbarkeit	x	x	x	x	x		x	x
Einprägsamkeit		x			x			x
Effektivität		x		x	x	x		x
Benutzerzufriedenheit				x	x			x

Tab. 1: Usabilityattribute verschiedener Definitionen

3 Ein Rahmenkonzept zur Messung der Benutzerfreundlichkeit von Modellierungssprachen

3.1 Usabilityattribute für Modellierungssprachen

In diesem Kapitel wird ein Rahmenkonzept erstellt, das als Grundlage einer künftigen Evaluation der Usability von Modellierungssprachen dient.

Im Bereich des Software Engineering kommen verstärkt Diagramme der Unified Modeling Language (UML) zum Einsatz. Zur Modellierung von Geschäftsprozessen im Sinne der Prozessmodellierung wird vor allem auf EPK und BPMN Diagramme zurückgegriffen [KST07]. Die genannten Modellierungssprachen sind grafische Sprachen, die insbesondere eine benutzerfreundliche grafikgestützte Anwendungs- und Geschäftsprozessmodellierung ermöglichen sollen [SuMe09]. Diese Tatsache unterstützt unser Vorhaben und rückt den Benutzer grafischer Modellierungssprachen in den Vordergrund [Ni06], [KST07]. Eine auf unserem Rahmenkonzept aufbauende Usabilitystudie ermöglicht eine künftige benutzernahe Entwicklung und Ausgestaltung grafischer Modellierungssprachen.

Unter Berücksichtigung der in Kapitel 2 analysierten Usabilitydefinitionen wird im Anschluss eine Extraktion von Attributen durchgeführt. Aufbauend wird eine einheitliche generische Usabilitydefinition für grafische Modellierungssprachen entwickelt. Das zu entwickelnde Rahmenkonzept sieht die Anwendung von fünf in den analysierten Quellen genannten Usabilityattributen vor: Erlernbarkeit, Einprägsamkeit, Effektivität, Effizienz und Benutzerzufriedenheit [CoLo99], [Ni06]. Nach Sibert und Jacob (2000) gibt die visuelle Wahrnehmung zusätzliche Informationen über die

Usability einer Anwendung [SiJa00]. Im Umfeld des Usability-Engineering wird die Usability von Bildschirmlayouts mit der Methodik des Eye-Tracking gemessen. Die bereits genannten Usabilityattribute liefern keinen Aufschluss über die visuelle Wahrnehmung seitens der Benutzer [PCV05]. Die Erweiterung um ein weiteres Usabilityattribut kann interessante Informationen über die visuelle Wahrnehmung von Modellierungssprachen und damit generierten Diagrammen hervorbringen [KEC99]. Um dies zu gewährleisten wird das sechste Usabilityattribut, die Wahrnehmbarkeit von Modelldiagrammen, in das Rahmenkonzept integriert. Im weiteren Verlauf dieses Kapitels wird jedes Attribut und seine Stellung in unserem Rahmenkonzept definiert:

Die **Erlernbarkeit** einer Modellierungssprache ist ein wichtiges Attribut für die Usabilityevaluation von Modellierungssprachen. Das Erlernen der praktischen Anwendung ist die erste Erfahrung mit der ein Benutzer einer neuen unbekannten Modellierungssprache konfrontiert wird [SiRo08], [Ma89]. Das Attribut **Einprägsamkeit** beschreibt die Möglichkeit, dass ein Benutzer, der bereits eine Modellierungssprache gelernt hat auch nach längerer Abstinenz diese wieder erfolgreich anwenden kann. Generell sollte es einem Benutzer ermöglicht sein, sich an die verschiedenen Elemente und die Syntax der Sprache einfach erinnern zu können [Ma89], [ReDr07]. Des Weiteren sollte einem Nutzer eine erfolgreiche zielorientierte Erfüllung bestimmter Modellierungsaufgaben durch die Modellierungssprache selbst ermöglicht werden. Dies bezüglich sollte ein Nutzer im Sinne der Effektivität Modelle mit möglichst wenigen syntaktische Fehlern erstellen bzw. interpretieren [Bo05a], [WaWe93]. **Effizienz** bezieht sich auf Benutzer, die über eine mittelmäßige bis hohe Modellierungserfahrung verfügen. Wenn ein Nutzer bereits eine Modellierungssprache erlernt hat, sollte eine effiziente Anwendung ermöglicht werden können. Eine Modellierungssprache ist effizient in ihrer Anwendung, wenn Nutzer Modelle schnell und mit möglichst wenigen syntaktischen Fehlern erstellen können [Bo05b], [WaWe93]. Das Attribut **Benutzerzufriedenheit** zielt auf den Nutzer und seine individuelle Zufriedenheit während des Modellierens bzw. Interpretierens von Modellen ab [SiWa07]. Die **visuelle Wahrnehmbarkeit** ist ein sehr wichtiges Attribut bei der Usabilityevaluation von Anwendungen und in unserem Zusammenhang Modellierungssprachen [Go04], [DMD08], [EhWi07], [Pr05]. In den meisten Fällen ist dieses Attribut weitestgehend bei der Interpretation von Modellen relevant. Die visuelle Wahrnehmbarkeit kann durch den Einsatz der Eye-Tracking-Methode gemessen werden. Bei dieser Technik werden zum einen Augenbewegungen und zum anderen einzelne Fixationspunkte registriert [Na01]. Bezüglich unseres Rahmenkonzeptes kann der Einbezug der Eye-Tracking-Methode zusätzlichen Aufschluss über nutzerbezogene kognitive Prozesse wie die Suche nach Information in einem Modell, die Aufnahme der gefundenen Information sowie die Verarbeitung der aufgenommenen Information geben [FSM50], [JaKa03].

Im Folgenden wird eine allgemeingültige Definition von Usability erstellt und auf Modellierungssprachen adaptiert:

Die Usability von Modellierungssprachen wird durch die Attribute Erlernbarkeit, Einprägsamkeit, Effektivität, Effizienz, Benutzerzufriedenheit sowie visuelle

Wahrnehmbarkeit definiert. Die Erlernbarkeit von Modellierungssprachen beschreibt die Tauglichkeit einer Sprache einfach und schnell erlernt werden zu können. Es sollte für einen Nutzer einfach sein, sich an die Sprache und ihre im Metamodell definierten verschiedenen Elemente und syntaktischen Regelungen auch nach längerer Abstinenz zu erinnern. Eine effektive, d.h. fehlerfreie Aufgabenerfüllung sollte durch die Modellierungssprache unterstützt werden. Weiterhin sollte ein hohes Maß an Produktivität zur Unterstützung einer effizienten Ausführung von Modellierungsaufgaben ermöglicht werden. Die Benutzung der Modellierungssprache sollte angenehm ausfallen. Nutzer sollten mit der Anwendung der Sprache zufrieden sein. Letztendlich sollte die Modellierungssprache eine einfache, geordnete und dadurch bequeme visuelle Wahrnehmbarkeit hinsichtlich Struktur, Gesamteindruck, Elemente, Shapes und Farben besitzen.

3.2 Zu berücksichtigende Variablen

Für die Entwicklung eines Rahmenkonzeptes müssen die für die Usabilityevaluation von Modellierungssprachen in Frage kommenden Variablen definiert und strukturiert werden. Generell können drei Variblentypen spezifiziert werden: abhängige, unabhängige und zu kontrollierende Variablen [BaLi02]. Die relevanten Variablen und ihre Relationen sind in Abbildung 1 dargestellt. Die jeweiligen Definitionen sowie die Zuordnung zu den jeweiligen Variablentypen werden im Anschluss durchgeführt.

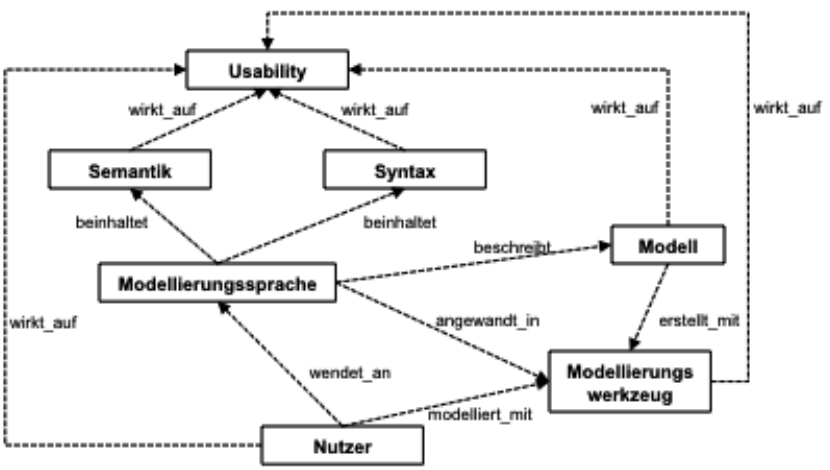


Abb. 1: Relevante Variablen im Kontext

Modelle sind abstrakte und immaterielle Bilder, die reale Domänen repräsentieren. In Modellen wird die reale Welt auf relevante Teilabschnitte reduziert [St73]. Ein sehr wichtiges Ziel der Modellierung ist das Erreichen einer Komplexitätsreduzierung durch Abstrahierung der Realität [BWW03]. So ist beispielsweise das Ziel von

Businessmodellen komplexe Szenarien vereinfacht darzustellen [BrZa09]. Der Schwierigkeits- und Komplexitätsgrad eines spezifischen Modells wirkt sich auf die Verständlichkeit und folglich auf die Usability der angewandten Modellierungssprache aus [MMR09]. Bei der Ermittlung der Usability von Modellierungssprachen muss die Komplexität des einzelnen zu erstellenden oder zu interpretierenden Modells separat erfasst und kontrolliert werden. Somit werden mögliche verfälschende Effekte auf das Gesamtergebnis vermieden.
Die Entwicklung eines Modells wird mit einer bestimmten Modellierungssprache durchgeführt. Es gibt grafische und textbasierte Modellierungssprachen. Das Rahmenkonzept, das in diesem Beitrag vorgestellt wird stützt sich auf grafische Modellierungssprachen wie EPK's, UML-Sprachen oder BPMN. Eine Modellierungssprache legt fest, wie und nach welchen Konventionen etwas logisch in einem Modell ausgedrückt werden kann [Kr07].
Der Benutzer modelliert mit einer bestimmten Modellierungssprache unter Anwendung eines Modellierungswerkzeugs. Das Modellierungswerkzeug wirkt zusätzlich auf die zu untersuchende Usability und muss daher als Kontrollvariable behandelt werden [Ni06]. Aus diesem Grund muss zwischen Spracheinflüssen und Werkzeugeinflüssen auf die Usability unterschieden werden [BaLi02].
Modellierungssprachen basieren auf Metamodellen, welche die Semantik und die Syntax der Sprache festlegen [KaKü02]. Unter Syntax subsumieren sich alle Regeln zum formalen Aufbau einer Sprache inklusive der syntaktischen Notation. Die hierin enthaltenen verschiedenen Elemente unterscheiden sich je nach Modellierungssprache in Farben und Shapes [Ha05]. Die Semantik stellt die inhaltliche Bedeutung von Sätzen und Wörtern der Sprache dar [Da03].
Die Benutzer von Modellierungssprachen unterscheiden sich hinsichtlich ihrer individuellen Modellierungserfahrung. Diese Tatsache beeinflusst usabilitybezogene Messungen [Ni06]. Folglich muss der Nutzer und seine individuelle Erfahrung im Umgang mit Modellierungssprachen als Kontrollvariable behandelt werden.
Der Benutzer kann generell zwei unterschiedlichen Situationen ausgesetzt sein: Softwarearchitekten und Prozessmodellierer werden vorzugsweise mit der Modellerstellung, also der Modellierung an sich konfrontiert werden. Andere Nutzergruppen wie z.B. Programmierer werden hauptsächlich bereits bestehende Modelle interpretieren d.h. die Modelle als Informationsquellen nutzen [SiWa07]. Somit können, je nachdem in welcher Situation sich der jeweilige Benutzer befindet, unterschiedliche Anforderungen definiert werden. Ein Modellierer stellt im Sinne der Usability folgende Anforderungen an eine Modellierungssprache:

- Schnelle und einfache Erlernbarkeit
- Einfaches Erinnern an die verschiedenen Elemente und Syntax der Sprache
- Ermöglichen einer effizienten Erfüllung bestimmter Aufgaben
- Zufriedenheit mit der jeweiligen Modellierungssprache

Betrachter von Modellen müssen den Prozessfluss und die Modellstruktur erkennen. Aus diesem Grund werden von einem Betrachter von Modellen folgende Anforderungen an

Modellierungssprachen gestellt: Intuitives und eindeutiges Modell hinsichtlich der Modellstruktur, Shapes und der Syntax.
Um eine Usabilitybewertung von Modellierungssprachen durchzuführen ist es wichtig, zwischen diesen beiden Situationen zu unterscheiden [SiWa07].
Zusammenfassend ist zu sagen, dass die Syntax, die Semantik, das Modell, das Modellierungswerkzeug und der Benutzer Variablen sind, die für eine Usabilitybewertung von Modellierungssprachen herangezogen werden sollten [Bo05a], [KHS05].

3.3 Die Entwicklung von Usabilitymetriken

Um eine Usabilitystudie durchzuführen ist es notwendig Metriken zu definieren [Se06]. In unserem Fall müssen Metriken für die verschiedenen in Kapitel 3.1 extrahierten Attribute entwickelt werden. Einige Metriken sind komplexer und werden in mathematischen Termen ausgedrückt während andere Metriken einfache quantifizierte Daten darstellen. Im Folgenden werden Usabilitymetriken für Modellierungssprachen auf Basis der in Kapitel 3.1. definierten Attribute generiert. Für die Bewertung der Usability von Modellierungssprachen wird zusätzlich die Eye-Tracking-Methode herangezogen. Dieses weitere Attribut erscheint bezüglich der Betrachtung und Interpretation von Modellen als sehr wichtig [Go04], [DMD08], [EhWi07], [Pr05].

Zur Operationalisierung von Effektivität muss der Output durch eine Messung von Quantität und Qualität der Zielerreichung einer Aufgabe festgehalten werden [Re93]. In der Literatur wird Quantität als das Verhältnis zwischen dem erreichtem Output einer Aufgabe und den eigentlichen Aufgabenzielen messbar gemacht. Die Qualität beschreibt in diesem Zusammenhang die eigentliche Zielerreichung [BeMa94]. Bevan (1995) definierte Effektivität als das Produkt aus Qualität und Quantität [Be95]. Überträgt man diesen Sachverhalt auf unser Rahmenkonzept lässt sich Effektivität anhand folgender Formel ausdrücken, wobei die Anzahl an Knoten und Kanten als Maßzahlen für den Fertigstellungsgrad sowie die jeweiligen Aufgabenziele herangezogen werden [Ka02]:

$$Effektivität\ (F) = \frac{\sum_{i=1}^{n}(N_{Aufgabe,i} + E_{Aufgabe,i})}{\sum_{i=1}^{n}(N_{Ziele,i} + E_{Ziele,i})} * \frac{\sum_{i=1}^{n}(N_{Ziele,i} + E_{Ziele,i} - R_i)}{\sum_{i=1}^{n}(N_{Ziele,i} + E_{Ziele,i})} \quad (1)$$

N=Knoten, E=Kanten, R=Fehler

Die Effizienz wird von menschlichen, ökonomischen und chronologischen Ressourcen beeinflusst. Effizienzmetriken berechnen sich aus der Effektivität unter Berücksichtigung des Ressourcenverbrauchs [BeMa94]. Hieraus resultierende Metriken beinhalten zu einem Großteil die Ressource Zeit bzw. die zur Aufgabenerfüllung benötigte Zeit [Vu08].

Somit kann dieser Bezug durch das Beziehen der Effektivität auf die benötigte Zeit zur

Aufgabenerfüllung ausgedrückt werden:

$$Effizienz\ (G) = \frac{F}{T} \tag{2}$$

F=Effektivität, T=benötigte Zeit zur Erledigung einer Aufgabe

Die Erlernbarkeit umschreibt den Lernaufwand der für das Erlernen der Syntax einer Modellierungssprache benötigt wird. Sowohl das Zeitverhalten (benötigte Zeit für die Aufgabenerfüllung) als auch die Fehlerfreiheit sind relevante Messelemente zur Bestimmung der Erlernbarkeit einer Modellierungssprache [Se06]. Innerhalb einer Untersuchung ist die Erlernbarkeit ein Prozess, der am besten durch Lernkurven beschrieben werden kann [TKM08]. Aus diesem Sachverhalt heraus kann die Erlernbarkeit durch mindestens zwei zeitlich versetzte Datenerhebungen und die hieraus zu berechnende Differenz bestimmt werden [Be95]. Nielsen (2006) behauptet, dass Anwendungen, die als gut erlernend eingestuft werden dem Nutzer kurze Erlernzeiten ermöglichen sollten [Ni06]. Des weiteren schlägt Nielsen (2006) Metriken wie beispielsweise die Quantität, Qualität oder das Zeitverhalten zur Messung der Erlernbarkeit vor [Ni06]. Die im vorigen Absatz eingeführte Effizienzmetrik beinhaltet alle von Nielsen genannten Teilmetriken und kann daraus als ein Indikator für die Erlernbarkeit einer Modellierungssprache betrachtet werden. Die Differenz dieser zeitlich verteilten Effizienzmesswerte gibt Aufschluss über die Erlernbarkeit einer Modellierungssprache. So kann bei zwei zeitlich versetzten Messpunkten mp und mp+1 die relative Abweichung Δ festgestellt werden [NeNa02], [GFA09]:

$$\Delta Erlernbarkeit = \frac{G_{mp+1} - G_{mp}}{G_{mp}} \tag{3}$$

G=Effizienz, mp=Messpunkt

Die Einprägsamkeit einer Modellierungssprache kann am besten als Output nach einer angemessenen Abstinenzphase gemessen werden. Der Benutzer sollte in diesem Fall die Sprache bereits erlernt haben [Ni06]. Die Abstinenzphase sollte etwa Minuten für einfache Elemente, Stunden für einfache syntaktische Regelungen und Wochen für eine vollständige Modellierungssprache umfassen [Se06]. Folglich setzen sich die Messwerte für die Einprägsamkeit aus Vergessenskurven, die aus zeitversetzten Wissensabfragen ermittelt werden können zusammen [NeUz00]. Die Einprägsamkeit von Modellierungssprachen stützt sich auf die verschiedenen Elemente sowie die Syntax. Eine gute Methode für die Messung der Einprägsamkeit von Modellierungssprachen stellen Wissenstests mit Fragen über die verschiedenen Elemente, Syntax und deren Anwendung innerhalb einer Modellierungssprache dar.

Eine Möglichkeit zur Messung der visuellen Wahrnehmbarkeit von Modellierungssprachen stellt die Eye-Tracking-Methode dar [Go04]. Eye-Tracking wurde von Fitts et al. (1950) zum ersten Mal wissenschaftlich untersucht und angewandt [FJM50]. Es gibt bezüglich des Eye-Trackings eine Vielzahl an Messwerten.

Nach methodischer Analyse dieser Messwerte haben wir uns innerhalb unseres Rahmenkonzepts für folgende Metriken entschieden: Fitts et al. (1950) schlagen als Messwert für die Schwierigkeit der Informationsgewinnung bzw. Interpretation eines Betrachtungsgegenstandes die zeitliche Länge der gemessenen Fixation vor. Während einer Fixation ist das Auge auf einen Bereich fixiert und somit eine Informationsaufnahme möglich. Die Dauer der Fixation hängt davon ab, wie schwierig oder einfach die Verarbeitung des Betrachteten abläuft. Wenn die Information schwieriger zu verarbeiten ist wird die Fixation länger andauern beziehungsweise werden in diesem Bereich gehäuft Fixationen auftreten [Du07]. Die Bewegungen zwischen den Fixationen werden Sakkaden genannt und können als schnelle Blicksprünge von einer Fixation zur nächsten bezeichnet werden. Innerhalb dieses Blicksprungs werden keine visuellen Informationen zum Gehirn gesendet. Eine Informationsaufnahme ist hier nicht möglich [JaKa03]. Somit beschreiben Fixiationen den kognitiven Prozess der Informationsextraktion und -verarbeitung während Sakkaden den Prozess der Informationssuche indizieren [PoBa05]. In unserem Rahmenkonzept deutet eine hohe aggregierte Sakkadenlänge auf eine intensive Suche hin [GoKo99]. Dies erschwert die Interpretation eines Modells. Je höher das Verhältnis aus Sakkadenlänge in Sekunden und Fixationslänge in Sekunden desto schlechter ist die visuelle Wahrnehmbarkeit einer Modellierungssprache. Hieraus folgt folgende Metrik:

$$Wahrnehmbarkeit = \frac{T_{Sak}}{T_{Fix}} \tag{4}$$

T_{Sak}=*Sakkadenlänge,* T_{Fix}=*Fixationslänge*

Der entscheidende Nutzen der Eye-Trackingmethode innherhalb dieser Untersuchung stützt sich auf die Analyse kognitiver Prozesse während der Modellinterpretation. Es können somit Rückschlüsse gezogen werden, in welchen Phasen der Interpretation (Informationssuche, -aufnahme, -verarbeitung) welche Art von sprachenspezifischen Barrieren auftreten, die eine Modellinterpretation erschweren. Weiterhin können heterogene Modelle bezüglich ihrer visuellen Wahrnehmbarkeit in Beziehung gesetzt werden.

Im Vergleich zu den anderen in diesem Beitrag vorgestellten Attributen ist die Benutzerzufriedenheit ein nutzerbezogenes individuelles Kriterium. Die Zufriedenheit eines Nutzers mit einer Modellierungssprache kann durch Interviews, Fragebögen oder durch Verhaltensbeobachtungen während der Modellierung bzw. Modellbetrachtung analysiert werden [Vu08]. Van Schaik und Ling (2007) schlagen die Anwendung der so genannten Visual Analogue Scale (VAS) vor. Die VAS ist eine graduierte Skala, auf der (beispielsweise) die individuelle Zufriedenheit mit einer Modellierungssprache seitens eines Nutzers evaluiert werden kann [VaLi07].

Zur Bestimmung der Usability von Modellierungssprachen ist es weiterhin notwendig die Variablen aus Kapitel 3.2. zu operationalisieren und damit messbar zu machen. Die Komplexität einer Modellierungssprache, d.h. insbesondere die Komplexität des Metamodells einer Modellierungssprache wirkt auf die verschiedenen Usabilityattribute.

Zur Analyse der Sprachkomplexität entwickelten Welke (1992) und darauf aufbauend Rossi und Brinkkemper (1996) Metriken, die auf dem OPRR Datenmodell basieren [We92], [RoBr96]. Danach kann eine Modellierungssprache als ein Sechstupel M={O,P,R,X,r,p} basierend auf dem OPRR (Object, Property, Relationship, Role) – Modell definiert werden.

Unter Berücksichtigung der Arbeiten von Rossi und Brinkkemper (1996) und Recker et al. (2009) haben wir folgende drei Metriken für unser Rahmenkonzept extrahiert [Re09], [RoBr96]:

- Anzahl der verschiedenen Elemente (E)
- Anzahl der Properties (P)
- Anzahl möglicher Beziehungstypen (R)

Die folgende Kennzahl wurde in Anlehnung an Rossi and Brinkkemper 1996 erstellt und definiert einen Komplexitätsvektor innerhalb eines 3-dimensionalen Koordinatensystems:

$$Sprachenkomplexität = \sqrt{E^2 + R^2 + P^2} \tag{5}$$

Unter besonderer Berücksichtigung visueller Faktoren, welche die Usability von Modellierungssprachen beeinflussen sind metamodellbasierende Metriken wie beispielsweise die Anzahl unterschiedlicher Elementgeometrien (Shapes) sowie die Anzahl unterschiedlicher Farben zu nennen [ElSc01].

4 Verwandte Arbeiten

In den vergangenen Jahren evaluierte eine Vielzahl an Forschungsgruppen Modellierungssprachen unter verschiedensten Gesichtspunkten. Die Ergebnisse stellen sehr oft Verbesserungsvorschläge für verschiedene Sprachen dar. Einerseits stützen sich diese Evaluationen auf empirische Datenaufnahmen und andererseits auf theoretische Bewertungen. Die verschiedenen Studien lassen sich drei Hauptkategorien zuordnen: 1) Vergleichsstudien, 2) Theoretisch-konzeptionelle Evaluationsstudien, 3) Empirische Evaluationsstudien.

1) Vergleichsstudien integrieren zu einem Großteil verschiedene Modellierungssprachen, die alle für die Modellierung einer einheitlichen Domäne angewendet werden. Hierbei wird analysiert, wie unterschiedliche Modellierungssprachen ein Modellierungsproblem bewältigen [OSV86], [BaHe00], [Lo90], [St86].

2) Theoretisch-konzeptionelle Evaluationsstudien formalisieren den Evaluationsprozess durch die Anwendung von Rahmenkonzepten (Frameworks) und anderen Referenzdisziplinen wie beispielsweise die kognitive Psychologie sowie die Philosophie [Bu86]. Arnesen and Krogstie (2005) bewerten Modellierungssprachen auf der Basis

eines adaptierten Qualitätsframeworks. Dieses Framework legt der auf mehreren Ebenen gemessenen semiotischen Modellqualität einen mengentheoretischen Ansatz zu Grunde. Die anknüpfende Evaluation basiert auf praktischen Erfahrungen und theoretischen Bewertungen von Modellierungssprachen [ArKr05]. Aufbauend auf dieser Studie wenden Wahl und Sindre (2005) das entwickelte Framework an und transferieren es auf eine Evaluation der BPMN. Hierbei fokussieren sie semantische, syntaktische und pragmatische Aspekte der verschiedenen Elemente der BPMN [WaSi05]. Siau und Wang (2007) bewerten Modellierungssprachen wie beispielsweise das Use-Case-Diagramm oder das Rich-Picture-Diagramm. Sie greifen hierbei auf eine Liste mit kritischen Fragen zur Wissensrepräsentation zurück. Auch diese Studie basiert auf praktischen Erfahrungen und theoretischen Evaluationen. Sie schlussfolgern, dass empirische Studien ein weiterführendes Vorgehen zur Vervollständigung ihrer Arbeit sein würden. [SiWa07]. Bobkowska (2005) entwickelt ein methodologisches Rahmenkonzept zur Evaluation graphischer Modellierungssprachen [Bo05a]. Dumas et al. (2005) entwerfen ein pattern-basiertes Framework zur Bewertung des Kontrollflusses, sowie der Daten- und Ressourcenperspektive der BPMN [DHR05]. Eloranta et al. (2006) untersuchen die BPMN und UML. Diese Evaluation basiert auf verschiedenen Konzepten wie dem Workflow-Patterns-Framework und dem Bunge-Weber-Wand Modell [EKT06].

3) Empirische Evaluationsstudien fokussieren Beobachtungen, Befragungen und Experimente unter Verwendung logischer und statistischer Methoden [CoSc05]. Recker and Dreiling (2007) führen eine empirische Studie zur Ermittlung des Verständnisses von EPK und BPMN durch. Das Ergebnis dieser Studie ist, dass Prozessmodellierer mit einer hohen Kenntnis einer Modellierungssprache sehr einfach andere neue Modellierungssprachen erlernen können [ReDr07]. Mendling und Strembeck (2008) analysieren anhand eines Fragebogens Faktoren, die das Verständnis von Prozessmodellen beeinflussen. Das Ergebnis dieser Arbeit unterstützt die Hypothese, dass individuelle, modellbezogene und inhaltliche Kriterien das Verständnis von Prozessmodellen beeinflussen [MeSt08].

Die analysierten verwandten Arbeiten weisen auf, dass in diesen hauptsächlich Teilaspekte der Usability von Modellierungssprachen bewertet werden. Siau und Rossi (2007) kommen zu dem Ergebnis, dass aktuell eine Mangel an umfassenden empirischen Evaluationstudien mit nutzerbezogenen Szenarien herrscht. Sie schlagen in hohem Maße künftige empirische Studien zur Ermittlung der Usability von Modellierungssprachen vor [SiRo08]. Mendling und Strembeck (2008) empfehlen künftige Studien zur Untersuchung der Verständlichkeit von Prozessmodellen [MeSt08]. Diese Ergebnisse zeigen einen hohen aktuellen Forschungsbedarf einer Studie über die Usability von Modellierungssprachen.

5 Diskussion und Ausblick

Die Ideen dieses Beitrags bilden eine Grundlage für künftige empirische Untersuchungen zur Benutzerfreundlichkeit von Modellierungssprachen. Nach dem Erstellen einer allgemeingültigen Usabilitydefinition für Modellierungssprachen wurden die relevanten Usabilityattribute extrahiert, definiert und an Modellierungssprachen angepasst. Darauf aufbauend wurden Metriken für jedes Usabilityattribut unter Berücksichtigung der verschiedenen Nutzersituationen (Modellentwicklung, Modellbetrachtung) methodisch entwickelt. Es wurde gezeigt, dass aktuelle und frühere Evaluationsstudien größtenteils nur einige Teilbereiche der Usabilityattribute von Modellierungssprachen bewerten. Diese Studien beschränken sich auf einige wenige Modellierungssprachen und Usabilityattribute. Das in diesem Beitrag entwickelte Rahmenkonzept stellt eine generische Basis für künftige empirische Studien zur Analyse der Benutzerfreundlichkeit von grafischen Modellierungssprachen dar.

Der nächste Forschungsschritt umfasst die Durchführung einer Datenaufnahme zur empirisch gestützten Untersuchung der Benutzerfreundlichkeit von Modellierungssprachen. Basierend auf den Ergebnissen dieses Beitrags wird ein Hypothesenmodell entwickelt und mit empirischen Daten überprüft. Ein erster Pretest im Sommer 2009 bestätigte das methodisch erarbeitete Design der Datenaufnahme und ist Grundlage für eine anknüpfende Datenaufnahme im Winter 2009/Frühling 2010. Teile dieser Untersuchung werden Interviews, Fragebögen, Experimente und die Eye-Tracking Methode darstellen. Wir planen mit unserer Forschungsarbeit neue Erkenntnisse im Bereich der Usabilityevaluation von Modellierungssprachen gewinnen zu können und darauf aufbauend benutzernahe Empfehlungen für die weitere Entwicklung von Modellierungssprachen aussprechen zu können.

Literaturverzeichnis

[ArKr05] Arnesen, S.; Krogstie, J.: Assessing Enterprise Modeling Languages using a Generic Quality Framework. In: Krogstie John, H.T.A., Siau Keng (ed.): Information Modeling Methods and Methodologies. Hershey PA: Idea Group, 2005; 63-79.

[AvFi95] Avison, D.E.; Fitzgerald, G.: Information Systems Development: Methodologies, Techniques and Tools. McGraw-Hill Book Company, New York, 1995.

[BaHe00] Barbier, F.; Henderson-Sellers, B.: Object modelling languages: An evaluation and some key expectations for the future. Ann. Softw. Eng. 10, 2000; 67-101.

[BaLi02] Bausell, R.B.; Li, Y.-F.: Power analysis for experimental research: A practical guide for the biological, medical, and social sciences. Cambridge Univ. Press, Cambridge, 2002.

[Be95] Bevan, N.: Measuring usability as quality of use. Software Quality Journal 4, 1995; 115-150.

[BeMa94] Bevan, N.; Macleod, M.: Usability Measurment in Context. Behaviour and Information Technology 13, 1994; 132-145.

[Bo05a] Bobkowska, A.: A framework for methodologies of visual modeling language evaluation. ACM International Conference Proceeding Series 214, 2005.

[Bo05b] Bobkowska, A.: Modeling Pragmatics for Visual Modeling Language Evaluation. Proceedings of the 4th international workshop on Task models and diagrams, Gdansk, Journal 127, 2005.

[BrZa09] Bridgeland, D.M.; Zahavi, R.: Business Modeling: A Practical Guide to Realizing Business Value. Elsevier, Burlington, 2009.

[Bu86] Bubenko, J.A.: Information system methodologies; a research view. Conference on Information systems design methodologies: improving the practice, Noordwijkerhout, Netherlands, 1986; 289-318.

[BWW03]Bullinger, H.J.; Warnecke, H.J.; Westkämpfer, E.: Neue Organisationsformen in Unternehmen - ein Handbuch für das moderne Management. Springer, Berlin, 2003.

[CoLo99] Constantine, L.L.; Lockwood, L.A.: Software for Use: A practical Guide to the Models and Methods of Usage-Centered Design Addison-Wesley, New York, 1999.

[CoSc05] Cooper, D.R.; Schindler, P.S.: Business Research Methods. McGraw-Hill, New York, 2005.

[CoKi06] Coursaris, C.; Kim, D.: A Qualitative Review of Empirical Mobile Usability Studies. Proceedings of the Twelfth Americas Conference on Information Systems, 2006.

[Da03] Dangelmaier, W.: Produktion und Information. Springer, Wiesbaden, 2003.

[DMD08] Das, S.; McEwan, T.; Douglas, D.: Using eye-tracking to evaluate label alignment in online forms. Proceedings of the 5th Nordic conference on Human-computer interaction: building bridges Lund, Sweden, 2008; 451-454.

[Du07] Duchowski, A.T.: Eye Tracking Methodology - Theory and Practice. Springer, New York, 2007.

[DuRe99] Dumas, J.; Redish, J.: A practical guide to usability testing. Greenwood Publishing Group, Westport 1999.

[DHR05] Dumas, M.; Hofstede, A.; Russel, N.: Pattern-based Analysis of BPMN - an extensive evaluation of the Control-flow, the Data and the Resource Perspectives. http://is.tm.tue.nl/staff/wvdaalst/BPM center/reports/2005/BPM-05-26.pdf, 2005.

[EhWi07] Ehmke, E.; Wilson, S.: Identifying web usability problems from eye-tracking data. Proceedings of the 21st British CHI Group Annual Conference on HCI 2007: People and Computers British Computer Society, University of Lancaster, United Kingdom, 2007; 119-128.

[EKT06] Eloranta, L.; Kallio, E.; Thero, I.: A Notation Evaluation of BPMN and UML AD. http://www.soberit.hut.fi/T-86/T-86.5161/2006/BPMN_vs_UML_final.pdf, 2006.

[ElSc01] Elsuwe, H.; Schmedding, D.: Metriken für UML-Modelle. Informatik Forschung und Entwicklung 18, 2001; 22-31.

[IEEE90] Institute of Electrical and Electronics Engineers: Standard Glossary of Software Engineering Terminology. http://www.idi.ntnu.no/grupper/su/publ/ese/ieee-se-glossary-610.12-1990.pdf, 1990.

[FJM50] Fitts, P.M.; Jones, R.E.; Milton, J.L.: Eye movements of aircraft pilots during instrument-landing approaches. Aeronautical Engineering Review 9, 1950; 24-29.

[Ga07] Gartner Research: Magic Quadrant for Business Process Analysis Tools. 2007.

[Go04] Gordon, I.E.: Theories of visual perception. Psychology Press, Hove, 2004.

[GoKo99] Goldberg, J.; Kotval, X.: Computer interface evaluation using eye movements: methods and constructs International Journal of Industrial Ergonomics, 24(6), 1999; 631-645.

[GFA09] Grossman, T.; Fitzmaurice, G.; Attar, R.: A survey of software learnability: metrics, methodologies and guidelines. Proceedings of the 27th international conference on Human factors in computing systems Boston, MA, USA, 2009; 649-658.

[Ha05] Havey, M.: Essential business process modeling. O'Reilly, Beijing, 2005.

[ISO98] International Organization for Standardization (ISO): Ergonomic Requirements for Office Work with visual Display Terminals (VDTs); Part 11: Guidance on Usability. ISO 9421-11, 1998.

[ISO06] International Organization for Standardization (ISO): Ergonomics of Human-System-Interaction; Part 110: Dialogue Principles. ISO 9241-110, 2006.

[JaKa03] Jacob, R.K.; Karn, K.S.: Eye Tracking in Human-Computer Interaction and Usability Research: Ready to Deliver the Promises. The Mind's Eye 2003; 573-605.

[Ka02] Kan, S.H.: Metrics and Models in Software Quality Engineering. Addison-Wesley, Boston, 2002.

[KaKü] Karagiannis, D.; Kühn, H.: Metamodeling Platforms. Invited Paper University of Vienna, 2002.

[KEC99] Karn, K., S.; Ellis, S.; Cornell, J.: The hunt for usability: tracking eye movements. CHI '99 extended abstracts on Human factors in computing systems Pittsburgh, Pennsylvania, 1999; 173-173.

[KST07] Krallmann, H.; Schönherr, M.; Trier, M.: Systemanalyse in Unternehmen. Oldenbourg, München, 2007.

[KHS05] Krogstie, J.; Halpin, T.A.; Siau, K.: Information modeling methods and methodologies. Idea Group Publ, Hershey PA, 2005.

[Lo90] Loy, P.: A comparison of object-oriented and structured development methods. SIGSOFT Softw. Eng. Notes 15, 1990; 44-48.

[Lu03] Ludewig, J.: Models in software engineering - an introduction. Software and Systems Modeling 2, 2003; 5-14.

[MaEr05] Matthes, F.; Ernst, A.M.: Enterprise Architecture Management Tool Survey 2005. University Munich, 2005.

[Ma89] Mayer, R.E.: Models for Understanding. Review of Educational Research 59, 1989; 43-64.

[MMR09] Melcher, J.; Mendling, J.; Reijers, H.A.; Seese, D.: On Measuring the Understandability of Process Models. http://digbib.ubka.uni-karlsruhe.de/volltexte/1000011993, 2009.

[MeSt08] Mendling, J.; Strembeck, M.: Influence Factors of Understanding Business Process

Models. Proceedings of the 11th International Conference on Business Information Systems 7, 2008; 142-153.

[Na01] Nahman: Using Eye-Tracking for Usability testing. http://www.namahn.com/resources/documents/note-eyetracking.pdf, 2001.

[NeNa02] Nembhard, D.; Napassavong, O.: Task complexity effects on between-individual learning/forgetting variability. International Journal of Industrial Ergonomics 29, 2002; 297-306.

[NeUz00] Nembhard, D.; Uzumeri, M.: Experimental learning and forgetting for manual and cognitive tasks. International Journal of Industrial Ergonomics 25, 2000; 315-326.

[Ni06] Nielsen, J.: Usability engineering. Kaufmann, Amsterdam, 2006.

[OSV86] Olle, T.W.; Sol, H.G., Verijin-Stuart, A.A.: A comparative evaluation of system development methods. Proceedings of the IFIP WG 8.1 working conference on Information systems design methodologies: improving the practice, Noordwijkerhout, Netherlands, 1986; 19-54.

[PoBa05] Poole, A.; Ball, L. J: Eye Tracking in Human-Computer Interaction and Usability Research: Current Status and Future Prospects. In C. Chaoui (Ed.), Encyclopedia of HCI. Idea Group, Pennsylvania, 2005.

[Pr94] Preece, J.; Rogers, Y.; Sharp, H.; Benyon, D.; Holland, S.; Carey, T.: Human Computer Interaction. Addison-Wesley, Wokingham, 1994.

[PCV05] Pretorius, M.C.; Calitz, A.P.; van Greunen, D.: The added value of eye tracking in the usability evaluation of a network management tool. Proceedings of the 2005 annual research conference of the South African institute of computer scientists and information technologists on IT research in developing countries White River, South Africa, 2005; 1-10.

[ReDr07] Recker, J.C.; Dreiling, A.: Does it matter which process modelling language we teach or use? An experimental study on understanding process modelling languages without formal education Australasian Conference on Information Systems, Toowoomba, 2007.

[Re09] Recker, J.C.; Zur Muehlen, M.; Keng, S., Erickson; J., Indulska, M.: Measuring Method Complexity: UML versus BPMN. Proceedings of the Fifteenth Americas Conference on Information Systems, San Francisco, California 2009.

[Re93] Rengger, R.; Macleod, M.; Bowden, R.; Blaney, M.; Bevan, N.: MUSiC Performance Measurement Handbook. National Physical Laboratory, Teddington, UK, 1993.

[RoBr96] Rossi, M.; Brinkkemper, S.: Complexity Metrics for Systems Development Methods and Techniques. Information Systems 21, 1996; 209-227.

[Sc04] Scholtz, J., Usability Evaluation. http://www.itl.nist.gov/iad/IADpapers/ 2004/Usability%20Evaluation_rev1.pdf, 2004.

[Se06] Seffah, A.; Donyaee, M.; Kline, R.; Padda, H.: Usability measurement and metrics: A consolidated model. Software Quality Control 14, 2006; 159-178.

[Sh91] Shackel, B.: Usability - Context, framework, definition, design and evaluation. In: Shackel, B.; Richardson, S. (eds.): Human Factors for Informatics Usability. University Press, Cambridge, 1991; 21-38.

[SiRo08] Siau, K.; Rossi, M.: Evaluation techniques for systems analysis and design modelling methods ; a review and comparative analysis. Information Systems Journal 2008;

[SiWa07] Siau, K.; Wang, Y.: Cognitive evaluation of information modeling methods. Information and Software Technology 49, 2007; 455-474.

[SiJa00] Sibert, L., E.; Jacob, R.: Evaluation of eye gaze interaction. Proceedings of the SIGCHI conference on Human factors in computing systems, The Hague, The Netherlands, 2000; 281-288.

[SDJ07] Sjoberg, D.; Dyba, T.; Jorgensen, M.: The Future of Empirical Methods in Software Engineering Research. Future of Software Engineering 2007; 358-378.

[St73] Stachowiak, H.: Allgemeine Modelltheorie. Springer, Wien, 1973.

[St86] Strom, R.: A comparison of the object-oriented and process paradigms. Proceedings of the 1986 SIGPLAN workshop on Object-oriented programming, Yorktown Heights, New York, United States, 1986; 88-97.

[SuMe09] Suul, L.; Mellouli, T.: Optimierungssysteme - Modelle, Verfahren, Software, Anwendungen. Springer, Berlin, 2009.

[TKM08] Tamir, D., Komogortsev, O.V., Mueller, C.J.: An effort and time based measure of usability. Proceedings of the 6th international workshop on Software quality, Leipzig, Germany, 2008.

[VaLi07] Van Schaik, P.; Ling, J.: Design parameters of rating scales for web sites. ACM Trans. Comput.-Hum. Interact. 14, 2007; 1-35.

[Vu08] Vuolle, M.; Aula, A.; Kulju, M.; Vainio, T.: Identifying Usability and Productivity Dimensions for Measuring the Success of Mobile Business Services. Advances in Human-Computer Interaction, 2008.

[WaSi05] Wahl, T.; Sindre, G.: An analytical evaluation of BPMN using a semiotic quality framework. International Workshop on Exploring Modeling Methods in Systems Analysis and Design, Porto, 2005.

[WaWe93] Wand, Y.; Weber, R.: On the ontological expressiveness of information systems analysis and design grammars. Information Systems Journal 3, 1993; 217-237.

[We92] Welke, R.: The case repository: more than another database application. In: Cottermann, W., Senn, J. (Hrsg.): Challenges and strategies for research in systems development Wiley Inc., 1992; 181-218.

Gebrauchstauglichkeit semiformaler Modellierungssprachen für das Anforderungsmanagement

Untersuchungsrahmen, Anwendungsfall und experimentelle Evaluation mittels Blickbewegungsregistrierung

Frank Hogrebe[1], Nick Gehrke[2] und Markus Nüttgens[3]

Abstract: Semiformale Modellierungssprachen zur Beschreibung technischer, organisatorischer oder betriebswirtschaftlicher Zusammenhänge haben eine zentrale Bedeutung für das Anforderungsmanagement (Requirements Engineering). Die Modifikation bestehender und die Entwicklung neuer semiformaler Modellierungssprachen nehmen trotz der vorhandenen Vielfalt weiter zu. Der Schwerpunkt von Forschungsarbeiten zur Gebrauchstauglichkeit (Usability) semiformaler Modellierungssprachen liegt traditionell im Bereich der softwaretechnischen und weniger der benutzerbezogenen Anforderungen. Der vorliegende Beitrag wertet 13 Arbeiten zur Gebrauchstauglichkeit bei der Erstellung und Nutzung von Informationsmodellen aus und entwickelt auf dieser Basis einen interdisziplinären Untersuchungsrahmen, der Konzepte der (Wirtschafts-)Informatik und der Kommunikationsforschung kombiniert und in einem Forschungsansatz operationalisiert. Dabei wird sowohl die Europäischen Usability-Norm EN ISO 9241, als auch die Methode der Blickbewegungsregistrierung (EyeTracking) einbezogen. Die experimentelle Evaluation des Untersuchungsrahmens erfolgt anhand eines pilotierten Anwendungsfalls zu Varianten der Ereignisgesteuerten Prozesskette (EPK).

1 Ausgangslage und Motivation

Eine zentrale Aufgabe der Wirtschaftsinformatik besteht in der Analyse und Gestaltung von Informationssystemen in Wirtschaft und Verwaltung. Aus dieser Aufgabenstellung leitet sich die Zielsetzung der Wirtschaftsinformatik ab, Konzepte, Methoden und Werkzeuge zu entwickeln, welche die Gestaltung von Informationssystemen in Wirtschaft und Verwaltung unterstützen [WK94, S.80f.]. Nach einer Marktstudie stiegen die Umsätze im Bereich des Marktes der Modellierungswerkzeuge zur Geschäftsprozessmodellierung in den Jahren 2004 – 2007 um durchschnittlich 15% [Ga07, S. 2]. Je nach Modellierungsadressat sind unterschiedliche Anforderungen zu stellen, die durch Modellierungswerkzeuge technisch realisiert werden. Die vorliegende Untersuchung legt den Fokus auf benutzerbezogene Anforderungen beim Einsatz semiformaler Modellierungssprachen und der Nutzung der resultierenden Modelle. Aus wissenschaftlicher Sicht stellen sich grundsätzliche Fragen hinsichtlich Gemeinsamkeiten und Unterschiede von Arbei-

[1] Universität Hamburg, Lehrstuhl für Wirtschaftsinformatik, Von-Melle-Park 5, D-20146 Hamburg, frank.hogrebe@wiso.uni-hamburg.de
[2] Universität Hamburg, Lehrstuhl für Wirtschaftsinformatik, Von-Melle-Park 5, D-20146 Hamburg, nick.gehrke@wiso.uni-hamburg.de
[3] Universität Hamburg, Lehrstuhl für Wirtschaftsinformatik, Von-Melle-Park 5, D-20146 Hamburg, markus.nuettgens@wiso.uni-hamburg.de

ten, die sich mit benutzerbezogenen Anforderungen an semiformale Modellierungssprachen befassen. Abbildung 1 gibt einen Überblick über den Untersuchungsgang, der von drei Forschungsfragen (F) geleitet wird:

F1: Welche Anforderungen werden allgemein an semiformale Modellierungssprachen gestellt und werden Anforderungen in einschlägigen Arbeiten auch gleich benannt und definiert bzw. entsprechen sie sich in ihrer Semantik?

F2: Haben sich bestimmte benutzerbezogene Anforderungen an die Gebrauchstauglichkeit semiformaler Modellierungssprachen herausgebildet?

F3: Ist Blickbewegungsregistrierung eine geeignete Methode zur Messung und Bewertung von Anforderungen an die Gebrauchstauglichkeit semiformaler Modellierungssprachen?

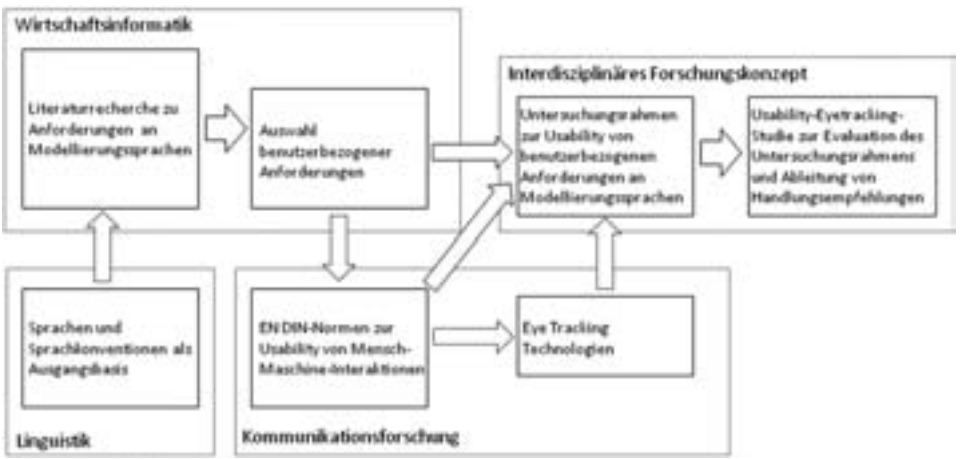


Abb. 1: Untersuchungsgang zur Forschungsarbeit

Systematische wissenschaftliche Untersuchungen von Sprachen gehören traditionell in den Forschungsbereich der Linguistik, die sich mit natürlichen Sprachen befasst [Pa06, S. 1]. Die vorliegende Arbeit legt das linguistische Sprachverständnis zu Grunde und dehnt es auf künstliche Sprachen der Informatik aus. Ausgangspunkt für die Untersuchung bildet eine Literaturrecherche zu den Anforderungen bei der Erstellung und Nutzung von Informationsmodellen. Ziel der Erhebung ist es, Hinweise darauf zu erhalten, welche Eigenschaften eine Modellierungssprache besitzen soll (Anforderungen) [Pa06, S. 57]. In wieweit eine Modellierungssprache und ihre Modelle den Anforderungen ihrer Benutzer entspricht, hängt von ihrer Gebrauchstauglichkeit (Usability) aus Sicht der jeweiligen Benutzer ab. Systematische Untersuchungen zur Gebrauchstauglichkeit sind insbesondere ein Untersuchungsfeld der Kommunikationsforschung. Arbeiten hierzu finden sich zur Web-Usability ([RP07], [Yo03]), Usability mobiler Systeme [KS04] und TV-Usability ([Ob07], [PG03]); einschlägige Normen zur Gebrauchstauglichkeit finden sich in den Teilen 11, 12 und 110 der Europäischen Usability-Norm EN ISO 9241 [IS99], [IS00], [IS08]. Aus den gewonnenen Erkenntnissen wird ein Untersuchungsrahmen zur Durchführung von Usability EyeTracking Studien zur Evaluation von benutzerbezogenen Anforderungen an semiformale Modellierungssprachen und ihrer Modelle

entwickelt. Dieser bezieht sowohl Erkenntnisse der (Wirtschafts-) Informatik als auch der Kommunikationsforschung ein. Als Instrumentarium für die Anwendung des Untersuchungsrahmens wird, neben Befragungstechniken, auch die Blickbewegungsregistrierung (EyeTracking) einbezogen, um neben qualitativen Indikatoren (wie subjektive Wahrnehmungen) auch quantitative Indikatoren (wie objektive Messungen zum Einfluss von Blickbewegungen auf die Zeitdauer der Modellierung von Modellen oder Fixationsanzahl und -dauer bei der Modellnutzung) gleichermaßen auswerten zu können. Der Untersuchungsgang fokussiert auf problem- und domänenunabhängige allgemeine benutzerbezogene Anforderungen, so dass sich die nachfolgenden Ausführungen ausschließlich auf semiformale Modellierungssprachen als invarianter Kern von Modellierungsmethoden konzentrieren.

Der Beitrag ist wie folgt aufgebaut: Im zweiten Abschnitt werden Arbeiten zu Anforderungen an die Erstellung und Nutzung von Informationsmodellen ausgewertet. Auf dieser Basis werden benutzerbezogene Anforderungen kriterienbasiert herausgearbeitet. Der dritte Abschnitt befasst sich mit den in der Kommunikationsforschung angewandten Teilen der Usability-Norm EN ISO 9241. In diesem Kontext wird auch die in der Kommunikationsforschung eingesetzte Methode der Blickbewegungsregistrierung einbezogen. Aus den gewonnenen Erkenntnissen wird im Abschnitt 4 ein Untersuchungsrahmen zur Messung und Bewertung der Anforderungen an die Gebrauchstauglichkeit semiformaler Modellierungssprachen entwickelt und zur vergleichenden Bewertung zweier Varianten der Ereignisgesteuerten Prozesskette (EPK) exemplarisch angewandt. Die Arbeit schließt mit einer Zusammenfassung, Hinweise zur Limitation der Ergebnisse und einem Ausblick auf den weiteren Forschungsbedarf.

2 Anforderungen an semiformale Modellierungssprachen

2.1 Verwandte Arbeiten und Untersuchungseinheiten

Der vorliegende Abschnitt grenzt den Untersuchungsgegenstand zu verwandten Arbeiten ab, die sich auch mit Anforderungen an semiformale Modellierungssprachen und ihren Informationsmodellen befassen. Für die Untersuchung werden einschlägige Arbeiten nach folgenden Kriterien ausgewählt:

- Kriterium 1 (qualitativ): Die Arbeiten müssen explizit Anforderungen an die Modellierung und Nutzung von Informationsmodellen zum Gegenstand haben.
- Kriterium 2 (quantitativ): Es müssen mindestens fünf verschiedene Anforderungen in den Arbeiten als Unterscheidungskriterien vorgeschlagen werden, um eine ausreichende Differenzierung zu ermöglichen.
- Kriterium 3 (quantitativ): Unter den fünf verschiedenen Anforderungen muss mindestens eine benutzerbezogene Anforderung Gegenstand der Arbeit sein, da diese im Fokus der angestrebten Untersuchung stehen.

Auf Grundlage dieser Kriterien werden die nachfolgenden Arbeiten als Untersuchungseinheiten für die Forschungsarbeit ausgewählt:

	Einbezogene Arbeiten	Jahr	Titel	Anforderungen
1.	Bailey / Pearson [BP83]	1983	Developmentof a Tool for Measuring and Analysing Computer User Satisfaction	39
2.	Baroudi / Orlikowski [BO88]	1988	A Short-Form Measure of User Information Satisfaction: A Psychometric Evaluation and Notes on Use	13
3.	Batini et al. [BCN92]	1992	Conceptual Database Design: An Entity-Relationship Approach	8
4.	Becker et al. [BSG99]	1999	Grundsätze ordnungsgemäßer Modellierung (GoM)	6
5.	Bertram [Be92]	1992	Aspekte der Qualitätsicherung von Unternehmensdatenmodellen	12
6.	Daneva et al. [DHS96]	1996	Benchmarking Business Process Models	7
7.	Doll / Torkzadeh [DT88]	1988	The Measurement of End-User Computing Satisfaction	12
8.	EN ISO 9241-12 [IS00]	2000	Ergonomische Anforderungen für Bürotätigkeiten mit Bildschirmgeräten; Teil 12: Informationsdarstellung	7
9.	Frank / van Laak [FL03]	2003	Anforderungen an Sprachen zur Modellierung von Geschäftsprozessen	12
10.	Ives et al. [IOB83]	1983	The Measurement of User Information Satisfaction	39
11.	Moody [Mo98]	1998	Metrics for Evaluating the Quality of Entity Relationship Models	8
12.	Patig [Pa06]	2006	Die Evolution von Modellierungssprachen	10
13.	Wixom / Todd [WT05]	2005	A Theoretical Integration of User Satisfaction and Technology Acceptance	9

Tab. 1: Untersuchungseinheiten der Forschungsarbeit

Die Arbeiten zeigen hinsichtlich der Anzahl der Anforderungen größere Unterschiede (von 6 bis 39 Unterscheidungskriterien). Eine Untersuchung auf der Aggregationsebene der Arten von Anforderungen an semiformale Modellierungssprachen (als übergeordnete Ebene) lässt dem gegenüber keine wissenschaftlich sinnvoll verwertbaren Erkenntnisse erwarten. Zu unterschiedlich sind die in den Arbeiten gewählten Aggregationsbündelungen. So bündeln Wixom und Todd [WT05, S. 90] in zwei Anforderungsarten, Frank und van Laak [FL03, S. 25-33] in drei und Doll und Torkzadeh [DT88, S. 268] in fünf; Bailey und Pearson [BP83], Becker et al. [Be99], Ives et al. [IOB83], [IS00] sowie Baroudi und Orlikowski [BO88] bilden keine Aggregationsbündel.

2.2 Benutzerbezogene Anforderungen

Welche Anforderungen wesentlich für die Benutzer sind (benutzerbezogene Anforderungen), hat sich unter Bezugnahme auf Tabelle 2, die alle Anforderungen der 13 Untersuchungseinheiten aufführt, nicht eindeutig herausgebildet. Der Tabelle liegt folgender Aufbau zugrunde:

- Die in den 13 Arbeiten (Untersuchungseinheiten) insgesamt herangezogenen 86 Anforderungen werden zunächst alphabethisch aufsteigend sortiert.
- Kriterien, die in 3 (und mehr) der Arbeiten als Anforderungen an die Modellierung und Nutzung von Modellen herangezogen werden, wurden im Weiteren an den Anfang gestellt (Kriterien 1 - 24).
- Innerhalb dieser 24 (häufigsten) Kriterien wird schließlich zwischen benutzerbezogenen (1 - 10) und systembezogenen Kriterien (11 - 24) differenziert.

Die Zuordnung von Anforderungen zu den 10 benutzerbezogenen Kriterien gründet sich dabei auf eine detaillierte Auswertung der den jeweiligen Anforderungen zugrunde liegenden Definitionen und semantischen Beschreibungen in den 13 Arbeiten.

Kriterium	Bailey / Pearson [BP83]	Baroudi / Orlikowski [BO88]	Batini et al. [BCN92]	Becker et al. [BSG99]	Betram [Be92]	Daneva et al. [DHS96]	Doll / Torkzadeh [DT88]	Frank / van Laak [FL03]	ISO 9241-12 [IS00]	Ives et al. [IOB83]	Moody [Mo98]	Patig [Pa06]	Wixom / Todd [WT05]	mindestens 3 Nennungen	davon benutzerbezogen
1 Einfachheit								x			x	x		x	x
2 Flexibilität	x				x					x	x		x	x	x
3 Genauigkeit	x	x					x			x			x	x	x
4 Klarheit				x			x		x					x	x
5 Präzision	x	x					x			x				x	x
6 Relevanz	x	x		x			x			x				x	x
7 Verständlichkeit	x	x			x	x		x	x	x	x	x		x	x
8 Vollständigkeit	x	x	x		x		x	x		x	x		x	x	x
9 Zeiterfordernis	x	x								x				x	x
10 Zweckmäßigkeit	x						x			x				x	x
11 Ablaufsteuerung	x	x								x				x	
12 Beziehung zum EDP Personal	x	x								x				x	
13 Dokumentation	x					x				x				x	
14 Eingebundenheit	x	x								x				x	
15 Einstellung des EDP Personals	x	x								x				x	
16 Format des Outputs	x						x			x			x	x	
17 Gültigkeit	x						x			x			x	x	
18 Integration	x									x	x		x	x	
19 Kommunikation mit dem EDP Personal	x	x								x				x	
20 Korrektheit			x					x			x			x	
21 Rechtzeitigkeit	x						x			x			x	x	
22 Trainingsgrad	x	x								x				x	
23 Zugänglichkeit	x									x			x	x	
24 Zuverlässigkeit	x	x					x			x			x	x	
25 Abstraktionsniveau					x										
26 Angemessenheit								x							
27 Anschaulichkeit								x							
28 Antwortzeit	x									x					
29 Arbeitsplatzauswirkungen	x									x					
30 Ausdruckskraft			x												
31 Ausdrucksstärke												x			
32 Ausführbarkeit												x			
33 Automationsunterstützung	x									x					
34 Benutzerfreundlichkeit							x								
35 Bestimmtheit der Prioritäten	x									x					
36 Beteiligung der Leitungsspitze	x									x					
37 Darstellung					x										
38 Datensicherheit	x									x					
39 Einheitlichkeit								x							
40 Erkennbarkeit									x						
41 Erwartungskonfirmität	x									x					
42 Erweiterbarkeit			x												
43 Fehlerbehebung	x									x					
44 Formale Kriterien					x										
45 Formalität												x			

Legende: EDP = Elektronische Datenverarbeitung (electronic data processing)

Tab. 2: Empirische Auswertung zu Anforderungen an Informationsmodellen (Ausschnitt)

Für eine empirische Anwendung ist es wesentlich, dass die hergeleiteten benutzerbezogenen Anforderungen in ausreichendem Maße überschneidungsfrei und damit abgrenzbar definiert sind. Wäre dies nicht der Fall, könnten Untersuchungen sowohl hinsichtlich ihrer Durchführung als auch ihrer Aus- und Bewertung erschwert werden. Die 10 benutzerbezogenen Anforderungen wurden daher auf Basis ihrer Definitionen und semantischen Beschreibungen weiter untersucht. Tabelle 3 listet sieben benutzerbezogenen Anforderungen auf, die im Ergebnis der Abgrenzungsprüfung als überschneidungsfrei klassifiziert werden. Dabei werden die übrigen drei nicht abgrenzbaren Anforderungen jeweils der Anforderung zugeordnet, von der sie nicht abgrenzbar waren.

Benutzerbezogene Anforderungen an semiformale Modellierungssprachen und Informationsmodelle	Nicht abgrenzbare benutzerbezogene Anforderungen
Einfachheit	
Flexibilität	
Genauigkeit	Präzision
Verständlichkeit	Klarheit
Vollständigkeit	Relevanz
Zeiterfordernis	
Zweckmäßigkeit	

Tab. 3: Benutzerbezogene Anforderungen an Semiformaler Modellierungssprachen und Informationsmodelle

Bei den Anforderungen „Einfachheit“, „Flexibilität“, „Zeiterfordernis“ und „Zweckmäßigkeit“ ergeben sich keine Schwierigkeiten bei der Frage der Abgrenzung und Überschneidungsfreiheit. Abgrenzungsschwierigkeiten treten hingegen bei den Anforderungen „Genauigkeit und Präzision“, „Verständlichkeit und Klarheit“ sowie „Vollständigkeit und Relevanz“ auf. Nachfolgend soll die Abgrenzungsprüfung, aufgrund der nicht a priori erkennbaren semantischen Übereinstimmung der Begriffe, exemplarisch an den Anforderungen „Vollständigkeit und Relevanz“ dargestellt werden. Die vollständige Analyse findet sich bei [HN09]:

Vollständigkeit wird in 9 der 13 Untersuchungseinheiten verwandt. Nach Bailey und Pearson [BP83, S. 541] bezeichnet Vollständigkeit den „auf den Inhalt bezogene Umfang der Ausgabeinformationen“. Ähnlich auch Wixom und Todd [WT05, S. 91], wonach Vollständigkeit „den Grad bezeichnet, in dem ein System alle notwendigen Informationen bereit stellt“. Damit legen beide einen systembezogenen Fokus. Nach Becker et al. [Be99, S. 140] wird ein Modell als vollständig bezeichnet, wenn es „alle relevanten Eigenschaften der Problemdomäne enthält“ und Bertram [Be92, S. 59] bezeichnet „ein Modell als vollständig, wenn alle Anforderungen im Unternehmen nach Daten sowie nach Herleitbarkeit von Verdichtungen abgedeckt werden“. Beide beschreiben Vollständigkeit folglich modellbezogen. Sprachenbezogen definieren Frank und van Laak [FL03, S. 26] Vollständigkeit unter Verwendung von Eindeutigkeit, indem „Vollständigkeit meint, dass alle in der Modellierungssprache verwendeten Konzepte sowie die Bedingungen ihrer Verwendung eindeutig definiert sein sollten“ Brodie [Br84, S.41] und McGee [Mc76, S. 379] teilen diese Ausfassung nicht, danach soll Eindeutigkeit alternative Äußerungen mit derselben deskriptiven Bedeutung in semiformalen Modellierungssprachen vermeiden. Eine explizit auf den Nutzer bezogene Definition wählt Moody [Mo98,

S. 214], der ein Modell dann als vollständig bezeichnet, wenn es „alle relevanten Eigenschaften enthält, die der Modellnutzer fordert“. In den Arbeiten von Baroudi und Orlikowski [BO88, S. 49], Doll und Torkzadeh [DT88, S. 268] und Ives et al. [IOB83, S. 792] wird die Anforderung Vollständigkeit als Differenzierungskriterium zwar verwandt, jedoch nicht definiert.

Die Anforderung Relevanz wird in fünf der Untersuchungseinheiten verwandt. Bailey und Pearson [BP83, S. 542] bezieht Relevanz auf „den Grad der Übereinstimmung zwischen dem, was ein Benutzer möchte oder benötigt und dem, was durch die Informationsprodukte und -services bereitgestellt wird“. Dieser Definititionsansatz findet sich auch bei der Anforderung „Vollständigkeit“, vgl. Becker et al. [Be99, S. 140], Bertram [Be92, S. 59], Moody [Mo98, S. 214], Wixom und Todd [WT05, S. 91]. Nach Becker et al. [Be99, S. 14] sind „die in einem Modell enthaltenen Elemente und Beziehungen genau dann relevant, wenn der Nutzeffekt der Modellverwendung sinken würde, falls das Modell weniger Informationen enthalten würde“. Auch hier ist der klare Bezug zur Vollständigkeit erkennbar. Keine Definitionen zu Relevanz finden sich hingegen bei Baroudi und Orlikowski [BO88, S. 49], Doll und Torkzadeh [DT88, S. 268] und Ives et al. [IOB83, S. 792], die gleichwohl „Relevanz“ als Differenzierungskriterium einsetzen.

Mit Blick auf die Ausführungen können die Anforderungen „Vollständigkeit“ und „Relevanz“ nicht ausreichend voneinander abgegrenzt werden. Vielmehr tendieren die Definitionen der „Relevanz“ inhaltlich zur Anforderung „Vollständigkeit“. In den Untersuchungsrahmen wird daher auf die Anforderung „Relevanz“ zugunsten der „Vollständigkeit“ verzichtet. Zur Abgrenzungsprüfung der übrigen Anforderungen erfolgte analog, vgl. [HN09].

3 Gebrauchstauglichkeit und Blickbewegungsregistierung in der Kommunikationsforschung

3.1 DIN EN ISO 9241

Gebrauchstauglichkeit wird definiert, als das „Ausmaß, in dem ein Produkt durch bestimmte Benutzer in einem bestimmten Nutzenkontext genutzt werden kann, um bestimmte Ziele effektiv, effizient und zufriedenstellend zu erreichen“ [IS99, S. 4]. Dieser Definitionsansatz wird auch hier zugrunde gelegt. Weitere Definitionen zur Usability finden sich bei Sarodnick und Brau [SB06, S. 17] und Nielsen [Ni03, S. 3]. Einschlägige Normen zur Usability finden sich in den Teilen 11, 12 und 110 der Europäischen Usability-Norm DIN EN ISO 9241 [IS99], [IS00], [IS08]. Die Teile 12 und 110 unterstützen dabei das Konzept der Gebrauchstauglichkeit, das Inhalt von Teil 11 der DIN EN 9241 ist. ISO 9241-12 definiert charakteristische Eigenschaften dargestellter Information und gibt Empfehlungen zur Darstellung von Information als Teil des Dialoges (Informationsdesign). Die Eigenschaften des Teiles 12 sind in der empirischen Auswertung der berücksichtigt (Tabelle 2). Dargestellte Informationen sind aus einer technikbezogenen Betrachtung zwangsläufig in Dialogen enthalten, die auf den Grundsätzen vom Teil 110

von ISO 9241 basieren. Diese unterstützen wiederum vorrangig die Gestaltung des dynamischen Verhaltens eines interaktiven Systems (Interaktionsdesign), das nicht Gegenstand dieser Untersuchung ist.

3.2 Blickbewegungsregistierung (EyeTracking)

Als EyeTracking oder Blickbewegungsregistierung bezeichnet man eine Methode, mit der der Blickverlauf einer Person beim Betrachten eines Gegenstandes oder einer Anwendung gemessen werden kann [RDD08], [Du03]. In der Taxonomie der Usability-Methoden wird EyeTracking den benutzerorientierten Methoden zugeordnet [Yo03, S. 125]. Bei der Kombination von EyeTracking- und Usability Tests werden Befragungsergebnisse durch quantitative EyeTracking Daten ergänzt. Dabei kommentiert der Proband während der Untersuchung seine Aktionen, Gefühle und Gedanken bei der Benutzung des Untersuchungsgegenstandes (qualitativer Aspekt). Dies kann parallel zur Untersuchung erfolgen oder in einer späteren strukturierten Befragung (vgl. Untersuchungsrahmen). Der Blickverlauf des Probanden wird bei der EyeTracking Methode aufgezeichnet und gemessen (quantitativer Aspekt). Dieser methodische Ansatz soll in dieser Untersuchung auf die Messung und Bewertung der Usability von semiformalen Modellierungssprachen und deren Modelle übertragen werden.

4 Anwendungsfall zur Gebrauchstauglichkeit semiformaler Modellierungssprachen

4.1 Untersuchungsrahmen und Hypothesen

Der Untersuchungsrahmen hat einerseits das Ziel, die aus der Literaturanalyse erarbeiteten benutzerbezogenen Anforderungen empirisch zu überprüfen. Andererseits soll ein Rahmen geschaffen werden, der eine Messung und Bewertung der Gebrauchstauglichkeit semiformaler Modellierungssprachen durch eine Kombination von Usability-Testelementen (Befragungstechniken) und der EyeTracking Methode ermöglicht. Der Untersuchungsrahmen versteht sich in diesem Sinne als Rahmenkonzept zur Planung und Durchführung zweck- und adressatenspezifischer Usability EyeTracking Studien. Anhand einer experimentellen Überprüfung soll nachfolgend evaluiert werden, wie auf Basis benutzerbezogener Anforderungen semiformale Modellierungssprachen bezüglich ihrer Gebrauchstauglichkeit verglichen werden können. Zur Überprüfung dieser Zielsetzungen werden folgende Hypothesen (H1 - H3) zugrunde gelegt:

H1: Die aus der Literaturanalyse erarbeiteten 7 Anforderungen werden durch die Probanden (Ersteller/Nutzer) als wesentliche benutzerbezogene Anforderungen bestätigt.

H2: Die 7 Anforderungen sind geeignete Kriterien, um subjektive Wahrnehmungen der Probanden durch objektive Experimente zu überprüfen.

H3: EyeTracking ist eine geeignete Methode zur Messung und Bewertung von Anforderungen an die Gebrauchstauglichkeit semiformaler Modellierungssprachen und Informationsmodelle.

Eine empirische Verbundstudie für die grafische Darstellung von unternehmensbezogenen Verwaltungsprozessen bildet den Anwendungsfall für die Untersuchung. Hierzu werden zwei Varianten der EPK (erweiterte EPK (eEPK) und objektorientierte EPK (oEPK)) bezogen auf die sieben Anforderungen verglichen; grundlegende Arbeiten zu diesen Modellierungsnotationen finden sich bei Keller et al. [KNS92] und Scheer et al. [SNZ97]. Die Untersuchung basiert auf 24 Probanden (12 Ersteller und 12 Nutzer), die sich je zur Hälfte auf Verwaltungsmitarbeiter und Studenten aufteilen, so dass sich eine 6/6/6/6-Aufteilung ergibt. Die Probanden werden einerseits bezüglich der Erfüllung der Anforderungen befragt (subjektive Wahrnehmung). Andererseits werden Experimente mit den Probanden durchgeführt, wie z.B. die Messung der Zeit für das Modellieren eines vorgegebenen Prozesses (Ersteller), das Zählen der Anzahl entdeckter Fehler bei der Betrachtung eines existierenden Prozesses (Nutzer) oder die Anzahl der Augenfixationen beim Lösen einer bestimmten Aufgabe (Ersteller und Nutzer). Den Anforderungen werden jeweils Experimente zugeordnet, um die Erfüllung der Anforderung durch messbares Verhalten der Probanden zu objektivieren. Anschließend erfolgt ein Vergleich der Meinung (Befragung) bezüglich der Anforderung und der Messung der Anforderung (Experiment) für jede Modellierungssprache.

Für die Befragung lagen den Probanden bezüglich der Anforderungen sog. semantischen Charakteristika vor, die aus den Definitionen und semantischen Beschreibungen in den 13 Arbeiten (vgl. Kap. 2.2) generiert wurden. Die Befragung selbst fand nach Durchführung der Experimente statt, die für die Top 5 der Anforderungen durchgeführt wurden. Die größte Bedeutung aus Sicht der Probanden haben die Anforderungen „Verständlichkeit“ (Ersteller: 8; Nutzer: 11 Nennungen), „Vollständigkeit“ (Ersteller: 7; Nutzer: 6) und „Einfachheit“ (Ersteller: 5; Nutzer: 7). Die Frage, ob nach einer für Sie wichtigen Anforderung nicht gefragt wurde, beantwortet keiner der 24 Probanden mit „ja“ (11 Ersteller und 10 Nutzer antworteten mit „Nein“; der Rest machte „keine Angaben“), so dass die sieben aus der Literaturanalyse entwickelten benutzerbezogenen Anforderungen als wesentlich angesehen werden. Die Hypothese 1 wird damit bestätigt. Die Probanden konnten bei der Befragung bezüglich der Anforderung an die semiformale Modellierungssprachen jeweils antworten: trifft zu, trifft teilweise zu, trifft eher nicht zu, trifft nicht zu. Tabelle 4 stellt den Untersuchungsrahmen zur empirischen Erhebung zu den benutzerbezogenen Anforderungen dar. Die Spalte „Messung“ beschreibt das Experiment für die entsprechende Anforderung.

Anforderung	*Semantische Charakteristika*	*Wahrnehmung (Befragung von Ersteller und Nutzer)*	*Messung (Experiment mit Ersteller oder Nutzer)*
Einfachheit	• geringe Anzahl von Begriffen und Symbolen • einfache Regeln zur Anwendung	Probanden wurden befragt: Die Modellierungssprache X ist einfach?	Die Probanden modellieren einen text-lich beschriebenen Prozess. Es wird die Anzahl der Modellierungsfehler erhoben (Ersteller).

Genauigkeit	• Korrektheit der Informationen	Probanden wurden befragt: Die Modellierungssprache X ist genau?	Keine Messung, da die Bedeutung der Anforderung für die Benutzer als gering angesehen wurde (Platz 7 von 7).
Verständlichkeit	• bekanntes Vokabular • leicht zu verstehen • Bedeutung leicht interpretierbar	Probanden wurden befragt: Die Modellierungssprache X ist verständlich?	Es werden ein modellierter Prozess präsentiert und Verständnisfragen dazu gestellt. Es wird erhoben, wie viele Fehler ein Proband bei der Beantwortung macht (Nutzer).
Vollständigkeit	• alle relevanten Eigenschaften sind vorhanden • liefert alle erforderlichen Informationen	Probanden wurden befragt: Die Modellierungssprache X ist vollständig?	Es wird ein Prozess textlich vorgegeben, der von den Probanden modelliert wird. Es wird erhoben, ob ein Proband mit der Modellierungssprache den Prozess voll-ständig modellieren kann (Ersteller).
Zeiterfordernis	• Zeitaufwand zur Erstellung eines Modells	Probanden wurden befragt: Die Modellierungssprache X ist zeitsparend?	Die Probanden modellieren einen textlich beschriebenen Prozess. Es wird die dafür benötigte Zeit erhoben (Ersteller).
Zweckmäßigkeit	• leichte Anwendbarkeit / Nutzbarkeit zur Aufgabenerfüllung • Aufwand und Nutzen stehen in einem angemessenen Verhältnis	Probanden wurden befragt: Die Modellierungssprache X ist zweckmäßig?	In einem gegebenen modellierten Prozess müssen die Probanden ein vorgegebenes Objekt finden. Es werden die Anzahl und die Länge der Fixationen durch EyeTracking gemessen. Je mehr und je längere Fixationen, desto größer wird der Aufwand und desto schlechter das Aufwand-Nutzen-Verhältnis.

Tab. 4: Untersuchungsrahmen zur Erhebung benutzerbezogener Anforderungen

4.2 Statistische Auswertung der Befragung

Im Folgenden werden die durch Befragung erhobenen Einschätzungen bezüglich der Anforderungen statistisch ausgewertet. Für alle Anforderungen waren die Antworten trifft zu (=4), trifft teilweise zu (=3), trifft eher nicht zu (=2), trifft nicht zu (=1) jeweils für die Sprache eEPK und oEPK möglich. Durch fehlende Angaben (keine Antwort) kann die Stichprobengröße pro Anforderung auch geringer als 12 sein. Tabelle 5 zeigt grundlegende statische Kennzahlen zur Einschätzung der Anforderungen durch die Probanden bezüglich der beiden Modellierungssprachen. Es wird deutlich, dass alle Anforderungen bis auf die Anforderung „Genauigkeit“ bei der Modellierungssprache oEPK von den Probanden im Mittel besser bewertet wurden als bei der eEPK. Die Anforderung „Genauigkeit“ wird im Mittel gleich eingeschätzt. Die Korrelationen oEPK-eEPK der gegebenen Antworten (rechte Tabelle) zeigen nur bei der Anforderung „Zeiterfordernis“ einen signifikanten (negativen) Zusammenhang [> (-) 0,5]. Offenbar empfinden Probanden, die eine Modellierungssprache als zeitsparend erachten, die jeweils andere als weniger zeitsparend. Im Folgenden wird geprüft, ob die Unterschiede in den Meinungen für jede Anforderung signifikant voneinander abweichen oder nicht. Hierfür wurde ein t-Test auf unterschiedliche Mittelwerte bei abhängigen Stichproben durchgeführt. Tabelle 6 zeigt die Ergebnisse der Signifikanztests.

Paired Samples Statistics

		Mean	N	Std. Dev.	Std. Error Mean
Pair 1	EPK_Einfachheit	3,00	12	,000	,000
	oEPK_Einfachheit	3,33	12	,778	,225
Pair 2	EPK_Flexibilität	2,73	11	,786	,237
	oEPK_Flexibilität	3,36	11	,674	,203
Pair 3	EPK_Genauigkeit	3,27	11	,786	,237
	oEPK_Genauigkeit	3,27	11	,647	,195
Pair 4	EPK_Verstaendlichkeit	3,42	12	,515	,149
	oEPK_Verstaendlichkeit	3,67	12	,492	,142
Pair 5	EPK_Vollständigkeit	3,17	12	,835	,241
	oEPK_Vollständigkeit	3,33	12	,778	,225
Pair 6	EPK_Zeiterfordernis	2,73	11	1,009	,304
	oEPK_Zeiterfordernis	3,18	11	,751	,226
Pair 7	EPK_Zweckmäßigkeit	2,82	11	,982	,296
	oEPK_Zweckmäßigkeit	3,36	11	,674	,203

Paired Samples Correlations

		N	Corr	Sig.
Pair 1	EPK_Einfachheit & oEPK_Einfachheit	12		
Pair 2	EPK_Flexibilität & oEPK_Flexibilität	11	,384	,230
Pair 3	EPK_Genauigkeit & oEPK_Genauigkeit	11	,036	,917
Pair 4	EPK_Verstaendlichkeit & oEPK_Verstaendlichkeit	12	-,120	,711
Pair 5	EPK_Vollständigkeit & oEPK_Vollständigkeit	12	,400	,127
Pair 6	EPK_Zeiterfordernis & oEPK_Zeiterfordernis	11	-,588	,057
Pair 7	EPK_Zweckmäßigkeit & oEPK_Zweckmäßigkeit	11	,281	,438

Tab. 5: Statistische Kennzahlen der Meinungen zu den Anforderungen

Paired Samples Test

		Paired Differences					t	df	Sig. (2-tailed)	Sig. (1-tailed)	"Winner"
		Mean	Std. Deviation	Std. Error Mean	Confidence Lower	Confidence Upper					
Pair 1	EPK_Einfachheit - oEPK_Einfachheit	-,333	,778	,225	-,737	,070	-1,481	11	,166	,083	oEPK
Pair 2	EPK_Flexibilität - oEPK_Flexibilität	-,636	,809	,244	-1,078	-,194	-2,609	10	,026	,013	oEPK
Pair 3	EPK_Genauigkeit - oEPK_Genauigkeit	,000	1,000	,302	-,546	,546	,000	10	1,000	,500	N/A
Pair 4	EPK_Verstaendlichkeit - oEPK_Verstaendlichkeit	-,250	,754	,218	-,641	,141	-1,149	11	,275	,137	oEPK
Pair 5	EPK_Vollständigkeit - oEPK_Vollständigkeit	-,167	,835	,241	-,699	,266	-,692	11	,504	,252	oEPK
Pair 6	EPK_Zeiterfordernis - oEPK_Zeiterfordernis	-,455	1,572	,474	-1,314	,405	-,959	10	,360	,180	oEPK
Pair 7	EPK_Zweckmäßigkeit - oEPK_Zweckmäßigkeit	-,545	1,036	,312	-1,111	,021	-1,747	10	,111	,056	oEPK

Tab. 6: t-Tests der Meinungen zu den Anforderungen (90% Konfidenz)

Die Spalte „Sig. (2-tailed)“ zeigt das Signifikanzniveau zu dem die Meinungen für eEPK und oEPK voneinander abweichen (2-seitige ungerichtete statistische Hypothese H_0: eEPK=oEPK; H_1: oEPK≠eEPK). Interessant ist dazu die gerichtete Annahme, dass die oEPK signifikant besser bewertet wird, als die eEPK (1-seitige gerichtete statistische Hypothese H_0: EPK=oEPK; H_1: oEPK>EPK). Aufgrund der Gerichtetheit der statistischen Hypothese zeigt die Spalte „Sig. (1-tailed)“ das entsprechende Signifikanzniveau, welches stets die Hälfte der ungerichteten Signifikanz ist. Im Ergebnis zeigt sich, dass auf einem 90%-Konfidenzniveau die oEPK bei den Anforderungen „Einfachheit“, „Flexibilität“ und „Zweckmäßigkeit“ als signifikant besser empfunden wird. Auch bei den übrigen Anforderungen geht in keinem Fall die eEPK als „Winner“ hervor.

4.3 Statistische Auswertung der Experimente

Analog zu den Meinungen werden nachfolgend die Ergebnisse der Experimente - wie Sie in Tabelle 4 beschrieben wurden - dargestellt. Dabei ist zu beachten, dass sich Pair 4 und 5 beide auf die Eigenschaften „Zweckmäßigkeit“ beziehen, da hier die Werte einmal für die Anzahl und einmal für die Länge der Fixationen im Rahmen des EyeTrackings erhoben wurden.

Paired Samples Correlations

		N	Corr	Sig
Pair 1	EPK_Verstaendlichkeit_Fehler & oEPK_Verstaendlichkeit_Fehler	12	-,106	,742
Pair 2	EPK_Einfachheit_Fehler & oEPK_Einfachheit_Fehler	12	,217	,498
Pair 3	EPK_Zeitertordernis_Zeit & oEPK_Zeitertordernis_Zeit	12	,283	,373
Pair 4	EPK_Zweckm_AnzahlFix & oEPK_Zweckm_AnzahlFix	12	-,053	,870
Pair 5	EPK_Zweckm_LängeFix & oEPK_Zweckm_LängeFix	12	,208	,517

Paired Samples Statistics

		Mean	N	Std. Deviation	Std. Error Mean
Pair 1	EPK_Verstaendlichkeit_Fehler	2,5000	12	1,16775	,33710
	oEPK_Verstaendlichkeit_Fehler	2,9167	12	1,83196	,52884
Pair 2	EPK_Einfachheit_Fehler	10,0833	12	12,65959	3,65451
	oEPK_Einfachheit_Fehler	2,5833	12	3,08835	,89153
Pair 3	EPK_Zeitertordernis_Zeit	9,4500	12	3,03060	,87486
	oEPK_Zeitertordernis_Zeit	6,4083	12	2,19481	,63359
Pair 4	EPK_Zweckm_AnzahlFix	297,1667	12	63,35016	18,28762
	oEPK_Zweckm_AnzahlFix	253,2500	12	62,83185	18,13799
Pair 5	EPK_Zweckm_LängeFix	169,3406	12	37,21454	10,74291
	oEPK_Zweckm_LängeFix	142,4663	12	50,92841	14,70177

Tab. 7: Statistische Kennzahlen der Experimente zu den Anforderungen

Bei der Betrachtung der Tabelle 7 ist zu beachten, dass hier ein kleiner Wert jeweils besser ist als ein großer (weniger Fehler sind besser, weniger Zeitverbrauch ist besser, weniger und kürzere Fixationen sind besser etc.). Man sieht, dass auch bei den EyeTracking Experimenten die oEPK besser abschneidet als die eEPK. Lediglich bei der Anforderung „Verständlichkeit“ wurden bei der eEPK weniger falsche Antworten gegeben als bei der oEPK. Bei den Korrelationen oEPK-eEPK kann jeweils keine Signifikanz festgestellt werden. Die kürzeren Fixationen bei der oEPK zeigen sich auch im visuellen Vergleich der sog. EyeTracking Heatmaps [HPN09]. Abb. 2 zeigt qualitativ jeweils dazu die ersten und letzten 30 Sekunden einer 5-minütigen Betrachtungsphase für eEPK und oEPK bezogen auf die Aufmerksamkeitschwerpunkte. Nutzer hatten hier die Aufgabe, das blau markierte Ziel zu finden. Am Ende der Betrachtungsphase ist nur bei der eEPK-Darstellung eine rote Färbung zu erkennen, die eine lange Fixationsdauer dokumentiert (rotes Oval). Aus der Untersuchung kann geschlussfolgert werden, dass EyeTracking eine geeignete Methode zur Messung und Bewertung von Anforderungen an die Gebrauchstauglichkeit semiformaler Modellierungssprachen ist. Hypothese 3 wird damit bestätigt.

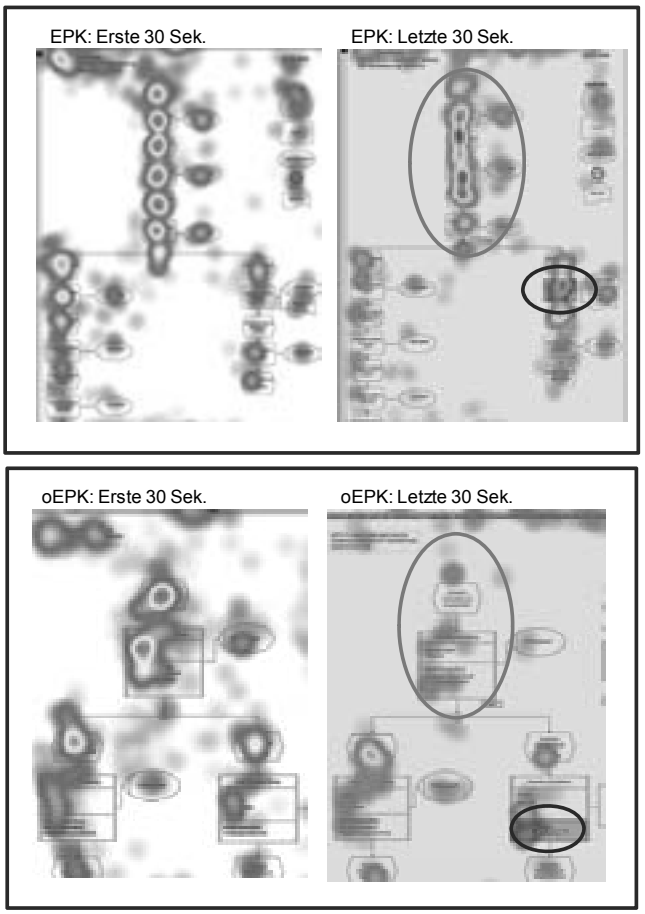


Abb. 2: eEPK- und oEPK-Fixationsvergleich

Paired Samples Test

		Paired Differences					t	df	Sig. (2-tailed)	Sig. (1-tailed)	"Winner"
		Mean	Std. Deviation	Std. Error Mean	Interval of the Lower	Upper					
Pair 1	EPK_Verstaendlichkeit_Fehler - oEPK_Verstaendlichkeit_Fehler	-,41667	2,27470	,65665	-1,59593	,76260	-,635	11	,539	,269	EPK
Pair 2	EPK_Einfachheit_Fehler - oEPK_Einfachheit_Fehler	7,50000	12,36197	3,56859	1,09122	13,90878	2,102	11	,059	,030	oEPK
Pair 3	EPK_Zeiterfordernis_Zeit - oEPK_Zeiterfordernis_Zeit	3,04167	3,19985	,92372	1,38278	4,70055	3,293	11	,007	,004	oEPK
Pair 4	EPK_Zweckm_AnzahlFix - oEPK_Zweckm_AnzahlFix	43,91667	91,55173	26,42871	-3,54625	91,37958	1,662	11	,125	,062	oEPK
Pair 5	EPK_Zweckm_LängeFix - oEPK_Zweckm_LängeFix	26,87433	56,48764	16,30658	-2,41040	56,15907	1,648	11	,128	,064	oEPK

Tab. 8: t-Tests der Messungen zu den Anforderungen (90% Konfidenz)

Der t-Test auf bessere (=kleinere) Mittelwerte zeigt, dass alle Ergebnisse bis auf die Anforderung „Verständlichkeit“ bei der oEPK mindestens auf einem 90% Niveau signifikant sind. Die mittlere Überlegenheit der eEPK bei „Verständlichkeit“ kann nicht als signifikant gedeutet werden. In der Tabelle 8 ist die Anforderung „Vollständigkeit“ nicht enthalten, da alle Probanden in der Lage waren, den vorgegebenen Prozess als eEPK und oEPK zu modellieren. Hier kann es insofern nur ein „Unentschieden“ geben.

4.4 Vergleich der Ergebnisse aus Befragung und Experiment

Nachdem die Ergebnisse der Meinungen und Messungen jeweils isoliert analysiert wurden, soll hier eine Gegenüberstellung als Gesamtergebnis dargestellt werden. Tabelle 9 zeigt die Ergebnisse der jeweiligen Analysen und stellt diese gegenüber. Klar erkennbar ist das signifikante Ergebnis pro oEPK für die Anforderungen „Einfachheit“ und „Zweckmäßigkeit“. Lediglich für eine Analyseseite signifikant, aber dennoch bei beiden im Mittel besser ist die oEPK bei der Anforderung „Zeiterfordernis“. Bei „Flexibilität“ wird die oEPK als besser empfunden, dies kann aber mangels Messung nicht experimentell bestätigt werden. Für die Anforderung „Genauigkeit“ kann keine Entscheidung für eine Modellierungssprache abgeleitet werden; dies ebenso aufgrund Insignifikanzen für die Anforderungen „Verständlichkeit“ und „Vollständigkeit“.

Zusammenfassend kann festgestellt werden, dass die oEPK im Experiment besser abschneidet (zum Teil signifikant). Die eEPK ist entweder unterlegen oder gleich auf (vgl. Tabelle 9). Die literaturbasierten allgemeinen benutzerbezogenen Anforderungen sind insgesamt geeignet subjektive Wahrnehmungen der Probanden durch objektive Experimente zu überprüfen. Die Hypothese 2 wird damit bestätigt.

Anforderung	"Winner"	"Winner“	Ergebnis eindeutig
Einfachheit	oEPK	oEPK	Ja, signifikant pro oEPK
Flexibilität	oEPK	keine Messung	Bei Meinung oEPK, keine Messung
Genauigkeit	N/A	keine Messung	Nein, bei Meinung unentschieden, keine Messung
Verständlichkeit	oEPK	eEPK	Nein, doppelt insignifikant, gegensätzlich
Vollständigkeit	oEPK	N/A	Meinung Nein (insignifikant), Messung unentschieden
Zeiterfordernis	oEPK	oEPK	Ja, aber eine Seite (Meinung) insignifikant pro oEPK
Zweckmäßigkeit	oEPK	oEPK	Ja, signifikant pro oEPK

Tab. 9: Gegenüberstellung Meinung und Messung (90% Konfidenz)

5 Zusammenfassung und weiterer Forschungsbedarf

5.1 Zusammenfassung

Die Forschungsarbeit wertet 13 Untersuchungen zur Gebrauchstauglichkeit semiformaler Modellierungssprachen aus. Aus 86 Anforderungen in den Arbeiten werden sieben allgemeine benutzerbezogene Anforderungen extrahiert. Bezogen auf die benutzerbezogenen Anforderungen wird der Definitionsansatz der Usability-Norm EN ISO 9241 als

Grundlage zur Evaluation der Gebrauchstauglichkeit semiformaler Modellierungssprachen zugrunde gelegt und die in der Kommunikationsforschung eingesetzte EyeTracking Methode bei der Entwicklung eines Untersuchungsrahmen für die Durchführung von Usability EyeTracking Studien einbezogen. Der Beitrag beschreibt einen prototypischen Anwendungsfall für den Untersuchungsrahmen. Untersuchungseinheiten bilden die eEPK und oEPK. Die Ergebnisse zu den Forschungsfragen können wie folgt zusammengefasst werden:

Ad. F1: Die Zahl der Anforderungen an die Modellierung und Nutzung von Informationsmodellen ist enorm. Allein 86 begrifflich unterschiedliche Anforderungen werden in den 13 Untersuchungseinheiten herangezogen. Dass begrifflich gleiche Anforderungen auch gleich definiert werden bzw. sich in ihrer Semantik entsprechen, konnte nur selten festgestellt werden.

Ad. F2: Auf Grundlage von drei Auswahlkriterien konnten 10 benutzerbezogene Anforderungen an die Modellierung und Nutzung von Informationsmodellen festgestellt werden, wovon sieben klar voneinander abgegrenzt werden konnten.

Ad. F3: EyeTracking erweist sich im Anwendungsfall als eine geeignete Methode zur Messung und Bewertung von benutzerbezogenen Anforderungen an die Gebrauchstauglichkeit semiformaler Modellierungssprachen und Informationsmodellen.

Die aufgestellten Hypothesen (H1 bis H3) können bestätigt und aus der Untersuchung folgende Handlungsempfehlungen abgeleitet werden:

(1) Der Untersuchungsrahmen bietet sowohl eine Grundlage für den Vergleich semiformaler Modellierungssprachen als auch von Informationsmodellen. Dies sollte in Folgeuntersuchungen weiter validiert werden.

(2) Die EyeTracking Methode hatte zuvor noch keine Anwendung in Untersuchungen zur Modellierung und Nutzung von Informationsmodellen. Die Erstergebnisse dieser Untersuchung sind vielversprechend. Hier sollten jedoch zunächst weitere Untersuchungen folgen, um besonders auch bezogen auf die visuellen EyeTracking Ergebnisse (Heatmaps oder Gazeplots, [HPN09]) die Relevanz der EyeTracking Methode für Untersuchungen im Forschungsbereich der Wirtschaftsinformatik weiter validieren zu können.

5.2 Limitationen und weiterer Forschungsbedarf

In der wissenschaftlichen Literatur haben sich unterschiedliche Ansätze zur Gebrauchstauglichkeit semiformaler Modellierungssprachen herausgebildet (vgl. Kap. 2.1). Der dargestellte Untersuchungsrahmen wird auf Basis erweiterter und objektorientierter Ereignisgesteuerten Prozessketten (eEPK und oEPK) prototypisch angewandt. In wieweit andere semiformale Modellierungssprachen für die Zielsetzungen geeignet sind, bleibt weiteren Untersuchungen vorbehalten. Die derzeitigen Ergebnisse der Forschungsarbeit zu Anforderungen an die Modellierung und Nutzung von Informationsmodellen zeigen

zudem eine große Vielfalt der Definitionsansätze (vgl. Kap. 2.2). Die Bildung semantischer Charakteristika bildet dabei einen ersten Schritt zur Analyse der Anforderungsdefinitionen; Gemeinsamkeiten und Unterschiede sind hier weiter zu untersuchen.

Die wissenschaftlichen Fragestellungen sind im Rahmen zukünftiger Arbeiten weiter auszuarbeiten. Hier sind eine Vielzahl von Forschungsansätzen und Anwendungsfälle für Usability-EyeTracking Untersuchungen vorstellbar, die in Form von disziplinübergreifenden Kooperationsprojekten zwischen Vertretern der (Wirtschafts-) Informatik und der Kommunikationsforschung entwickelt werden können.

Literaturverzeichnis

[BCN92] Batini, C.; Ceri, S.; Navathe, S. B.: Conceptual Database Design: An Entity-Relationship Approach. Benjamin-Cummings, Redwood City et al., 1992.

[Be92] Bertram, M.: Aspekte der Qualitätssicherung von Unternehmensdatenmodellen. In: Mitteilungen der GI-Fachgruppe Entwicklungsmethoden für Informationswerkzeuge und deren Anwendungen, Nr. 2, 1992, S. 56 - 60.

[Be99] Becker, J.; Schütte, R.; Geib, T.; Ibershoff, H.: Grundsätze ordnungsgemäßer Modellierung (GoM). Abschlussbericht, 1999. Online verfügbar unter: http:// uni-hannover.de/ edoks/e001/303489413.pdf (zuletzt besucht 26.01.2010).

[BO88] Baroudi, J.J.; Orlikowski, W. J.: A Short-Form Measure of User Information Satisfaction: A Psychometric Evaluation and Notes on Use. In: Journal of Management Information Systems, Vol. 4, No. 4, 1988, S. 44 - 59.

[BP83] Bailey, J. E.; Pearson, S. W.: Development of a Tool for Measuring and Analysing Computer User Satisfaction. In: Management Science, Vol. 29, No. 5, 1983, S. 530-545.

[Br84] Brodie, M. L.: On the Development of Data Models, in: Brodie, M. L.; Mylopoulos, J.; Schmidt, J. W. (Eds.): On Conceptual Modelling: Perspectives from Artificial Intelligence, Databases, and Programming Languages, Springer, Berlin et al, 1984, S. 19 - 47.

[DHS96] Daneva, M.; Heib, R.; Scheer, A.-W.: Benchmarking Business Process Models. In: Arbeitsbericht 136 des Instituts für Wirtschaftsinformatik, Universität des Saarlandes, 1996, S. 1-29.

[DT88] Doll, W. J. / Torkzadeh, G.: The Measurement of End-User Computing Satisfaction. In: MIS Quarterly, Vol. 12, No. 2, 1988, S. 259 - 274.

[Du03] Duchowski, A. T.: Eye Tracking Methodology: Theory and Practice. Springer, London, 2003.

[FL03] Frank, U.; van Laak, B. L.: Anforderungen an Sprachen zur Modellierung von Geschäftsprozessen, Arbeitsberichte des Instituts für Wirtschaftsinformatik, Nr. 34, Universität Koblenz/Landau, 2003.

[Ga07] Gartner Research Group (Hrsg.): Magic Quadrant for Business Process Analysis Tools: 2H07-1H08. RAS Core Research Note G00148777, R 2337 06192008, 2007.

[HN09] Hogrebe, F.; Nüttgens, M.: Rahmenkonzept zur Messung und Bewertung der Gebrauchstauglichkeit von Modellierungssprachen: Literaturauswertung und Untersuchungsrahmen für Usability-Eyetracking-Studien. In Nüttgens, M. (Hrsg.): Arbeitsbe-

richte zur Wirtschaftsinformatik der Universität Hamburg, Nr. 7 / April 2009, Hamburg 2009.

[HPN09] Hogrebe, F.; Pagel, S.; Nüttgens, M.: Einsatz von Eyetracking zur Messung und Bewertung der Usability von Modellierungssprachen für das Prozessmanagement. Ergebnisse eines Experimentes zum praktischen Nutzen von Heatmaps und Gazeplots. In: ERP Management 5 (2009) 4, Schwerpunktheft Usability, S. 23-26, GITO-Verlag, Berlin.

[IS99] Europäische Norm EN ISO 9241-11: Ergonomische Anforderungen für Bürotätigkeiten mit Bildschirmgeräten; Teil 11: Anforderungen an die Gebrauchstauglichkeit - Leitsätze. DIN Deutsches Institut für Normung e.V., Beuth Verlag, Berlin, 1999.

[IS00] Europäische Norm EN ISO 9241-12: Ergonomische Anforderungen für Bürotätigkeiten mit Bildschirmgeräten; Teil 12: Informationsdarstellung. DIN Deutsches Institut für Normung e.V., Beuth Verlag, Berlin, 2000.

[IS08] Europäische Norm EN ISO 9241-110: Ergonomie der Mensch-System-Interaktion, Teil 110: Grundsätze der Dialoggestaltung, DIN Deutsches Institut für Normung e.V., Beuth Verlag, Berlin, 2008.

[IOB83] Ives, B.; Olson, M. H.; Baroudi, J. J.: The Measurement of User Information Satisfaction, in: Communications of the ACM, Vol. 26, No. 10, 1983, S. 785 - 793.

[KNS92] Keller, G.; Nüttgens, M.; Scheer, A. W.: "Semantische Prozeßmodellierung auf der Grundlage Ereignisgesteuerter Prozeßketten (EPK)“. Veröffentlichungen des Instituts für Wirtschaftsinformatik, Universität des Saarlandes, Heft 89, 1992.

[KS04] Kjeldskov, J., Stage, J.: New Techniques for Usability Evaluation of mobile systems. International Journal on Human-Computer Studies 60(5), 2004, 599 - 620.

[Mc76] McGee, W. C.: On User Criteria for Data Model Evaluation. In: ACM Transactions on Database Systems, Vol. 1, No. 4, 1976, S. 370 – 387.

[Mo98] Moody, D. L.: Metrics for Evaluating the Quality of Entity Relationship Models, in: Ling, T. W.; Ram, S.; Lee, M.-L. (Hrsg.): Conceptual modeling: proceedings / ER ’98, 17th International Conference on Conceptual Modeling, Lecture Notes in Computer Science, Vol. 1507, Springer, Singapore, 1998, S. 211 - 225.

[Ni03] Nielsen, J.: Usability engineering. Kaufmann, Amsterdam et al., 2003.

[Ob07] Obrist, M.; Bernhaupt, R.; Beck, E.; Tscheligi, M.: Focusing on Elderly: An iTV Usability Evaluation Study with Eye-Tracking. In: Cesar, P. et al. (Hrsg.): EuroITV 2007, Springer, Berlin, Heidelberg, 2007, S. 66 - 75.

[Pa06] Patig, S.: Die Evolution von Modellierungssprachen. Frank & Timme Verlag, Berlin, 2006.

[PG03] Pemberton, L., Griffiths, R.: Usability Evaluation Techniques for Interactive Television. In: Stephanidis, C. (Hrsg.). Proceedings of HCI International 2003 (4), 2003, S. 882-886.

[RDD08] Ramanauskas, N.; Daunys, G.; Dervinis, D.: Investigation of Calibration Techniques in Video Based Eye Tracking System, in: Miesenberger, K. et al. (Hrsg.): ICCHP 2008, Springer, Berlin, Heidelberg, 2008, S. 1208 - 1215.

[RP07] Rosen, D. E.; Purinton, E.: Website Design: Viewing the Web as a Cognitive Landscape, Journal of Business Research, 57 (7), 2007, 787 - 794.

[SB06] Sarodnick, F.; Brau, H.: Methoden der Usability Evaluation. Wissenschaftliche Grundlagen, praktische Anwendung. Huber, Bern, 2006.

[SNZ97] Scheer, A.W.; Nüttgens, M.; Zimmermann, V.: Objektorientierte Ereignisgesteuerte Prozeßketten (oEPK) – Methode und Anwendung. Institut für Wirtschaftsinformatik,

Heft 141, Universität des Saarlandes. Saarbrücken, 1997.

[WK94] Wissenschaftliche Kommission Wirtschaftsinformatik (WKWI): Profil der Wirtschaftsinformatik, in Wirtschaftsinformatik, 36. Jahrgang, Heft 1, 1994, S. 80 - 81.

[WT05] Wixom, B. H.; Todd, P. A.: A Theoretical Integration of User Satisfaction and Technology Acceptance. In: Information System Research, Vol. 16, No. 1, 2005, S. 85 - 102.

[Yo03] Yom, M.: Web usability von Online-Shops. Better-Solutions-Verlag Gierspeck, Göttingen, 2003.

Ein automatisiertes Verfahren zur Sicherstellung der konventionsgerechten Bezeichnung von Modellelementen im Rahmen der konzeptionellen Modellierung

Jörg Becker[1], Patrick Delfmann[2], Sebastian Herwig[3], Łukasz Lis[4], Andrea Malsbender[5] und Armin Stein[6]

Abstract: Eine kritische Voraussetzung für den erfolgreichen Einsatz von fach-konzeptionellen Modellen ist ihre Verständlichkeit und Vergleichbarkeit. Modelladressaten müssen in die Lage versetzt werden, den mit den Modellen vermittelten Inhalt eindeutig zu erkennen. Dies verlangt das Vorhandensein eines gemeinsamen Begriffsverständnisses unter den Modellierern. Insbesondere für Modellklassen, für deren Modellelemente sich satz- bzw. phrasenorientierte Bezeichnungspraktiken etabliert haben, bildet die Herstellung eines solchen gemeinsamen Begriffsverständnisses eine besondere Herausforderung. Der vorliegende Beitrag adressiert diese Problematik und präsentiert einen linguistischen Ansatz zur Formalisierung von Bezeichnungskonventionen für konzeptionelle Modelle. Die Konventionen werden auf Basis eines Domänenbegriffsmodells und natürlichsprachlicher Syntax formalisiert und im Rahmen der Modellierung automatisiert durchgesetzt. Der Schwerpunkt dieses Beitrags liegt auf der Analyse der durch Modellierer eingegebenen Modellelementbezeichnungen und dem im Falle von Konventionsverletzungen sich anschließenden automatischen Vorschlagen von konventionskonformen Alternativbezeichnern.

1 Motivation

Empirische Studien zeigen, dass sich arbeitsteilig erstellte konzeptionelle Modelle erheblich in ihren Bezeichnern unterscheiden können, selbst wenn der gleiche Sachverhalt adressiert wird [HS06]. Darüber hinaus ergeben sich solche Variationen auch dann, wenn die Modelle durch die gleichen Personen zu unterschiedlichen Zeitpunkten erstellt werden. Die für die Nutzung von konzeptionellen Modellen notwendige Vergleichbarkeit ist also nicht per se gegeben. Als Folge gestaltet sich die Analyse solcher Modelle – bspw. für Integrations- oder Benchmarkingzwecke – gemeinhin äußerst aufwändig [PS00; VTM08]. Die in den Bezeichnern auf unterschiedliche Art und Weise explizierte

[1] European Research Center for Information Systems (ERCIS), Westfälische Wilhelms-Universität Münster, Leonardo-Campus 3, 48149 Münster, Deutschland becker@ercis.uni-muenster.de
[2] European Research Center for Information Systems (ERCIS), Westfälische Wilhelms-Universität Münster, Leonardo-Campus 3, 48149 Münster, Deutschland delfmann@ercis.uni-muenster.de
[3] European Research Center for Information Systems (ERCIS), Westfälische Wilhelms-Universität Münster, Leonardo-Campus 3, 48149 Münster, Deutschland herwig@ercis.uni-muenster.de
[4] European Research Center for Information Systems (ERCIS), Westfälische Wilhelms-Universität Münster, Leonardo-Campus 3, 48149 Münster, Deutschland lis@ercis.uni-muenster.de
[5] European Research Center for Information Systems (ERCIS), Westfälische Wilhelms-Universität Münster, Leonardo-Campus 3, 48149 Münster, Deutschland malsbender@ercis.uni-muenster.de
[6] European Research Center for Information Systems (ERCIS), Westfälische Wilhelms-Universität Münster, Leonardo-Campus 3, 48149 Münster, Deutschland stein@ercis.uni-muenster.de

Information ist in geeigneter Weise zu vereinheitlichen, um die Vergleichbarkeit der Modelle herzustellen.

Im Kontrast zu bestehenden Ansätzen (Abschnitt 2) präsentiert der vorliegende Beitrag einen linguistischen Ansatz zur Formalisierung von Bezeichnungskonventionen für konzeptionelle Modelle, der Ambiguitäten in den Modellelementbezeichnern bereits während der Modellkonstruktion verhindert. Die Konventionen werden auf Basis eines Domänenbegriffsmodells und natürlichsprachlicher Syntax formalisiert und im Rahmen der Modellierung automatisiert durchgesetzt. Der Schwerpunkt dieses Beitrags liegt auf der Analyse der durch Modellierer eingegebenen Modellelementbezeichnungen und dem im Falle von Konventionsverletzungen sich anschließenden automatischen Vorschlagen von konventionskonformen Alternativbezeichnern. Im Rahmen des vorliegenden Beitrags wird die Funktionsweise des Ansatzes am Beispiel der englischen Sprache verdeutlicht. Hierzu werden zunächst verwandte Ansätze analysiert und vom vorliegenden Ansatz abgegrenzt (Abschnitt 2). Da eine konzeptionelle Vorstellung des Ansatzes bereits Gegenstand vorangegangener Beiträge war [DHL09a; DHL09b], wird auf die Konzeption nur kurz eingegangen (Abschnitt 3), bevor die formale Realisierung des Ansatzes detailliert dargestellt wird (Abschnitt 4). Die Anwendbarkeit des Ansatzes wird kurz anhand seiner Implementierung als Modellierungssoftware gezeigt (Abschnitt 5), bevor der Beitrag mit einem Überblick über bisherige Erfahrungen mit der Anwendung des Ansatzes und einem Ausblick auf weiteren Forschungsbedarf schließt (Abschnitt 6).

2 Verwandte Ansätze

In der Vergangenheit ist eine ganze Reihe von Ansätzen entwickelt worden, die das Problem der mangelnden Vergleichbarkeit von Bezeichnungen in konzeptionellen Modellen adressieren. Diese lassen sich durch zwei Dimensionen klassifizieren. Die erste Dimension unterteilt die Ansätze in *Ex-post-* und *Ex-ante-Ansätze*: Einerseits werden existierende Modelle untersucht, konfliktäre Bezeichner identifiziert und diese durch entsprechende Verfahren aufgelöst. Andererseits werden Verfahren vorgeschlagen mit dem Ziel, das Auftreten konfliktärer Bezeichner von vornherein zu verhindern. Die zweite Dimension klassifiziert die Ansätze nach der Struktur der betrachteten Bezeichner. Zum Einen werden ausschließlich *einzelne Wörter* als Bezeichner zugelassen, zum Anderen werden *Satzstrukturen, d. h. Phrasen*, analysiert. Darüber hinaus existiert eine Reihe von verwandten Ansätzen, die eine Normierung der natürlichen Sprache zum Ziel haben, deren Anwendungsgebiete sich jedoch *außerhalb der konzeptionellen Modellierung* befinden.

Frühe Ansätze der 1980er und 1990er Jahre adressieren bspw. das Problem der Datenbankintegration und setzen in einem ersten Schritt bei der Integration der Datenbankschemata an [BL84; BKK91; LB01; RB01]. Diese *einzelwortbezogenen Ex-post-Ansätze* fokussieren Datenmodellierungssprachen, häufig insbesondere Dialekte des Entity-Relationship-Modells (ERM [Ch76]). Durch den Vergleich von Bezeichnungen der Schemaelemente werden Ähnlichkeiten identifiziert, wobei betont wird, dass ein solcher

Vergleich ausschließlich manuell unter Einbeziehung der Schemakonstrukteure erfolgen kann.

Phrasenbezogene Ex-post-Ansätze untersuchen nicht einzelne Wörter als Bezeichner von Modellelementen, sondern sogenannte Konzepte. Konzepte umfassen mehrere Domänenbegriffe, die üblicherweise in Ontologien [Gr93; Gu98] abgelegt und durch semantische Beziehungen verbunden sind. So kann bspw. der Begriff „Rechnung" mit dem Begriff „prüfen" in Beziehung gesetzt werden, so dass bereits in der Ontologie ausgedrückt ist, dass Rechnungen geprüft werden können. Über diese Konzepte werden die Ähnlichkeitsbeziehungen zwischen Modellen hergestellt [Hö07; EKO07; Sa07].

Einzelwortbezogene Ex-ante-Ansätze, deren Ziel es ist, konfliktäre Bezeichner bereits im Vorfeld zu vermeiden, schlagen sogenannte Namenskonventionen vor. Namenskonventionen werden i. A. in Form von Glossaren oder auch Ontologien spezifiziert, die die für ein Modellierungsprojekt oder eine Modellierungsdomäne gültigen Begriffe enthalten. Während der Modellierung werden diese Vorgaben genutzt, um konventionsgerechte Modelle zu erstellen und so konfliktäre Bezeichner zu vermeiden. Entsprechende Ansätze werden bspw. von [Gr04; BDW07] für Prozessmodelle vorgeschlagen.

Phrasenbezogene Ex-ante-Ansätze wurden insbesondere in den 1990er Jahren entwickelt und fordern neben einer Standardisierung der gültigen Domänenbegriffe auch eine Standardisierung der zur Benennung von einzelnen Modellelementen erlaubten Satzstrukturen [Ro96; Ku00]. Die gültigen Begriffe werden dabei in sogenannten Fachbegriffsmodellen [KR98] abgelegt. Die Standardisierung der Satzstrukturen erfolgt durch textuelle Empfehlung. Ein alternativer Ansatz definiert Geschäftsobjekte sowie Verrichtungen auf diesen und führt sie als Anweisungen in Prozessmodellen zusammen [NZ98].

Das Problem konventionsgerechter Sprachverwendung prägt auch einige *Ansätze ohne Fokus auf konzeptionelle Modellierung*, die nicht direkt das Anwendungsgebiet der konzeptionellen Modellierung adressieren. [Or97] schlägt eine Normierung der natürlichen Sprache mithilfe eines Normlexikons der Fachbegriffe sowie einer auf Satzbauplänen basierenden Normgrammatik vor. Den Einsatzzweck einer solchen Normsprache sieht er dabei beim sogenannten *methodenneutralen Fachentwurf* von Informationssystemen im Übergang von fachlichen, natürlichsprachlichen Aussagen der Benutzer zu diagrammsprachlicher Darstellung des Entwurfs (z. B. in Form konzeptioneller Modelle). Dem Ziel der Anforderungserhebung widmen sich auch einige Ansätze aus dem Bereich des Requirements Engineering. [MK98] schlagen hierfür bspw. die Erstellung eines vorstrukturierten Glossars vor. Dies kann nicht nur manuell erfolgen, sondern auch in Form teilautomatisierter Extraktion aus natürlichsprachlichen Dokumenten. Ähnlich wie bei [Or97] werden darüber hinaus Möglichkeiten (teil-)automatisierter Überführung in konzeptionelle Modelle diskutiert.

Grundsätzlich lassen sich alle vorgestellten Ansätze der ersten vier Gruppen zur Herstellung der Vergleichbarkeit konzeptioneller Modelle anwenden. Speziell für Modellierungssprachen, die nicht einzelne Wörter sondern Sätze oder Phrasen als Bezeichner verwenden (z. B. Prozessmodellierungssprachen), sind jedoch insbesondere die einzelwort-

bezogenen Ansätze nicht verwendbar, da sich konfliktäre Bezeichner in Form von Satzstrukturen auf diese Weise nicht vermeiden lassen. Die hier vorgestellten einzelwortbezogenen Ex-ante-Ansätze wurden zwar für Prozessmodelle entwickelt, betrachten jedoch innerhalb der Satzstrukturen der Bezeichner ausschließlich einzelne Begriffe.

Hinsichtlich der Analyse von Satzstrukturen in Modellbezeichnern sind phrasenbezogene Ansätze flexibler. Ex-post-Ansätze sind in der Lage, mangelnde Vergleichbarkeit durch Gegenüberstellung der Modelle und Analyse ihrer Bezeichnungsstrukturen aufzulösen. Die vorherige Vermeidung konfliktärer Bezeichner erscheint hier allerdings zielführender, da dies eine aufwändige Analyse der fertiggestellten Modelle obsolet macht. Freilich ist eine vorherige Vermeidung konfliktärer Bezeichner nur dann möglich, wenn nicht bereits Modelle existieren, die ohne ausreichende Berücksichtigung von Bezeichnungskonventionen erstellt wurden. Die aufgeführten phrasenbezogenen Ex-ante-Ansätze bieten daher aus Sicht der Autoren eine vielversprechende Grundlage für die Formulierung eines Ansatzes zur Herstellung der Vergleichbarkeit konzeptioneller Modelle. Die Ansätze von [Ro96; Ku00] erlauben die Formulierung von Phrasenstrukturen einerseits und Begriffskonventionen andererseits. Es fehlt allerdings eine Formalisierung, die für eine automatisierte Durchsetzung von Namenskonventionen notwendig ist. Letztere ist für den Erfolg von Namenskonventionen kritisch. Der Ansatz von [NZ98] ist formalisiert, betrachtet jedoch ausschließlich die Benennung von Funktionen und Ereignissen Ereignisgesteuerter Prozessketten (EPK [KNS92]) mit einer definierten Menge an Phrasentypen. Obwohl die verwandten Ansätze zur Sprachnormierung nicht direkt auf das hier vorliegende Problemfeld übertragen werden können, liefern sie durchaus interessante Lösungsansätze. Die im Folgenden verwendeten Phrasenstrukturkonventionen entsprechen bspw. in etwa den Satzbauplänen von [Or97].

Der vorliegende Ansatz ist durch (1) seine Fokussierung auf konzeptionelle Modellierung, (2) seine Anwendbarkeit auf beliebige Modellierungssprachen, (3) die explizite Berücksichtigung von Satzfragmenten als Modellelementbezeichner, (4) seine Formalisierung und (5) die hierdurch automatisierbare Sicherstellung der konventionstreuen Modellierung von existenten Ansätzen abzugrenzen.

3 Konzeptionelle Lösung

Als konzeptionelle Basis dient ein Ordnungsrahmen für formalisierte Bezeichnungskonventionen (vgl. Abbildung 1, vgl. im Folgenden auch ausführlich [DHL09b]). Bezeichnungskonventionen basieren auf zwei wesentlichen linguistischen Bestandteilen, die abhängig von einem gegebenen Modellierungskontext (z. B. Domäne, Projekt, Unternehmen) zu spezifizieren sind. Einerseits unterliegt der natürlichsprachliche Teil jeder Modellierungssprache linguistischen Regeln. Hierbei handelt es sich neben gültigem Flexionsverhalten von Wörtern auch um die Einhaltung von syntaktischen Regeln innerhalb von Satzfragmenten. Andererseits wird durch die Modellierungsdomäne ein spezieller Domänenwortschatz vorgegeben, dessen ausschließliche Nutzung bei der Modellierung zu gewährleisten ist. Er beinhaltet eine Menge von gültigen Begriffen, die zur Nutzung

innerhalb der Modellbezeichnungen zur Verfügung stehen und repräsentiert damit eine Untermenge des durch die natürliche Sprache gegebenen Gesamtwortschatzes. Dabei sind die gültigen Substantive, Verben, Adjektive und Adverbien festzulegen, da diese die Domänensemantik enthalten. Andere Wortklassen wie z. B. Präpositionen sind generell domänenneutral und daher stets gültig.

Analog zur Einschränkung des Domänenwortschatzes ist auch die Syntax der zu verwendenden Bezeichnerphrasen dem Modellierungskontext entsprechend einzuschränken. Diese sogenannten Phrasenstrukturkonventionen werden auf Basis vordefinierter Phrasenstrukturregeln mittels sogenannter Phrasentypen verwirklicht (z. B. <*Verb, Imperativ*> <*Substantiv*>). Ähnlich wie Satzbaupläne bei [Or97] spezifizieren sie die erlaubte Anordnung von Wortarten innerhalb einer Phrase sowie deren jeweilige Flexion. Solche Phrasentypen repräsentieren eine Teilmenge der Syntax einer natürlichen Sprache (bspw. Englisch). Zur Beherrschbarkeit der hochkomplexen Syntax natürlicher Sprachen und zur Schaffung einer gemeinsamen Sprachbasis schränkt der Ansatz die Syntax ein.

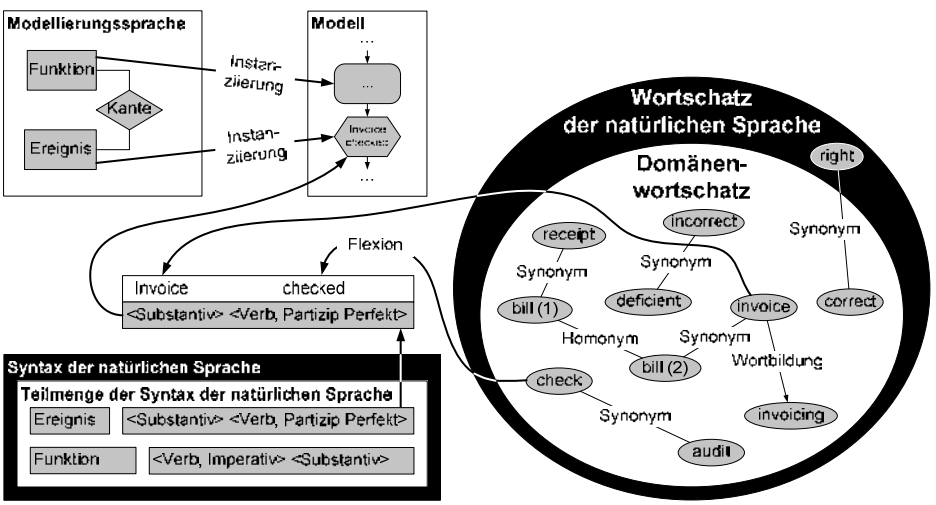


Abb. 1: Ordnungsrahmen für formalisierte Bezeichnungskonventionen

Die jeweiligen Konventionen sind abhängig von der Modellierungssprache. So werden etwa Funktionen einer EPK mit Verrichtungen bezeichnet (z. B. <*Verb, Imperativ*> <*Substantiv*>, konkret bspw. „Check invoice“), während Bezeichnungen von Ereignissen Zustände angeben (z. B. <*Substantiv*> <*Verb, Partizip Perfekt*>, konkret bspw. „Invoice checked“) [Ro96, Ku00]. Für jeden Modellelementtyp der jeweils verwendeten Modellierungssprache ist mindestens eine Phrasenstrukturkonvention zu definieren. Die konkreten Ausprägungen von Phrasenstrukturkonventionen werden mithilfe der gültigen Wörter aus dem Domänenwortschatz realisiert, die entsprechend ihrer Stellung im Satz bzw. der Bezeichnerphrase zu flektieren sind (vgl. Abbildung 1). Grundsätzlich lassen sich so beliebige syntaktische Muster formulieren. Diese Muster unterliegen ausschließlich der Restriktion, dass sie eine Teilmenge einer natürlichen Sprachsyntax repräsentieren müssen (d. h. Ketten von flektierten Wortformen, deren Instanzen entweder einen

gültigen natürlichsprachlichen Satz oder eine gültige natürlichsprachliche Phrase ergeben). Welche Muster gültig sein sollen, ist im Rahmen der Konventionendefinition festzulegen.

Zusammenfassend wird eine gemeinsame Sprachbasis für Modellierungsprojekte geschaffen, die auf einer formalen, natürlichsprachlichen Grammatik basiert. Deren Nichtterminalsymbole bilden die Phrasenstrukturen, und deren Terminalsymbole werden durch die in die Phrasen eingesetzten und korrekt flektierten Wörter repräsentiert.

Die Bezeichnungskonventionen, d. h. der Domänenwortschatz und die Phrasenstrukturkonventionen, sind einmalig für jeden Modellierungskontext zu spezifizieren, wobei auf ggf. bestehenden Konventionen aufgebaut werden kann. Idealer Weise sollte diese Aufgabe von einem Gremium von Modellierungs- und Domänenexperten vor Beginn des Modellierungsprojekts wahrgenommen werden, die in sowohl der Domäne üblichen Begriffe kennen als auch beurteilen können, welche Phrasenstrukturen für welche Modellierungssprachen geeignet sind.

Die Anwendbarkeit des vorgestellten Rahmens kann nur dann sichergestellt werden, wenn er als Modellierungswerkzeug implementiert bzw. in ein solches integriert wird. Trotz einer solchen Werkzeugunterstützung erscheint den Autoren die notwendige Auswahl von Phrasenstrukturen durch den Modellierer und die anschließende Auswahl von gültigen Domänenbegriffen (bspw. über Drop-down-Listen) sowie deren Flexion und Einsetzen in die Phrase jedoch wenig komfortabel und auch nicht effizient. Der Modellierer sollte in die Lage versetzt werden, in gewohnter Weise Modellelementbezeichner einzugeben. Die Analyse dieser Bezeichner hinsichtlich ihrer Konformität mit den Bezeichnungskonventionen und das Vorschlagen alternativer Bezeichnungen im Falle der Konventionsverletzung werden deswegen durch ein automatisiertes Verfahren realisiert.

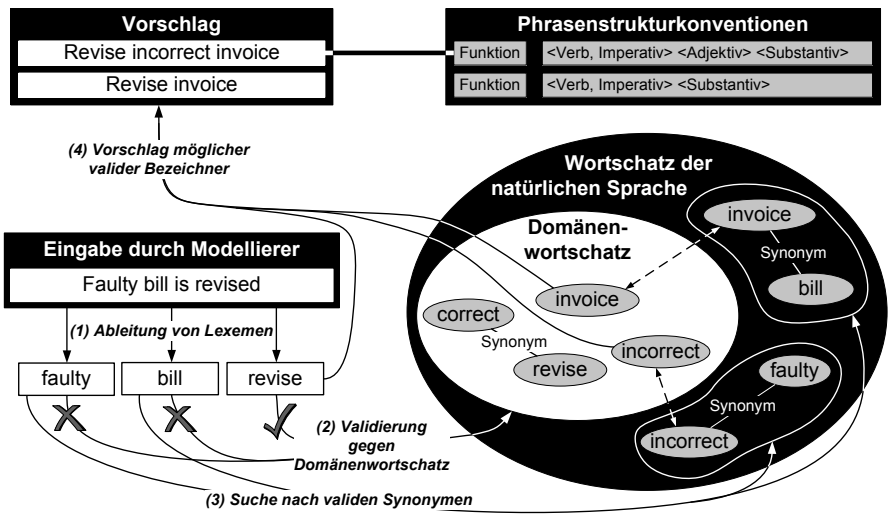


Abb. 2: Abgleich von Modellelementbezeichnern mit Modellierungskonventionen

Während der Modellierung werden die eingegebenen Bezeichnungen simultan mit den Bezeichnungskonventionen abgeglichen (vgl. im Folgenden Abbildung 2). Einerseits findet dieser Abgleich bezüglich der verwendeten Syntax statt. Andererseits werden die verwendeten Begriffe bezüglich ihrer Gültigkeit überprüft (2). Im Falle einer positiven Validierung wird die eingegebene Bezeichnung als konventionsgerecht angesehen und für das Modellelement übernommen. Genügt die eingegebene Bezeichnung den Konventionen nicht, werden dem Modellierer alternative Wörter bzw. Phrasen zur Verfügung gestellt. Im Falle ungültiger Wörter werden im Gesamtwortschatz der natürlichen Sprache Synonyme gesucht und mit dem Domänenwortschatz abgeglichen (3). Gibt der Modellierer bspw. das Wort „bill“ ein und entspricht dies nicht den Konventionen, so wird die Menge der Synonyme zu „bill“ mit dem Domänenwortschatz abgeglichen. Ein entsprechend gültiges Synonym ist bspw. „invoice“, das dem Modellierer als Alternative angeboten wird. Genügen alle Wörter den Konventionen, wird geprüft, ob die eingegebene Phrase den Phrasenstrukturkonventionen entspricht. Ist die Phrase nicht korrekt, so werden die zur Verfügung stehenden, bereits auf Gültigkeit überprüften Wörter in die möglichen Phrasen eingesetzt (4). Der Modellierer kann auf dieser Basis eine Entscheidung treffen, welche der aufgelisteten Phrasen den gewünschten Inhalt am besten wiedergibt. Durch die Möglichkeit, innerhalb des Domänenwortschatzes die einzelnen Beschreibungen der verwendeten Wörter nachzuschlagen, werden Missverständnisse auf Seiten des Modellierers ausgeschlossen.

Zur Realisierung eines solchen automatisierten Verfahrens sind neben der formalen Spezifikation gültiger Phrasen und Domänenbegriffe folgende technische Komponenten notwendig:

- ***Linguistischer Parser (LP):*** Linguistische Parser analysieren die Struktur eines Satzes oder einer Satzphrase und liefern die syntaktische Beschreibung der Phrasenstruktur sowie die Grundformen der verwendeten Wörter (die sogenannten *Lexeme*) zurück. Die konkrete Phrasenstruktur einer eingegebenen Satzphrase kann so gegen die erlaubten Phrasentypen und die verwendeten Lexeme gegen das Domänenlexikon validiert werden. Das hierfür entwickelte automatisierte Verfahren wird im Abschnitt 4 ausführlich präsentiert.
- ***Lexikalische Dienste:*** Entsprechen die eingegebenen Wörter nicht den Konventionen, so wird ein *allgemeines Lexikon der verwendeten natürlichen Sprache (AL)* nach entsprechenden Synonymen durchsucht, die wiederum mit dem Domänenwortschatz abgeglichen werden. Auf diese Weise können gültige Alternativen zu nicht validen Wörtern gefunden und vorgeschlagen werden. Voraussetzung hierfür ist, dass in der Datenbasis des Lexikons ein umfassender Wortschatz sowie entsprechende Synonymbeziehungen zwischen den Wörtern hinterlegt und abrufbar sind. Entsprechen die verwendeten Wörter oder Phrasen nicht den Konventionen, so ist eine alternative Phrase mit gültigen Wörtern zu konstruieren. Hierfür ist es notwendig, diese Wörter entsprechend ihrer Stellung im Satz automatisiert mithilfe eines *Flexionsdienstes (FD)* zu beugen. Wird ein Wort verwendet, welches nicht den Konventionen entspricht, wohl aber ein Wort,

das von diesem abstammt oder umgekehrt, so kann das gültige Wort mithilfe eines *Wortbildungsdienstes (WBD)* ermittelt werden.

4 Automatisierung

Grundlage für die Durchsetzung von Modellierungskonventionen bildet ein teilweise durch Nutzerinteraktion gestütztes automatisiertes Verfahren, das auf den Domänenwortschatz und die Phrasenstrukturkonventionen sowie die in Abschnitt 3 angesprochenen externen technischen Komponenten zugreift. Die Realisierung wird im Folgenden schrittweise mithilfe einer differenzierten Betrachtung erläutert.

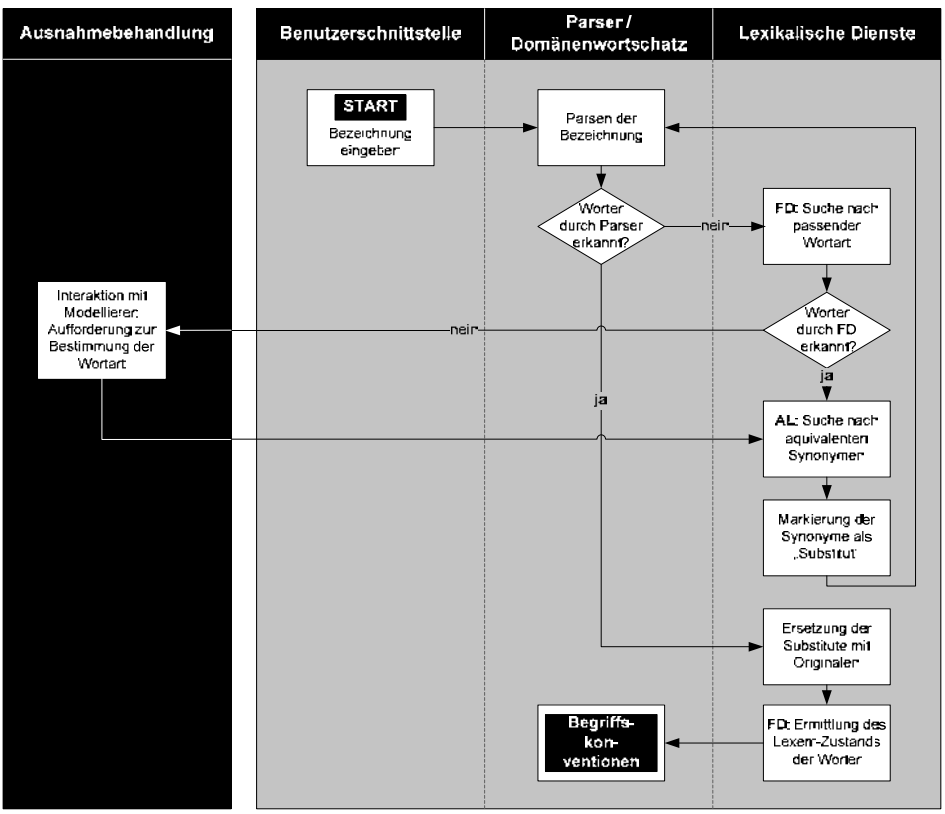


Abb. 3: Analyse der Benutzereingabe

Nach Eingabe der Bezeichnung eines Modellelements wird die Phrase mithilfe des Parsers in ihre Wortbestandteile zerlegt (vgl. Abbildung 3). Die einzelnen Wörter werden hinsichtlich ihrer Wortart und -form durch den Parser klassifiziert. Sollte der Parser ein Wort nicht erkennen, so wird das betroffene Wort an den Flexionsdienst übergeben. Mithilfe dieses Dienstes wird seine Wortart bestimmt. Ist auch durch den Flexionsdienst

keine eindeutige Bestimmung möglich, so findet eine Ausnahmebehandlung bezüglich des betroffenen Wortes statt und der Einsatz des Modellierers wird erforderlich. Kann durch den Flexionsdienst die Wortart bestimmt werden, so wird im allgemeinen Lexikon der natürlichen Sprache ein passendes Synonym ausgewählt und in der identischen Flexionsform in die Bezeichnung als Substitut eingesetzt. Diese Ersetzung erweist sich als unumgänglich, da der Parser ansonsten ggf. fehlerhafte Informationen bezüglich der Wortart und -form der anderen abhängigen Wörter der Phrase zurückliefert. Nach Ersetzung des nicht erkannten Wortes wird die modifizierte Bezeichnung erneut geparst.

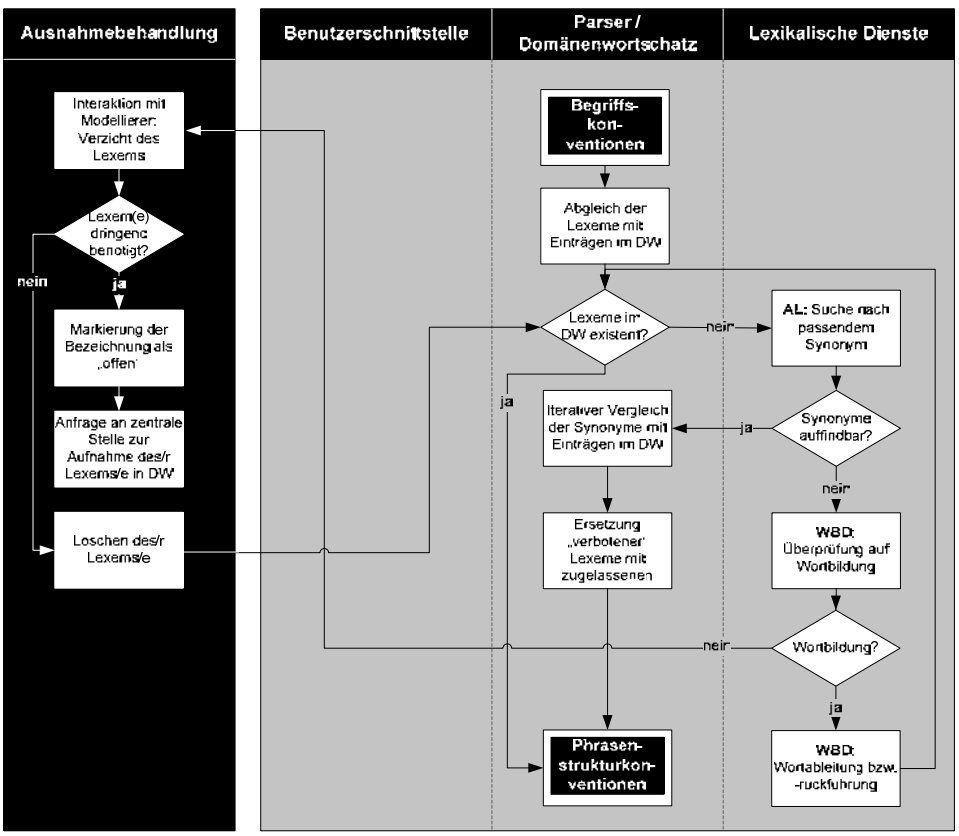


Abb. 4: Abgleich der Benutzereingabe mit dem Domänenwortschatz

Werden alle Wörter durch den Parser erkannt oder bereits nicht erkannte Wörter durch Substitute ersetzt, so werden die identifizierten Wortarten und -formen für die spätere Verwendung zwischengespeichert. In einem weiteren Schritt werden ggf. bestehende Substitute durch ihre Originale ersetzt. Die Wörter der Phrase werden durch den Flexionsdienst in ihre Lexemform transformiert und wiederum zwischengespeichert. Auf Basis dieser Lexeme wird der Abgleich der Begriffskonventionen im zweiten Schritt des Verfahrens durchgeführt. Dieser richtet sich an die domänenspezifischen Begriffe, die

durch die in der Phrase verwendeten Substantive, Verben und Adjektive repräsentiert sind. Der Abgleich mit dem Domänenwortschatz wird somit ausschließlich für diese Wortarten durchgeführt (vgl. Abbildung 4).

Die zuvor zwischengespeicherten Wörter werden mit den Einträgen im Domänenwortschatz abgeglichen. Ist ein Wort nicht Bestandteil des Domänenwortschatzes, wird mithilfe des allgemeinen Lexikons der verwendeten natürlichen Sprache nach zugehörigen Synonymen gesucht. Die aufgefundenen Synonyme werden im Anschluss ebenfalls mit den Einträgen im Domänenlexikon abgeglichen. Bei Auffinden eines passenden Synonyms wird das gemäß den Konventionen nicht erlaubte Wort durch das Synonym ersetzt.

Werden im allgemeinen Lexikon keine passenden Synonyme gefunden, so wird von einer Wortbildung ausgegangen. Eine Wortbildung bezeichnet die Abstammung eines Wortes von einem anderen. Bspw. stammt das Substantiv „approval“ vom Verb „approve“ ab. Ob eine Wortbildung vorliegt, wird mithilfe des Wortbildungsdienstes überprüft. Handelt es sich um eine solche, so werden entsprechend abstammende Wörter bzw. das zu Grunde liegende Wort ermittelt. Die resultierenden Wörter erfahren im Anschluss ebenfalls einen Abgleich mit den Einträgen im Domänenwortschatz. Handelt es sich laut Wortbildungsdienst nicht um eine Wortbildung, findet eine Ausnahmebehandlung statt. Hierbei wird mit dem Modellierer interagiert und ein möglicher Verzicht des Wortes erfragt. Stimmt der Modellierer einem solchen Verzicht zu, wird das betroffene Wort gelöscht und das Verfahren mit den restlichen Wörtern weiter durchlaufen. Anderenfalls wird die gesamte Bezeichnung als „offen“ markiert und eine Anfrage an eine zentrale Stelle – das bereits in Abschnitt 3 erwähnte Expertenteam – durchgeführt. Diese zentrale Stelle entscheidet, ob von Modellierern angefragte Erweiterungen mit in die Konventionen aufgenommen werden oder nicht. Im Falle der Löschung eines nicht validen Wortes wird die Bearbeitung der Bezeichnung mit den restlichen Wörtern fortgesetzt.

Sind alle Wörter mit dem Domänenwortschatz abgeglichen und dementsprechend als konventionstreu bestätigt, wird im dritten Teil des Verfahrens die Anpassung der Bezeichnung an die gegebenen Phrasenstrukturen vollzogen. Diese erfordert in einem ersten Schritt einen Abgleich der verfügbaren Wortarten mit den gegebenen Phrasenstrukturkonventionen (vgl. Abbildung 5). Sind zur Realisierung der Phrasenstrukturen nicht alle benötigten Wortarten vorhanden, findet erneut eine Ausnahmebehandlung statt. Bei dieser wird der Modellierer aufgefordert, das fehlende Wort der vorgegebenen Wortart zu ergänzen. Das dabei eingegebene Wort wird ebenfalls mit den Begriffskonventionen abgeglichen. Es wird zu der bereits bestehenden Liste der verfügbaren Wörter ergänzt und durchläuft den zweiten Schritt des Verfahrens erneut.

Sind alle Wortarten zur Realisierung der möglichen Phrasenstrukturen vorhanden, wird umgekehrt geprüft, ob bei der Realisierung möglicher Phrasen einige Wörter nicht verwendet werden können, d. h. ob die vom Modellierer ursprünglich eingegebene Phrase zu viele Wörter enthält. Der Modellierer wird auf einen möglichen Verlust hingewiesen und bezüglich der Notwendigkeit dieser Wörter befragt. Verzichtet der Modellierer auf überzählige Wörter, werden sie aus der Liste der verfügbaren gelöscht. Anderenfalls

wird die Bezeichnung als „offen“ markiert und eine Anfrage an die bereits erwähnte zentrale Stelle zur etwaigen Neuaufnahme eines Phrasentyps mit einer ausreichenden Zahl an Wörtern in die Phrasenstrukturkonventionen gestellt.

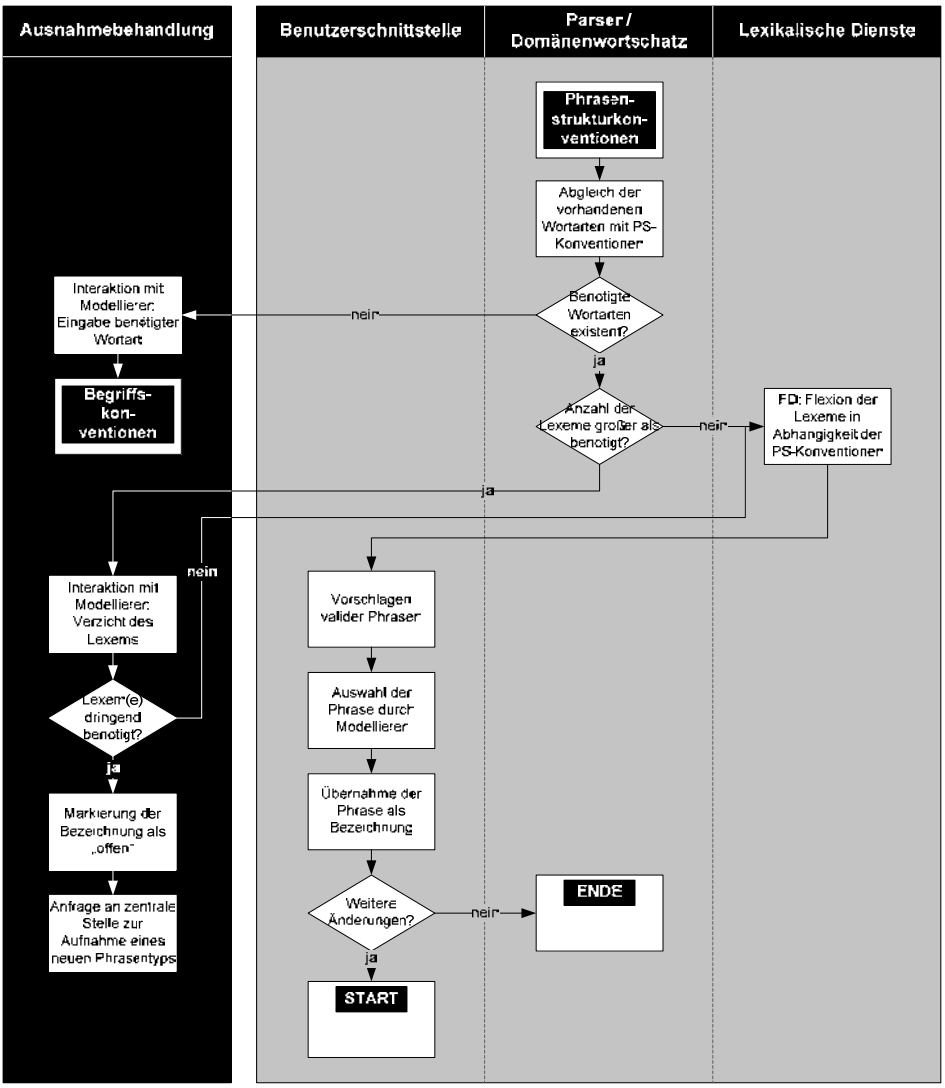


Abb. 5: Abgleich der Benutzereingabe mit den Phrasenstrukturkonventionen

Nachdem die notwendigen Wortarten hinsichtlich der Phrasenstrukturen abgeglichen sind, werden sie mithilfe des Flexionsdienstes in die jeweils benötigten Wortformen flektiert und an die Phrasenstruktur angepasst. Dabei werden dem Modellierer alle syntaktisch möglichen Phrasen angeboten. Existieren exemplarisch zwei Substantive (z. B.

„invoice“ und „receipt“) und ein Verb (z. B. „check“), und die Phrasenstrukturkonvention hat folgende Erscheinung: <*Verb, Imperativ*> <*Substantiv*>, so ergeben sich die Möglichkeiten „Check invoice“ und „Check receipt“.

Diese Möglichkeiten werden dem Modellierer in Form einer Liste präsentiert, woraufhin dieser eine Auswahl trifft. Die ausgewählte Phrase wird in die Bezeichnung übernommen und der Modellierer nach weiteren Änderungen gefragt. Sind weitere Änderungen gewünscht, wird die gesamte Phrase an den Startpunkt des Verfahrens übergeben und erneut untersucht. Wünscht der Modellierer keine weiteren Änderungen, wird die Bezeichnung als valide gekennzeichnet und die Ausführung abgeschlossen.

Die Komplexität dieses Verfahrens vermittelt einen Eindruck des Aufwands, der mit einer manuellen Sicherstellung der Konventionstreue von Modellelementbezeichnern verbunden wäre. Durch die Automatisierung bleibt diese Komplexität für den Modellierer jedoch verborgen, da sie im Hintergrund abläuft. Der Modellierer erhält nur dann Hinweise und Alternativvorschläge für die Benennung eines Modellelements, wenn er eine Konvention verletzt.

5 Technische Realisierung

Eine technische Umsetzung des hier vorgestellten Ansatzes liegt als Modellierungsprototyp vor. Der Modellierer wird durch Popup-Fenster auf etwaige Konventionsverletzungen hingewiesen und erhält konventionstreue Alternativvorschläge (vgl. Abbildung 6).

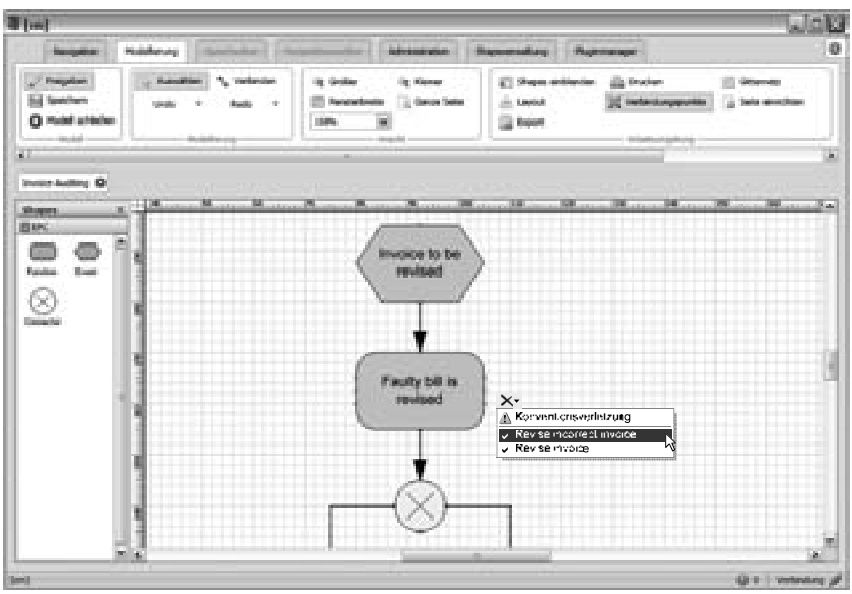


Abb. 6: Umsetzung des Ansatzes als Modellierungswerkzeug

Das im Modellierungswerkzeug realisierte Verfahren greift auf die bereits in Abschnitt 3 angesprochenen technischen Komponenten über Schnittstellen zu. Es existiert eine Vielzahl solcher Dienste, deren Funktionsumfang, Schnittstellenkonfiguration und Leistung erheblich variieren. Gemäß den Anforderungen des vorgestellten Ansatzes wurden diese Dienste analysiert und entsprechend geeignete ausgewählt:

- ***Linguistischer Parser:*** Als Parser mit umfangreicher Funktionalität stehen der *IPS-Parser* [We92], der *PET-Parser* [Ca00], der *XIP-Parser* [Xe09] und der *Stanford-Parser* [KM03] zur Verfügung. Zur Unterstützung des vorgestellten Ansatzes ist es notwendig, dass die grammatikalische Struktur der Phrase als Anordnung von Wortarten mit Angabe derer Flexionsformen durch einen Parser identifiziert und in strukturierter Form zur Validierung gegen erlaubte Phrasentypen bereitgestellt werden. Solange diese Ausgabe mit dem Spezifikationsformat der Phrasentypen kompatibel ist, sind die interne Arbeitsweise des Parsers und der dort eventuell verfolgte grammatikalische Formalismus für das hier vorgestellte Verfahren irrelevant. Die Parser *XIP* und *PET* sowie der *Stanford-Parser* bieten eine umfassende und strukturierte Ausgabe von grammatikalischen Eigenschaften. Beim *XIP*- und *PET-Parser* erfolgt dies in Form einer Datei. Der *Stanford-Parser* stellt hingegen eine native Schnittstelle zur Verfügung, über die sowohl Analysen angestoßen als auch die Ergebnisse abgefragt werden können. Entgegen dem *XIP*- und *IPS-Parser*, die ausschließlich als webbasierte Demonstrationen verfügbar sind, ist beim *Stanford-Parser* hierdurch auch die Möglichkeit gegeben, diesen eng in Softwareumgebungen zu integrieren. Um mit einem simultanen Abgleich der eingegebenen Bezeichnungen mit den Bezeichnungskonventionen einen durchgängigen Modellierungsfluss gewährleisten zu können, ist zudem eine geringe Antwortzeit notwendig. Von den untersuchten Parsern weisen sowohl der *PET-Parser* als auch der *Stanford-Parser* eine Verarbeitungsgeschwindigkeit auf, die einen durchgängigen Modellierungsfluss zulässt. Anhand der dargestellten Anforderungen und aufgrund der freien Verfügbarkeit hat sich der *Stanford-Parser* für den Modellierungsprototyp als am besten geeignet erwiesen.

- ***Lexikalische Dienste:*** Mit *GermaNet* [KL02], *wortschatz.uni-leipzig.de* und *canoo .net* sind Lexika für das Deutsche und mit *WordNet* [Fe98] das Englische verfügbar, die einen umfassenden Wortschatz sowie Synonymbeziehungen bereitstellen. Im Rahmen der prototypischen Realisierung wurde der vorgestellte Ansatz exemplarisch auf die englische Sprache angewendet. Als allgemeines Lexikon wird der weit verbreitete und sprachwissenschaftlich fundierte Dienst *WordNet* eingesetzt, der neben seinem Synonymdienst Wortbildungsfunktionalitäten bereitstellt. Darüber hinaus liefert *WordNet* auch Flexionsinformationen, eine automatisierte Durchführung der Flexion wird jedoch nicht unterstützt. Da im Englischen eine zumeist regelbasierte Flexion möglich ist, konnte diese Funktionalität in Eigenentwicklung realisiert werden.

6 Fazit und Ausblick

Die Einhaltung von Konventionen bei der Benennung von Elementen in konzeptionellen Modellen ermöglicht die Realisierung eines gemeinsamen Begriffsverständnisses. Modelladressaten können auf diese Weise den mit den Modellen vermittelten Inhalt eindeutig und einheitlich wahrnehmen. Nach Auffassung der Autoren kann die faktische Durchsetzung solcher Konventionen nur dann erfolgen, wenn sie bereits während der Modellierung automatisiert stattfindet.

Ferner darf eine effektive Unterstützung zur Einhaltung von Modellierungskonventionen den Modellierungsprozess nicht maßgeblich beeinträchtigen. Vielmehr muss die Modellierung in gewohnter Form stattfinden können, ohne dass der Modellierer sich selbst um die Einhaltung der Konventionen zu kümmern hat. Das in Abschnitt 4 vorgestellte Verfahren übernimmt diese Aufgabe und verlagert den mit der Einhaltung von Modellierungskonventionen verbundenen Aufwand in den Hintergrund.

Die Ergebnisse erster Tests des in Abschnitt 5 vorgestellten Modellierungsprototyps zeigen, dass eine performante und effektive Modelliererunterstützung realisiert werden konnte. Konkret wurden den Tests ein Domänenwortschatz mit ca. 200 Lexemen und ca. 15 linguistische Patterns zu Grunde gelegt. Zukünftige Tests werden auch wesentlich größere Wortschätze berücksichtigen. Es hat sich allerdings gezeigt, dass die Hauptrechenzeit nicht auf den Abgleich des Wortschatzes entfällt (der in Datenbankform vorliegt, so dass bekannte Optimierungen, bspw. Indizes, auch bei relativ großen Wortschätzen greifen), sondern auf die Recherche im allgemeinen Lexikon.

Das entwickelte Verfahren reduziert das Eingreifen des Modellierers auf ein Minimum. Wenn die eingegebenen Bezeichner mit den festgelegten Konventionen konform sind, werden diese direkt akzeptiert. Anderenfalls werden dem Modellierer den Konventionen entsprechende Alternativvorschläge unterbreitet, aus denen lediglich ein passender ausgewählt werden muss. Befürchtungen, der Modellierungsprozess könne durch lange Antwortzeiten der linguistischen Dienste und häufige Störungen in Form von alternativen Bezeichnungsvorschlägen zu sehr beeinträchtigt werden, konnten nicht bestätigt werden. Vielmehr bewegen sich die Antwortzeiten der Dienste im Untersekundenbereich. Die aus der Ermittlung der Alternativvorschläge resultierende Zeitverzögerung ist minimal und auf Seiten des Modellierers kaum wahrnehmbar. Das automatische Vorschlagen alternativer Bezeichner wird ersten Beobachtungen zufolge nicht als störend empfunden, sondern sogar begrüßt.

Obwohl die ersten Ergebnisse zufriedenstellend sind, steht eine umfangreiche Evaluation noch aus. Dafür sind in erster Linie Laborexperimente geplant, in denen die gleiche Modellierungsaufgabe von drei Gruppen (ohne Konventionen, mit papierbasierten Konventionen, mit Werkzeugunterstützung) bearbeitet wird. Dabei werden die Bearbeitungszeiten der einzelnen Gruppen verglichen. Zudem werden die resultierenden Modelle mit den Konventionen abgeglichen. In diesem Rahmen ist auch der zusätzliche Aufwand zu ermitteln, den eine nachträgliche Anpassung der Modelle nach sich zieht, welche ohne methodische Unterstützung zur Einhaltung der Modellierungskonventionen erstellt wur-

den. Darüber hinaus ist die Frage der Nutzerakzeptanz zu stellen. Obwohl erste Tests gezeigt haben, dass die Modellierer entgegen evtl. zu befürchtenden Akzeptanzbarrieren das „Erzwingen“ konventionstreuer Modelle sogar begrüßen, muss dieser Aspekt mittelfristig zum Gegenstand konkreter Fallstudien gemacht werden.

Der Umgang der zur Verfügung stehenden Parser mit Nominalphrasen (z. B. „bill of material“) stellt bisher noch ein Problem dar. Die Erkennung solcher Phrasen in gegebenen Satzfragmenten wird von den getesteten Parsern nicht unterstützt. Konkret müssten solche Nominalphrasen zusammengefasst und komplett an einen Synonymdienst übergeben werden, um valide Alternativphrasen zu finden. Eine entsprechende Erweiterung des Ansatzes ist geplant. Konkret könnte die Erkennung solcher Nominalphrasen analog zu den Phrasenstrukturkonventionen durch syntaktische Muster realisiert werden.

In der momentanen Implementierung des Ansatzes ist pro Bedeutung exakt ein Lexem vorgesehen. Zwar lässt das Domänenlexikon die Definition von Synonymen zu, als gültig gilt jedoch immer das als „dominant“ markierte Lexem. Im Weiteren ist zu prüfen, ob diese eher strenge Verfahrensweise beizubehalten oder zu lockern ist. Alternativ könnten bspw. auch die nicht dominanten, im Domänenlexikon als synonym definierten Lexeme als gültig angesehen werden.

Tests bzgl. der Implementierung des Ansatzes auf Basis der *deutschen* Sprache haben sowohl Erschwernisse als auch Erleichterungen gezeigt. Erleichterungen bestehen bspw. darin, dass anstatt von Substantivketten (wie im Englischen üblich) zusammengesetzte Substantive verwendet werden. Dies macht ihre Identifikation als Einheit einfacher. Erschwernisse bestehen bspw. in der automatisierten Flexion bzw. Rückführung in die Lexemform. Hier kommt ausschließlich tabellenbasierte Flexion in Frage, da sich der Großteil der deutschen Wörter nicht nach festen Regeln flektieren lässt. Entsprechende (kostenpflichtige) Onlinedienste stehen jedoch auch für die deutsche Sprache zur Verfügung (z. B. *canoo.net*). Eine vollständige Realisierung des Ansatzes auf Basis der deutschen Sprache ist mittelfristig geplant.

Literaturverzeichnis

[BDW07] Born, M.; Dörr, F.; Weber, I.: User-friendly semantic annotation in business process modelling: In: Weske, M., Hacid, M.-S., Godart, C. (Hrsg.): Proceedings of the International Workshop on Human-Friendly Service Description, Discovery and Matchmaking (Hf-SDDM 2007) at the 8th International Conference on Web Information Systems Engineering (WISE 2007). Nancy 2007, S. 260-271.

[BKK91] Bhargava, H. K.; Kimbrough, S. O.; Krishnan, R.: Unique Name Violations, a Problem for Model Integration or You Say Tomato, I Say Tomahto. ORSA Journal on Computing 3 (1991) 2, S. 107-120.

[BL84] Batini, C.; Lenzerini, M.: A Methodology for Data Schema Integration in the Entity Relationship Model. IEEE Transactions on Software Engineering 10 (1984) 6, S. 650-663.

[Ca00] Callmeier, U.: PET – A Platform for Experimentation with Efficient HPSG Processing Techniques. Natural Language Engineering 6 (2000) 1, S. 99-108.

[Ch76] Chen, P. P.-S.: The Entity-Relationship Model – Toward a Unified View of Data. ACM Transactions on Database Systems 1 (1976) 1, S. 9-36.

[DHL09a] Delfmann, P.; Herwig, S.; Lis, L.: Konfliktäre Bezeichnungen in Ereignisgesteuerten Prozessketten – Linguistische Analyse und Vorschlag eines Lösungsansatzes. In: Proceedings des 8. GI-Workshops EPK 2009: Geschäftsprozessmanagement mit Ereignisgesteuerten Prozessketten. Berlin 2009.

[DHL09b] Delfmann, P.; Herwig, S.; Lis, L.: Unified Enterprise Knowledge Representation with Conceptual Models – Capturing Corporate Language in Naming Conventions. In: Proceedings of the 30th International Conference on Information Systems (ICIS 2009). Phoenix, Arizona, USA, 2009.

[EKO07] Ehrig, M.; Koschmider, A.; Oberweis, A.: Measuring Similarity between Semantic Business Process Models. In: Roddick, J. F.; Hinze, A. (Hrsg.): Proceedings of the Fourth Asia-Pacific Conference on Conceptual Modelling (APCCM) 2007. Ballarat 2007, S. 71-80.

[Fe98] Fellbaum, C. (Hrsg.): WordNet: An Electronic Lexical Database. Cambridge 1998.

[Gr04] Greco, G.; Guzzo, A.; Pontieri, L.; Saccà, D.: An ontology-driven process modeling framework. In: Galindo, F.; Takizawa, M.; Traunmüller, R. (Hrsg.): Proceedings of the 15th International Conference on Database and Expert Systems Applications (DEXA 2004). Zaragoza 2004, S. 13-23.

[Gr93] Gruber, T. R.: A Translation Approach to Portable Ontology Specifications. Knowledge Acquisition 5 (1993) 2, S. 199-220.

[Gu98] Guarino, N.: Formal Ontology and Information Systems. In: N. Guarino (Hrsg.): Proceedings of the 1st International Conference on Formal Ontologies in Information Systems. Trento 1998, S. 3-15.

[Hö07] Höfferer, P.: Achieving business process model interoperability using metamodels and ontologies. In: Österle, H.; Schelp, J.; Winter, R. (Hrsg.): Proceedings of the 15th European Conference on Information Systems. St. Gallen 2007, S. 1620-1631.

[HS06] Hadar, I.; Soffer, P.: Variations in conceptual modeling: classification and ontological analysis. Journal of the AIS 7 (2006) 8, S. 568-592.

[KL02] Kunze, C.; Lemnitzer, L.: GermaNet – representation, visualization, application. In: Proceedings of the LREC 2002 Conference, Volume V., S. 1485-1491.

[KM03] Klein, D.; Manning, C. D.: Accurate unlexicalized parsing. In: Proceedings of the 41st Meeting of the Association for Computational Linguistics – Volume 1. Sapporo 2003, S. 423-430.

[KNS92] Keller, G.; Nüttgens, M.; Scheer, A.-W.: Semantische Prozeßmodellierung auf der Grundlage „Ereignisgesteuerter Prozeßketten (EPK)“. In: A.-W. Scheer (Hrsg.): Veröffentlichungen des Instituts für Wirtschaftsinformatik 89. Saarbrücken 1992.

[KR98] Kugeler M.; Rosemann, M.: Fachbegriffsmodellierung für betriebliche Informationssysteme und zur Unterstützung der Unternehmenskommunikation. In: Informationssystem Architekturen. Fachausschuss 5.2 der Gesellschaft für Informatik e. V. (GI), 5

(1998) 2, S. 8-15.

[Ku00] Kugeler, M.: Informationsmodellbasierte Organisationsgestaltung. Modellierungskonventionen und Referenzvorgehensmodell zur prozessorientierten Reorganisation. Berlin 2000.

[LB01] Lawrence, R.; Barker, K.: Integrating Relational Database Schemas using a Standardized Dictionary. In: Proceedings of the 2001 ACM symposium on Applied computing (SAC). Las Vegas 2001.

[MK98] Mayr, H. C.; Kop, C.: Conceptual Predesign – Bridging the Gap between Requirements and Conceptual Design. In: Proceedings of the 3rd International Conference on Requirements Engineering (ICRE'98) 1998, S. 90-100.

[NZ98] Nüttgens, M.; Zimmermann, V.: Geschäftsprozeßmodellierung mit der objektorientierten Ereignisgesteuerten Prozeßkette (oEPK). In: Maicher, M.; Scheruhn, H.-J. (Hrsg.): Informationsmodellierung – Branchen, Software- und Vorgehensreferenzmodelle und Werkzeuge. Wiesbaden 1998, S. 23-36.

[Or97] Ortner, E.: Methodenneutraler Fachentwurf. Stuttgart 1997.

[PS00] Phalp, K.; Shepperd, M.: Quantitative analysis of static models of processes. Journal of Systems and Software 52 (2000) 2-3, S. 105-112.

[RB01] Rahm, E.; Bernstein, P. A.: A Survey of Approaches to Automatic Schema Matching. The International Journal on Very Large Data Bases 10 (2001) 4, S. 334-350.

[Ro96] Rosemann, M.: Komplexitätsmanagement in Prozeßmodellen. Methodenspezifische Gestaltungsempfehlungen für die Informationsmodellierung. Wiesbaden 1996.

[Sa07] Sabetzadeh, M.; Nejati, S.; Easterbrook, S.; Chechik, M.: A Relationship-Driven Framework for Model Merging, Workshop on Modeling in Software Engineering (MiSE'07). 29th International Conference on Software Engineering, Minneapolis 2007.

[VTM08] Vergidis, K.; Tiwari, A.; Majeed, B.: Business process analysis and optimization: beyond reengineering. IEEE Transactions on Systems, Man, and Cybernetics 38 (2008) 1, S. 69-82.

[We92] Wehrli, E.; Clar, R.; Merlo, P.; Ramluckun, M.: The IPS system. In: Boitet, C. (Hrsg.): Actes du quinzième colloque international en linguistique informatique. Coling 1992, S. 870–874.

[Xe09] Xerox Incremental Parser (XIP). http://orchid.xrce.xerox.com. 12.09.2009.

Modeling Systems of Systems as Nested Actor Systems Based on Petri Nets

Matthias Wester-Ebbinghaus[1], Daniel Moldt[2] and Simon Adameit[3]

Abstract: Modern software systems are frequently characterized as systems of systems. Agent-orientation as a software engineering paradigm exhibits a high degree of qualification for addressing many of the accompanying challenges. However, systems of systems demand for means of hierarchical/recursive decomposition that are not inherently rooted in the agent-oriented paradigm. We present a model that still relies on the actor metaphor, but shifts the focus to collective agency. We propose a universal model of a system unit that both embeds system actors and is itself embedded as a collective system actor in surrounding system units. Consequently, we can apply our model of a system unit at arbitrary levels of a system of systems and compose the overall system by means of nested actor hierarchies. (High Level) Petri nets as our modeling technique supply precise operational semantics for the functioning of these kind of systems. In addition, we offer abstraction mechanisms that allow for rather high-level or low-level views and smooth transitions between them.

1 Introduction

Modern software systems are frequently characterized as inherently distributed and heterogeneous *systems of systems* [Mai99] whose parts potentially exhibit a great deal of operational and managerial independence [LMW05, Nor06, HHVE07]. An increasing emphasis is laid on the co-evolution of software and social systems that together form socio-technically integrated information systems. Among the large spectrum of work concerned with these kinds of systems we take agent-orientation as a vantage point for our work presented here. Agent-orientation as a software engineering paradigm is in many respects very well suited to deal with the above mentioned kind of systems. It advocates flexible, high-level interactions between system parts that exhibit a great deal of local freedom and initiative (agents). Furthermore, it fosters socio-technical integration by applying concepts to the software-technical side that are congruent to the real-world inspirations: positions, roles, services, delegation, speech-act based interactions, norms, etc.

However, it is the system of systems aspect that poses problems for agent-orientation. A system of systems perspective inherently demands for different levels of abstractions with

[1] University of Hamburg, Department of Informatics, Vogt-Kölln-Straße 30, D-22527 Hamburg, wester@informatik.uni-hamburg.de
[2] University of Hamburg, Department of Informatics, Vogt-Kölln-Straße 30, D-22527 Hamburg, moldt@informatik.uni-hamburg.de
[3] University of Hamburg, Department of Informatics, Vogt-Kölln-Straße 30, D-22527 Hamburg, 6adameit@informatik.uni-hamburg.de

regards to the granularity of system parts studied at each level. This is especially true under a socio-technical perspective as social systems inherently exhibit multiple levels of (actor) abstraction [Sco03]. Agent-orientation on the other hand features a rather flat decomposition, namely "decomposing problems in terms of autonomous agents" [Jen00]. As analyzed in [BvdT05] the *component view* of objects where objects may be composed of other objects in a recursive fashion was lost with the transition from object- to agent-orientation.[4]

Consequently, we have derived an approach that still relies on the actor metaphor but shifts the focus from the classical individual agent concept to the concept of a collective actor. The basic idea is depicted in Figure 1 in UML style. We use the general term *system unit* for

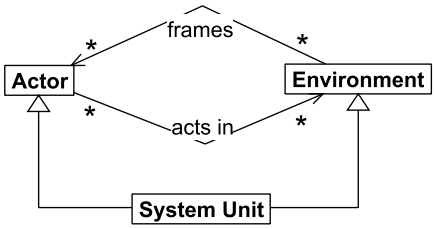


Fig. 1: Basic system unit concept

collective actors at the software level in order to express its capacity to provide a building block for wider systems while at the same time being a differentiated and useful entity of its own. A system unit consists of actors that embody it and and to whom the system unit provides an environment. While the embedded actors are for one part engaged in activities that are completely internal to the system unit, they are also engaged in activities where they act *on behalf of* the system unit in wider surroundings that are again system units. Thus, they make the system unit a collective actor that acts „as a whole" in surrounding system units. We arrive at a recursive understanding of systems of systems composed of collective actors. Of course such a system also includes individual actors at some point. These can be understood as classical agents.

The paper at hand deals with one specific aspect of our overall approach and closes the gap between different themes of earlier work. Figure 2 helps us to position our work both in the wider context of our research efforts and in relation to other approaches.

Our wider approach ORGAN (**Org**anizational **A**rchitecture with **N**ets) evolves around an architectural proposal for software systems of systems that we term *multi-organization systems (MOS)*. The concept of a MOS is inspired by real-world organizational settings. It features software units derived from different kinds of collective social actors, e.g. *teams, departments, enterprises* and *organizational fields*. Each of these units is characterized by a distinct configuration of certain actors and associated activities. We have first introduced

[4] It should be noted that agent-orientation has begun to make efforts to overcome these drawbacks. Nowadays, there exist approaches that view collectivities of agents as first class abstractions that can collectively act on their own (e.g. holons [FSS03], JACKTeams [AG09], socially-constructed agents [BvdT05], platform agents [Röl04], organizations as specialized agents [HMMF08]).

a reference architecture for MOS in [WEMRM07] and in [WEM08] (top left of Figure 2). The proposal is quite abstract, being more of a coarse characterization. In [WEM09], we have supplied a precise technical model of how software systems based on the collective actor idea can be operationalized (bottom left of Figure 2). It is based on the high-level Petri net formalism of reference nets [Kum02] that fuses classical Petri nets semantics with the possibility of nesting Petri nets inside each other (following the initial nets-in-nets approach in [Val03]) and have them communicate via synchronous channels. In addition, reference nets have excellent tool support by means of the RENEW [KWD09] tool which made our operational models directly executable. However, between the abstract concept

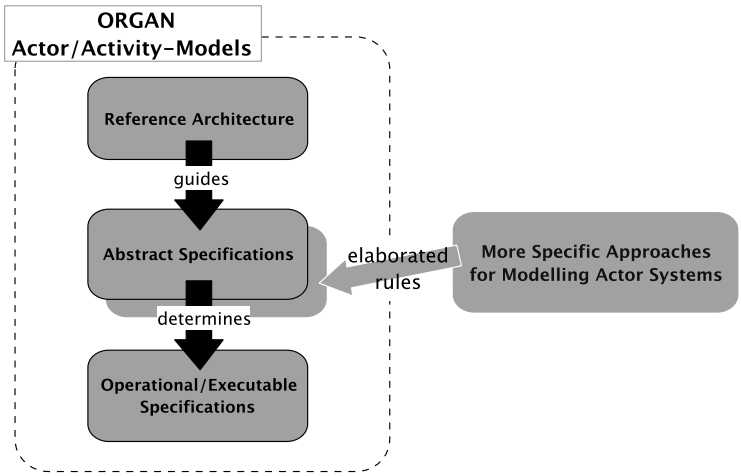


Fig. 2: ORGAN actor/activity-models

of a MOS and the fine-grained technical mechanisms to operationalize collective actor systems, we have to date not really elaborated on how to actually specify/model specific instances of these kinds of systems. In this paper we close this gap. We introduce a modeling approach that is based on the precise operational semantics of Petri net but that also offers various abstraction mechanisms to hide arbitrarily many details (middle left of Figure 2). Consequently, this approach can be used to smoothly transcend from very detailed to very abstract models of a system of systems and vice versa. Note that this approach is not *necessarily* tied to our MOS reference architecture (this accounts for the operational/executable models as well). It features generic principles to model systems of systems based on the universal concepts of actors and activities. The modeled collectivities need not fit into the categories of the MOS reference architecture. Consequently, the approach presented here can also be regarded independently from our wider research context.

Finally, the figure shows a relation to approaches external to ORGAN. The generic and universal character of our approach allows (and in many cases demands) to be supplemented by more specific approaches. Our approach alone basically allows to model which kinds of actors take part in which kinds of activities. But existing approaches to model collectivities of agents (e.g. teams, groups, organizations, institutions, c.f. [BHS07]) feature more elaborated means for describing structural and behavioral configurations. These

are of course specific to their respective approach. Consequently, such approaches can be consulted selectively to enrich (parts of) our models with additional rules. We have shown in [WEKBM08] the value of considering different existing modeling approaches from the field of multi-agent systems with respect to addressing different system levels in a MOS.

2 Actors and Activities

The idea is to develop a universal model of a system unit. We follow Bertalanffy's [Ber56] classical definition of a system *"A system is a set of interacting units with relationships among them"*. We can identify the general notions of structure and behavior of a system. In our case we choose to conceptualize internal system parts and their interactions as actors and activities in which the actors participate. They will be considered in this section. The topic of additional relationships between actors will be covered in the subsequent sections.

We use the term activity in accordance with the UML [Boc03]. An activity models some part of system's behavior. It does so by combining elementary behavioral elements (i.e. actions) into more complex specifications of behavior by means of control and data flows. This of course leaves much room for specialization. In the following we will look at the specific understanding of activities for our system unit model.

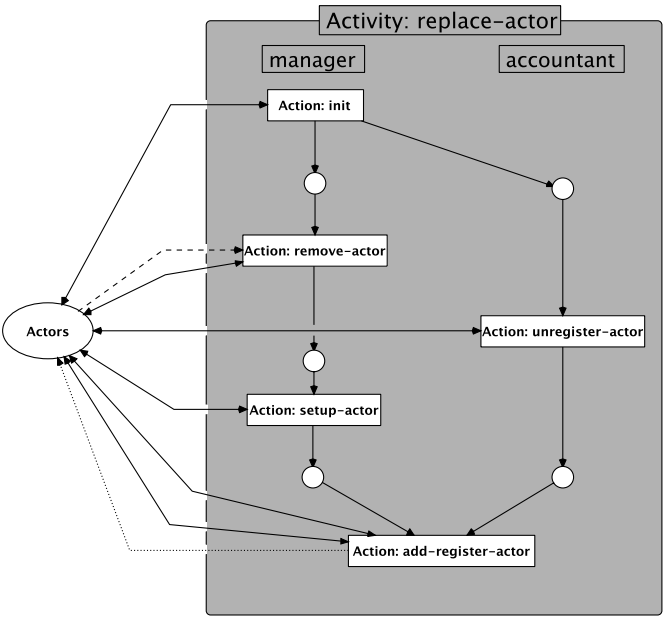


Fig. 3: A system unit net with one activity scheme

Figure 3 shows a fragment of a specific system unit as a Petri net. There is a place to the left of the figure where all the actors of the system unit reside. In addition, it contains a subnet to the right that embodies an activity specification that the actors can utilize. The actions that the activity is composed of are modeled by transitions. An action is always

performed by an actor or synchronously by multiple actors. The participation of an actor in an action execution can have three possible effects on the actor:

1. **use/modify** *(modeled by an arc with double arrow tips):* The actor contributes to the execution of an action. So the actor is just „used“ to accomplish the action. In addition, the actor might also undergo changes because of its participation.
2. **add** *(modeled by a dotted arc with the arrow tip at the actor place):* In the course of the action, the actor is added to the system unit. In a closed system unit, this concerns only the creation and addition of an actor by another actor who is already part of the system. In the case of an open system unit (c.f. Section 4), it may also be the case that new actors enter the system from the outside.
3. **remove** *(modeled by a dashed edge with the arrow tip at the action transition):* The actor is removed from the system unit. This could just be the actor itself leaving. Or it could be the forced removal of the actor by another one who has the required authority. In the case of an open system unit, removal might also mean migration to other system units outside of the current one.

The activity subnet showed here follows similar conventions like AUML interaction diagrams (c.f. [CMR03]). This means that vertically aligned transitions belong to an *activity role* and are to be executed by the same actor for one activity instance. The name of the role is displayed at the top of this so called *life line* of the role. In addition to life lines, places that connect transitions horizontally model a message exchange between the actors that occupy the respective roles. In particular, the activity shown here models the replacement of an actor of the system unit by another actor. the activity includes roles for a manager and an accountant. The actor playing the manager role tells the actor playing the accountant tole to un-register an existing actor. Afterwards, the manager removes this actor from the system and sets up a new one. Finally, manager and accountant synchronize for a joint action to add and register the new actor simultaneously.

It should be emphasized that the activity subnet shown here does not model *one* specific occurrence of an activity. It rather models a scheme for or a pattern of an activity. It can be used in multiple instances. However, if the model should not just be used for illustrative purposes but also to be directly executed (e.g. in the context of a Petri net tool like RENEW), of course additional technical details are necessary, especially if the activity subnet should be usable in multiple instances at the same time. We will not deepen on this topic in this paper (but see the conclusion).

As one can imagine, if one wants to model a complete system unit with a possibly high number of activities, the detailed approach from the last figure becomes unfeasible. Thus we present our first mechanism of abstraction in Figure 4. The resulting model shows the activities that can take place and the corresponding roles that are part of the activities and need to be played by actors. The edges give information about whether the actor that plays the role will be used, added or removed during the activity. The model still has a Petri net semantics , it is just that all activities are folded into one transition and appear as atomic. Behavior specifications in terms of control and data flow are now hidden.

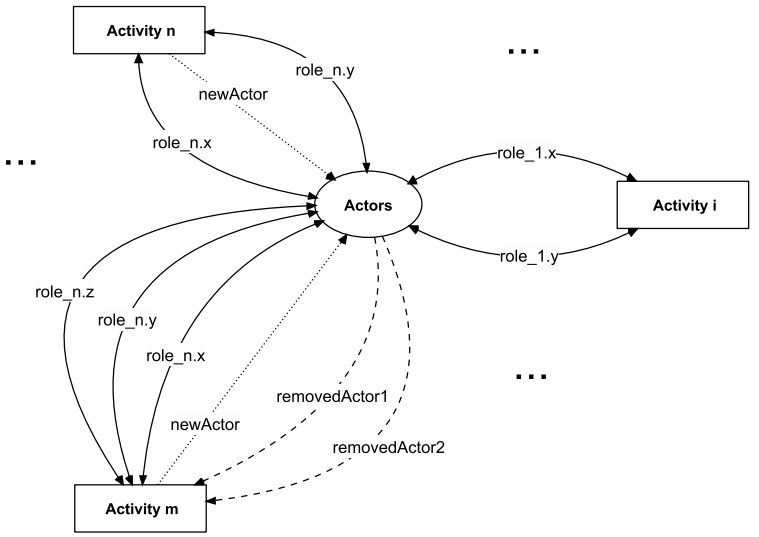


Fig. 4: System unit net with abstract activity specifications

3 Actor and Activity Sets

The abstraction level from model from Figure 4 still requires to consider each individual activity specification. It may be desirable to model actor-activity relationships on a still higher level of abstraction. For this purpose, this section introduces to model *sets* of actors and activities. This also allows to regard subsets and super-sets (set unions) depending on the desired level of abstraction. Just as different hierarchy levels of actor and activity set inclusion are regarded, we obtain hierarchical levels of system abstraction.

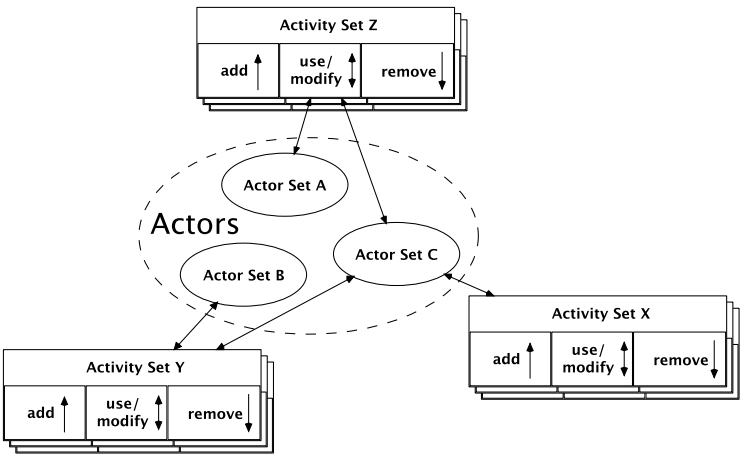


Fig. 5: Actor and activity sets

Figure 5 gives a first illustration. Now, actors are no longer associated with activities via specific effects (usage, addition, removal). Instead, it is just stated, which actor sets are associated with which activity sets. Consequently, the model does no longer exhibit Petri net semantics. However, Petri net semantics may be re-obtained by refining the model in terms of the two abstraction levels presented in the former section.

Despite losing specific information concerning activity execution, the model from Figure 5 does also introduce information that was not present beforehand. By „coloring“ the former uniform set of actors and relating the resulting distinguishable actor subsets with different activity sets, we can begin to make stronger statements concerning actor relationships. For example, we see that the actor sets A and B both are directly related to actor set C as they are engaged in the same activity sets respectively while A and B are not directly related to each other.

However, in order so transcend between abstraction levels of a system, it is not feasible to always be obliged to model *all* identifiable actor and activity sets exhaustively at a time. Consequently, we introduce shorthand notations for modeling relationships between actor and activity sets. These shorthands are based on the introduction of set unions on the other hand and the identification of subsets as the counterpart.

First of all, Figure 6 introduces the modeling of *contingent* activity participation of actors in activities. This shorthand notation models the circumstance that an actor set only partic-

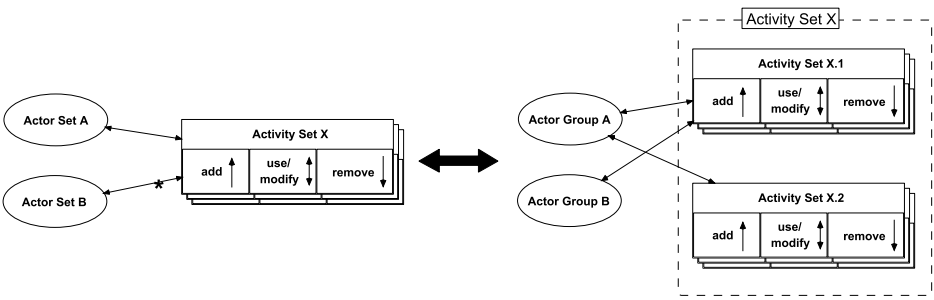


Fig. 6: Shorthand notation (on the left side): Contingent activity participation

ipates in *some* activities of an activity set. Consequently, resolving the shorthand notation, we obtain two subsets of the earlier activity set, one subset where the actor set is required and one subset where the actor set is not associated. The actor set A is only included in Figure 6 for illustrative purposes. As it is associated to the original activity set via an ordinary participation arc, it consequently is required for both identified activity subsets.

The next step is to directly abstract away from some particular actor sets. For this purpose, our model allows to build the union of these actor sets and just consider the newly obtained super-set as a holistic actor set. In the opposite direction, the modeler might see the necessity to identify subsets for an actor set that was considered as one single actor set before. Building actor set unions or identifying actor subsets always goes hand in hand with considerations concerning associated operations on the activity set side and considerations concerning activity participation on the different levels of abstraction.

Figure 7 exemplifies the shorthand notation of building actor set unions and how to resolve these shorthands when subsets are (re-)identified. In Figure 7 (a), resolving the actor set

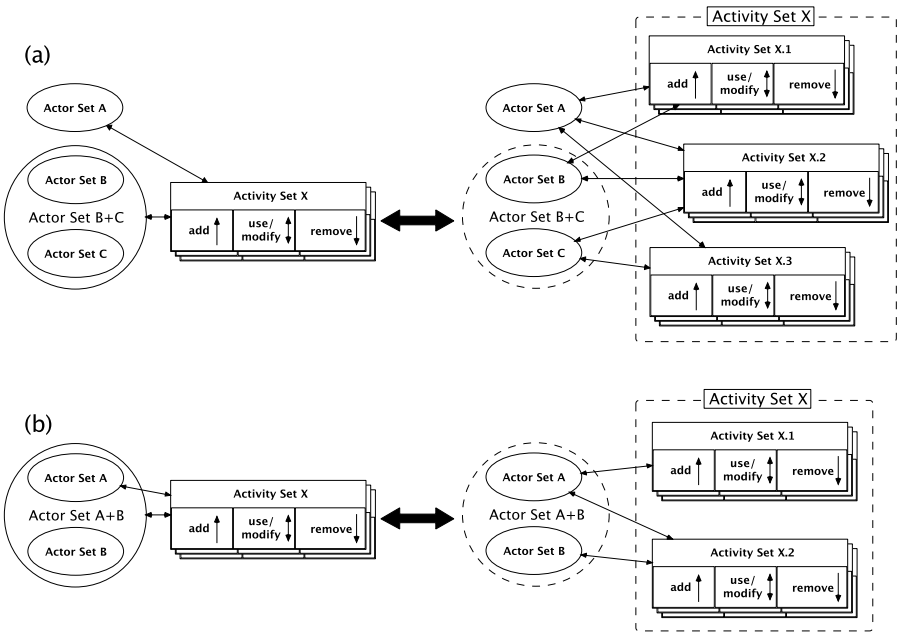


Fig. 7: Shorthand notation (on the left side): Hierarchical actor sets

B+C to the subsets B and C introduces three *potential* activity subsets for the former activity set X. However, not all of the subsets actually have to exist. Some might just be empty. Figure 7 (b) illustrates the case where activity participation of an actor set and one of its subset are mixed.

As actors and activities are dual concepts, the source of changing the level of abstraction might also be activity set union or the identification of activity subsets. Figure 8 illustrates this as a counterpart to the former figures. All in all, no matter whether set unions or

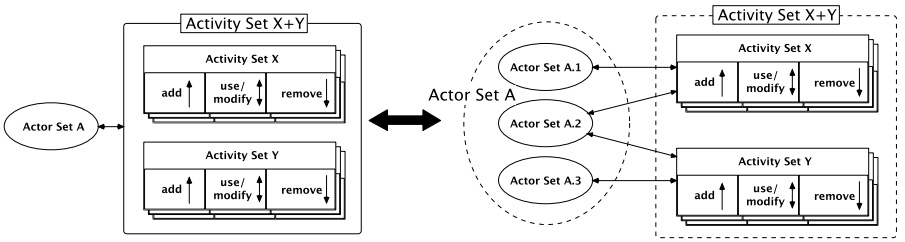


Fig. 8: Shorthand notation (on the left side): Hierarchical activity sets

identification of subsets are initiated from the actor or the activity side, they always lead to according considerations on the opposite side.

In the notations we have introduced so far the duality between actors and activities has been preserved. Both can be refined and coarsened in similar ways by introducing subsets and super-sets. These abstraction mechanisms enable us for example to design a system unit in a top-down manner by starting with the most abstract view of just differentiating between very coarse actors and activities and refining them until we arrive at Petri net models. As a counterpart, it is also possible to design a system in a bottom-up manner by identifying specific activities and actors in the beginning, modeling these and abstracting away from them to appropriate sets and super-sets as one discovers overall relations.

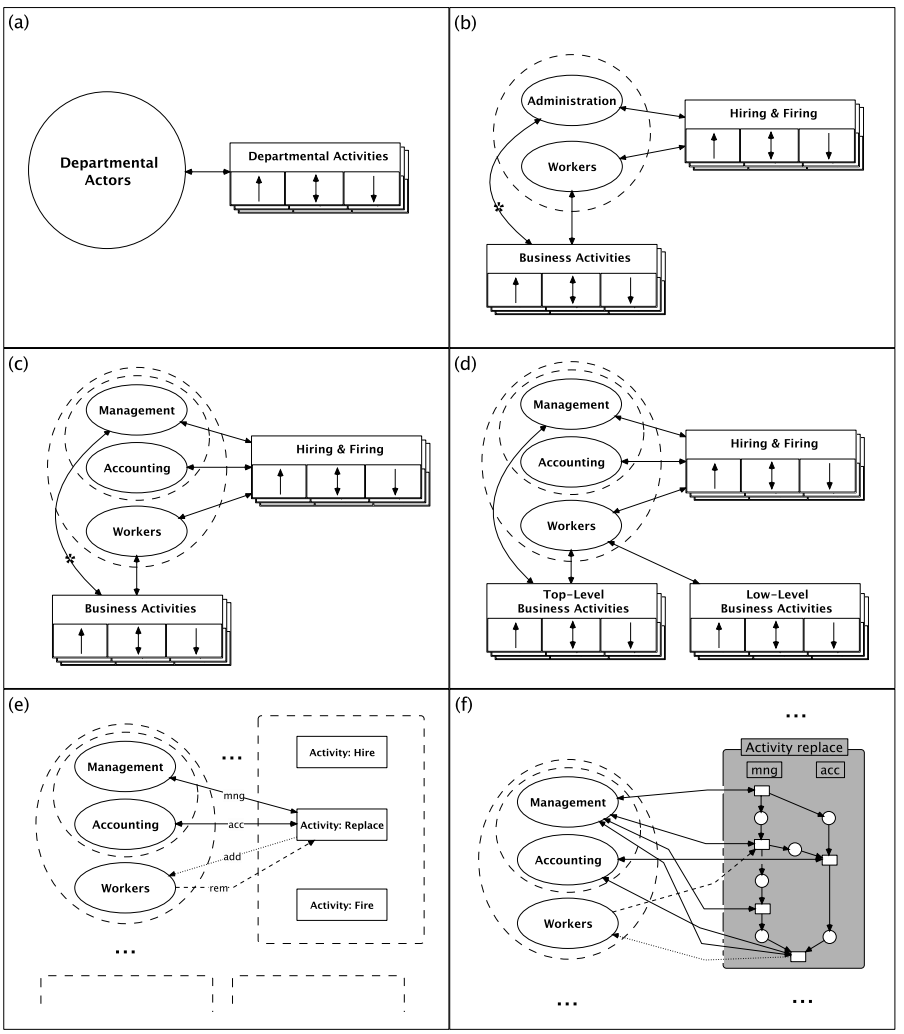


Fig. 9: Exemplary top-down design of a particular system unit.

Even more important, our model allows to transcend back and forth between levels of abstraction and especially to focus on a particular part of a system, elaborate on it and leave

its context on an abstract level. As an illustration of combining all the abstraction mechanisms introduced thus far, Figure 9 shows an exemplary top-down design of a system unit.

The model we have considered so far fulfills the requirements set by Bertalanffy's definition of a system. It allows us to model structure as well as behavior of a system. We allow for a smooth modulation of abstraction for both of these. But so far, we have only regarded *one* system unit. In the next section, we advance to regard systems of systems.

4 Collective Actors

Our claim is that the ideal building block for systems of systems is based on the philosophy of having system units that act both as environments for actors and as actors themselves. Until now, we have talked about actors mainly in terms of actor sets. Even on the more

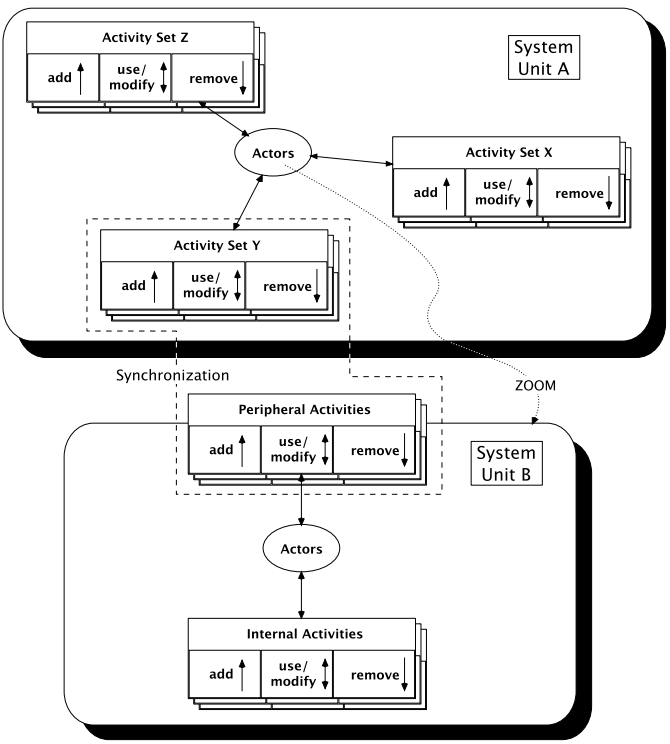


Fig. 10: Collective agency via peripheral activities

detailed level of Section 2 we have said nothing more than that actors have to be able to occupy activity roles in order to interact with other actors. Now, following our initial proposal, the actors of a system unit can again be system units as collective actors. A system unit's collective actions are actually carried out by its internal actors who act *on behalf of* or *in the name of* the system unit as a whole. Following the approach taken so

far, actions are only carried out in the course of activities. Consequently, a system unit that acts as a whole has to offer activities that include actions that take effect not only in the context of the system unit itself but also in surrounding system units and thus at higher system levels. This idea is modeled in Figure 10. System unit B acts in the context of system unit A because it utilizes peripheral activities to allow its internal actors to carry out actions that appear as actions of B itself at the level of A.

We want to keep up with our emphasis on Petri net semantics underlying all of our models and elaborate on the Petri net semantics of peripheral activities in Figure 11. We see that

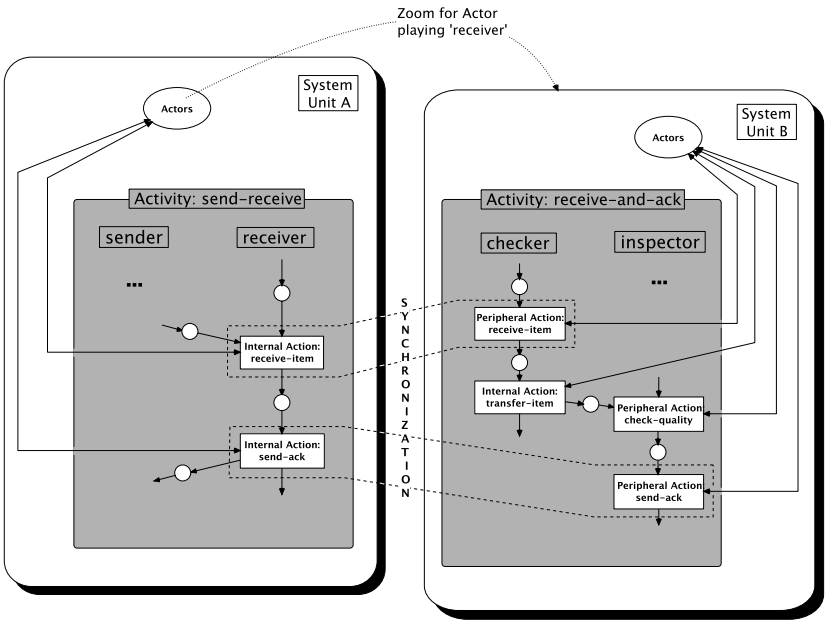


Fig. 11: Petri net semantics of peripheral activities

a peripheral activity includes peripheral actions. These are exactly the actions that have effects on multiple system levels. It is important to note that for a peripheral activity not all actions have to be peripheral. In Figure 11 there are system unit A and system unit B. System unit B is an actor embedded by system unit A. System unit A offers the activity send-receive. In order to participate in this activity, B has to implement a role of the activity. In this case, it is the receiver role. At the level of A, it simply appears that B carries out the actions required by the role. But from the perspective of B itself, this is accomplished via the receive-and-ack activity. This activity features two activity roles itself and the peripheral actions required to play the receiver role are distributed over both roles. In addition, some internal actions are necessary. Concerning Petri net semantics, transitions for peripheral actions have to be synchronized with transitions for actions at the level of the embedding system unit as it is illustrated in the figure. As mentioned earlier in this article, the high level Petri net formalism of reference nets exactly offers both the possibilities of net nesting and transition synchronization between horizontally adjacent nets. It is no problem to obtain arbitrarily deep nesting of actors and arbitrarily long synchronization

chains between actions of multiple system units. In the example, it might for example be possible that the actor playing the role checker in the context of B is again a system unit as a collective actor. Consequently, its actions to implement the checker role would again be peripheral actions that would have to be carried out by *its* embedded actors.

Basically, we arrive at an operational understanding of collective agency. In the example, we have two actors that act on behalf of a system unit and together implement an activity role that the system unit as whole plays at an even higher system level. This understanding of collective agency finally allows us to model systems of systems based on nested actor systems and Figure 12 exemplifies an extract of an overall system that evolves around one specific system unit A. In the figure, different synchronization lines are illustrated,

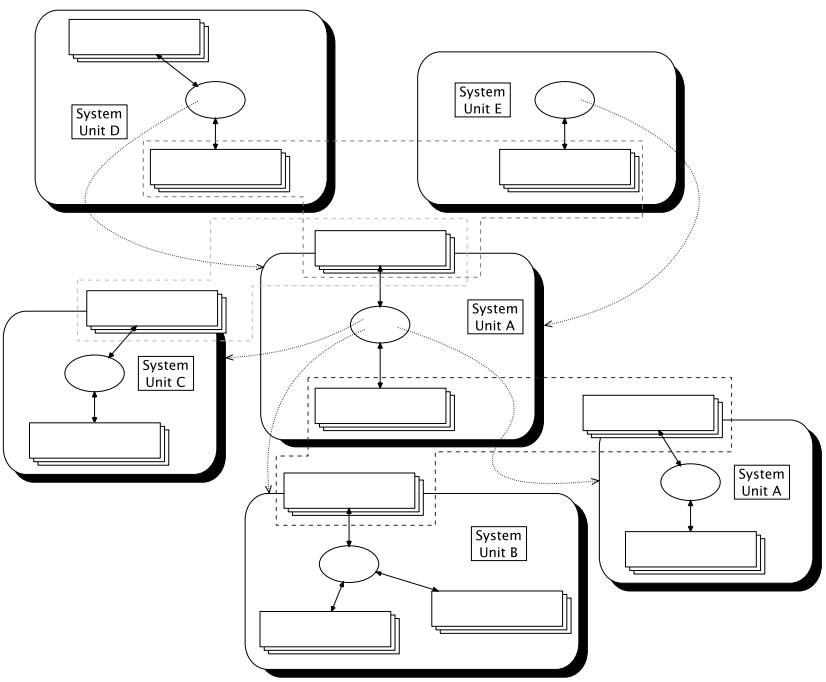


Fig. 12: Simple system of systems setting based on system units

showing different cases of activity layering for the sake of collective agency. The figure also illustrates that actor nesting needs not be unique and disjoint. It is possible that actor A is both embedded in D and E. Consequently, there exists a connection between D and E that these two might not even be aware of.

5 Conclusion and Outlook

In this paper, we have presented a modular approach to comprehend systems of systems by means of composing system units. Each system unit may be regarded under a platform perspective, where it offers environment frames for its inhabitants. Furthermore the same

unit may be regarded under a collective agency perspective, where it collectively acts as a holistic entity in the context of a higher-level system unit. The model is based on Petri net semantics which gives a clear view of how to operationalize it. However, we have also introduced a variety of abstraction mechanisms that allow to smoothly transition from low-level to high-level system views and back.

In the introduction, we have described how the work presented in this paper fits into the wider context of our research. Figure 2 from the introduction illustrates how the modeling approach presented in this paper fits between our abstract proposal of a reference architecture for MOS and our fully executable operational models. Here, we want to be a bit more specific concerning this fit by being more specific concerning the architecture and the operational/executable model. This illustrates the gap that we closed with the work in this paper. For the reference architecture, we have qualitatively distinguished departments, organizations, organizational fields and the society as iteratively embedded specific types of system units. A coarse overview of the proposal is given in Figure 13.

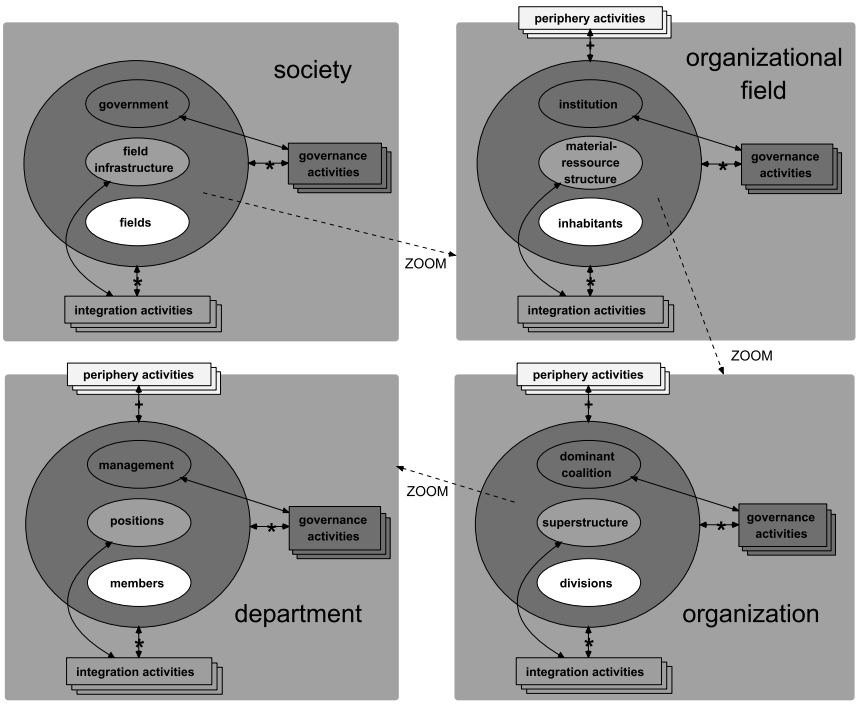


Fig. 13: Architectural proposal for multi-organization systems

In the opposite direction, we have based our proposal for operationalizing collective actor systems on reference net semantics. Its most basic idea is shown in Figure 14. Here, system activities rest on a specific place and for action execution, a synchronization between actor and activity nets has to be carried out. Internal actions are distinguished from peripheral actions. For internal actions, a system unit calls an internal actor via an :act()-channel in

order to be synchronized with an activity (called via an :iStep()-channel). In the peripheral action case, things are similar but with the addition that now the system unit itself is also called via its own :act() channel by its surrounding system unit (and now the activity is called via a :pStep()-channel). The figure exemplifies one particular case where an action

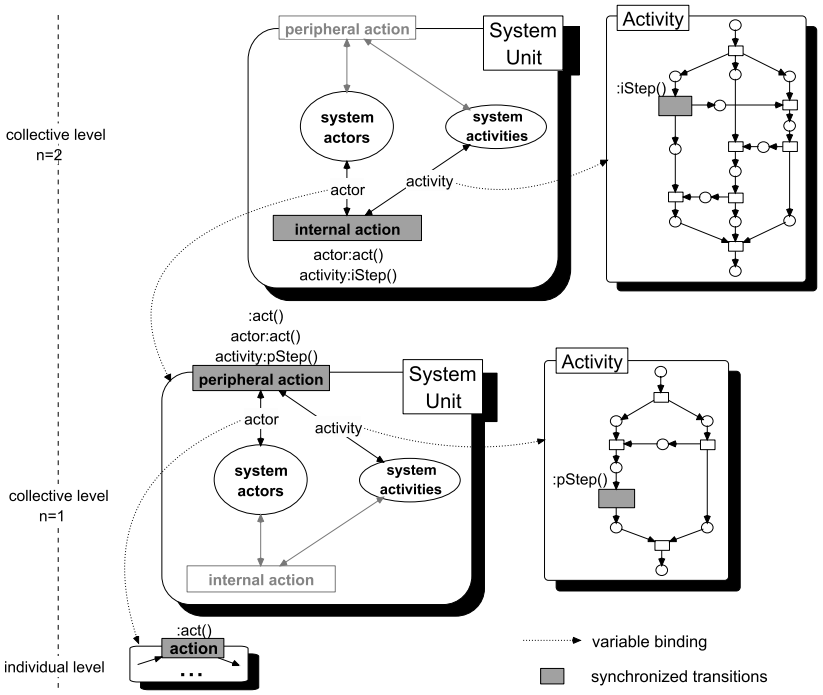


Fig. 14: Operationalization of system units: An action occurrence

occurs that is an internal action for one system unit and a peripheral action for its embedded system unit. However, the figure still lacks a considerable amount of necessary operational details.

References

[AG09] AOS-Group. Jack Intelligent Agents Team Manual. Available at: http://www.aosgrp.com/documentation/jack/JACK_Teams_Manual_WEB/index.html, 2009.

[Ber56] Ludwig von Bertalanffy. General System Theory. In Ludwig von Bertalanffy and Anatol Rapoport, editors, *General Systems: Yearbook of the Society for the Advancement of General Systems Theory*, volume 1, pages 1–10. Ann Arbor, MI: The Society, 1956.

[BHS07] Olivier Boissier, Jomi Hübner, and Jaime Simao Sichman. Organization Oriented Programming: From Closed to Open Organizations. In G. O'Hare, A. Ricci, M. O'Grady, and O. Dikenelli, editors, *Engineering Societies in the Agents World VII*, volume 4457 of *LNCS*, pages 86–105. Springer, Heidelberg, 2007.

[Boc03] Conrad Bock. UML 2 Activity and Action Models. *Journal of Object Technology*, 2(5):43–53, 2003.

[BvdT05] Guido Boella and Leendert van der Torre. Organizations as Socially-Constructed Agents in the Agent-Oriented Paradigm. In Marie Pierre Gleizes, Andrea Omicini, and Franco Zambonelli, editors, *Engineering Societies in the Agents World V*, volume 3451 of *Lecture Notes in Computer Science*, pages 1–13. Springer Verlag, 2005.

[CMR03] Lawrence Cabac, Daniel Moldt, and Heiko Rölke. A Proposal for Structuring Petri net-based Agent Interaction Protocols. In Wil M. P. van der Aalst and Eike Best, editors, *Application and Theory of Petri Nets 2003*, volume 2679 of *Lecture Notes in Computer Science*, pages 102–120. Springer Verlag, 2003.

[FSS03] Klaus Fischer, Michael Schillo, and Jörg Siekmann. Holonic Multiagent Systems: A Foundation for the Organization of Multiagent Systems. In *Holonic and Multi-Agent Systems for Manufacturing, First International Conference on Industrial Applications of Holonic and Multi-Agent Systems (HoloMAS)*, volume 2744 of *Lecture Notes in Computer Science*, pages 71–80. Springer Verlag, 2003.

[HHVE07] Andreas Hess, Bernhard Humm, Markus Voss, and Gregor Engels. Structuring Software Cities - A Multidimensional Approach. In *Proceedings of the 11th IEEE International Enterprise Distributed Object Computing Conference (EDOC 2007)*, pages 122–129, 2007.

[HMMF08] Christian Hahn, Cristian Madrigal-Mora, and Klaus Fischer. A Platform-Independent Metamodel for Multiagent Systems. *Autonomous Agents and Multi-Agent Systems*, 18(2), 2008.

[Jen00] Nicholas Jennings. On agent-based software engineering. *Artificial Intelligence*, 177(2):277–296, 2000.

[Kum02] Olaf Kummer. *Referenznetze*. Logos Verlag, Berlin, 2002.

[KWD09] Olaf Kummer, Frank Wienberg, and Michael Duvigneau. *Renew – User Guide*. Universit"at Hamburg, Fachbereich Informatik, Theoretische Grundlagen der Informatik, Hamburg, release 2.2 edition, 2009. Verf"ugbar "uber die Internetseite http://www.renew.de.

[LMW05] Josef Lankes, Florian Matthes, and Andre Wittenburg. Softwarekartographie: Systematische Darstellung von Anwendungslandschaften. *Wirtschaftsinformatik 2005*, 2005.

[Mai99] Mark Maier. Architecturing Principles for Systems-of-Systems. *Systems Engineering*, 1(4):267–284, 1999.

[Nor06] Linda Northrop. *Ultra-Large-Scale Systems: The Software Challenge of the Future*. Software Engineering Institute, Carnegie Mellon, 2006.

[Röl04] Heiko Rölke. *Modellierung von Agenten und Multiagentensystemen*. Logos Verlag Berlin, 2004.

[Sco03] W. Richard Scott. *Organizations: Rational, Natural and Open Systems*. Prentice Hall, 2003.

[Val03] Rüdiger Valk. Object Petri Nets: Using the Nets-within-Nets Paradigm. In Jörg Desel, Wolfgang Reisig, and Grzegorz Rozenberg, editors, *Advanced Course on Petri Nets 2003*, volume 3098 of *Lecture Notes in Computer Science*, pages 819–848. Springer Verlag, 2003.

[WEKBM08] Matthias Wester-Ebbinghaus, Michael Köhler-Bußmeier, and Daniel Moldt. From Multi-Agent to Multi-Organization Systems: Utilizing Middleware Approaches. In Alexander Artikis, Gauthier Picard, and Laurent Vercouter, editors, *International Workshop Engineering Societies in the Agents World (ESAW 08)*, 2008.

[WEM08] Matthias Wester-Ebbinghaus and Daniel Moldt. Structure in Threes: Modelling Organization-Oriented Software Architectures Built Upon Multi-Agent Systems. In *Proceedings of the 7th International Conference an Autonomous Agents and Multi-Agent Systems (AAMAS'2008)*, pages 1307–1311, 2008.

[WEM09] Matthias Wester-Ebbinghaus and Daniel Moldt. Modeling an Open and Controlled System Unit as a Modular Component of Systems of Systems. In Michael Köhler-Bußmeier, Daniel Moldt, and Olivier Boissier, editors, *Proceedings of the International Workshop on Organizational Modelling (OrgMod 2009), University of Paris*, pages 81–100, 2009.

[WEMRM07] Matthias Wester-Ebbinghaus, Daniel Moldt, Christine Reese, and Kolja Markwardt. Towards Organization–Oriented Software Engineering. In Heinz Züllighoven, editor, *Software Engineering Konferenz 2007 in Hamburg: SE'07 Proceedings*, volume 105 of *LNI*, pages 205–217. GI, 2007.

Modellierung von dynamischen Zielen in Agentensystemen mit Ziel/Transitions-Netzen

Dennis Chong[1]

Abstract: Die geeignete Darstellung von Zielen ist ein wichtiger Aspekt bei der Modellierung von Agentensystemen. Ziele stellen mentale Zustände eines Agenten dar und können sich zur Laufzeit ändern (dynamische Ziele). Sie sind deshalb als dynamische Elemente auf der Modellebene abzubilden. Für die Modellierung von Agentensystemen mit dynamischen Zielen wird in dieser Arbeit die Klasse der Ziel/Transitions-Netze vorgeschlagen. Der Ansatz beruht auf Prädikat/Transitions-Netzen, einer verbreiteten höheren Petri-Netz-Art.

1 Motivation

Um dem zunehmenden Trend von autonomen Software-Systemen mit verteilten Daten und Lösungen zu begegnen, hat sich in der Softwaretechnik das Agentenkonzept etabliert. Agenten repräsentieren in sich geschlossene Software-Entitäten, die über einen längeren Zeitraum in einer Umgebung persistent sind und die Fähigkeit zu autonomen Aktionen und Reaktionen haben (vgl. [Woo02]). Das Verhalten eines Agenten ist dabei an seine Ziele gekoppelt. Ziele stellen mentale Zustände eines Agenten dar und beschreiben die Beschaffenheit der Umgebung, die er mit seinen Handlungen zu erreichen versucht. Sie können zur Laufzeit Änderungen unterliegen, weil Agenten in der Lage sind, auf Grundlage ihrer Erfahrungen selbstständig Ziele zu definieren, zu verwerfen und zu modifizieren (vgl. [KR91]). Dies kann geschehen, wenn ein Ziel erreicht wurde oder weil es aus anderen Gründen nicht mehr sinnvoll erscheint, es in der ursprünglichen Form beizubehalten. In diesem Zusammenhang bedeutet Dynamik, dass Ziele einerseits das Verhalten des Systems beeinflussen und anderseits vom Verhalten des Systems beeinflusst werden.

Ein verbreiteter Ansatz zur Modellierung von Agenten ist die *Agent UML* [BMO00]. Die Agent UML ist eine Weiterentwicklung der Unified Modeling Language (UML), um die besonderen Aspekte von Agenten zu berücksichtigen. Sie besitzt jedoch keine formale Semantik und kann nicht dazu verwendet werden, formale Aussagen über das Modell-Verhalten zu machen, weil die Systemabläufe nicht eindeutig festgelegt sind. Für die *formale* Spezifikation von Agenten kommen deshalb überwiegend modal- und temporallogische Ansätze (implizit oder explizit) zum Einsatz. Beispiele sind das BDI-Modell nach Rao und Georgeff [RG95], die intentionale Theorie nach Cohen und Levesque [CL90] und die agentenorientierte Programmiersprache (AOP) nach Shoham [Sho93]. Im Gegensatz

[1] Institute of Artificial Intelligence Methods and Information Mining (AIM), Hochschule Bremerhaven, Fachbereich 2 - Informatik, dchong@hs-bremerhaven.de

zur Agent UML besitzen diese Modelle aber keine graphische Repräsentation. Die Verfügbarkeit einer graphischen Repräsentation kann jedoch in der Praxis ein entscheidendes Kriterium für ein Modell sein, wie der Erfolg der UML zeigt. Formale Ansätze zur Modellierung von Agenten, die gleichzeitig über eine graphische Repräsentation verfügen, basieren in der Regel auf Graphtransformationen oder Petri-Netzen. Beispiele sind [DHK01] und [HKHK$^+$07] für Graphtransformationen und [XVIY03] und [MW97] für (höhere) Petri-Netze. In den entsprechenden Ansätzen ist es auf unterschiedliche Weise möglich, Ziele eines Agenten im Modell abzubilden. Die Dynamik von Zielen wird allerdings nicht unterstützt. Aus diesem Grund soll in dieser Arbeit ein neuer, formaler Ansatz vorgeschlagen werden, der über eine graphische Repräsentation verfügt und gleichzeitig dynamische Ziele eines Agenten explizit berücksichtigen kann.

Höhere Petri-Netze besitzen zahlreiche Vorteile, weshalb sie in dieser Arbeit als Ausgangspunkt ausgewählt wurden. Zu diesen Vorteilen zählen: die formale Syntax und Semantik, die graphische Repräsentation, die strukturierte und kompakte Darstellung, die Verbreitung in der Literatur, die Berücksichtigung von Nebenläufigkeit und Nichtdeterminismus und die Ausführbarkeit (Simulation) von Petri-Netzen. In den folgenden Abschnitten soll mit der Einführung von Ziel/Transitions-Netzen die Klasse der Prädikat/Transitions-Netze [Gen87] um die Fähigkeit zur Darstellung von dynamischen Zielen erweitert werden. Zu diesem Zweck wird der weitere Verlauf dieser Arbeit wie folgt gegliedert: In Abschnitt 2 werden die Grundideen von Ziel/Transitions-Netzen informell eingeführt. Die algebraischen Grundlagen werden in Abschnitt 3 behandelt. In Abschnitt 4 wird die Klasse der Ziel/Transitions-Netze formal definiert. Abschnitt 5 diskutiert einige Aspekte zur Mächtigkeit und in Abschnitt 6 wird ein Beispiel für einen Agenten mit dynamischen Zielen präsentiert. Den Abschluss dieser Arbeit bildet der Ausblick in Abschnitt 7.

2 Informelle Einführung

In Prädikat/Transitions-Netzen werden die Netz-Elemente mit algebraischen Ausdrücken beschriftet. Auf diese Weise kann die Zahl der notwendigen Stellen und Transitionen in einem Netz begrenzt werden. Gleichzeitig existiert ein wichtiger Zusammenhang zwischen Prädikat/Transitions-Netzen und logischen Formeln. Die Stellen eines Netzes können als *variable Prädikate* interpretiert werden, so dass sich mit Prädikat/Transitions-Netzen prädikatenlogische Formeln repräsentieren lassen (vgl. [GL81]). Die Extension eines variablen Prädikates ist dabei eindeutig durch die Markierung der entsprechenden Stelle festgelegt. Das Prädikat ist bei einer Markierung genau dann erfüllt, wenn die Stelle eine Marke enthält, die die Argumente des Prädikates in Form eines Tupels enthält. Umgekehrt lassen sich mit logischen Formeln, die variable Prädikate enthalten, auch Markierungen eines Netzes beschreiben. Die Idee besteht nun darin, den Zusammenhang von Stellen und Prädikaten bzw. Markierungen und Formeln auszunutzen, um Aussagen über Zustände des Modells als dynamische Elemente auf der Modell-Ebene abzubilden. Voraussetzung ist ein gesonderter Typ von Stellen, der nicht mit Tupeln, sondern mit Formeln markiert wird.

Abbildung 1 zeigt ein einfaches Beispiel für ein Ziel/Transitions-Netz, das die Verwendung von variablen Prädikaten verdeutlichen soll. In der Anfangsmarkierung sei die Stelle

A mit dem Tupel (0) markiert. Mit dem Schalten der Transition t_1 wird die entsprechende Marke entfernt und eine neue Marke auf A erzeugt. Das hier dargestellte Netz setzt keine Bindung der Variablen im Nachbereich einer Transition durch den Vorbereich voraus, so dass beim Schalten von t_1 die Variable y mit einer beliebigen natürlichen Zahl belegt werden kann. Die durch einen doppelten Rand hervorgehobene Stelle Z kann mit Formeln, d.h. mit Zielen, markiert werden. Für diese Stelle sei die Anfangsmarkierung durch die Formel $A(1)$ gegeben, d.h. das Ziel besteht zunächst darin, das Tupel (1) auf der Stelle A zu erzeugen. A stellt dabei ein variables Prädikat dar, das mit der Stelle A assoziiert wird. Die Transition t_1 ist durch eine Schleife mit der Stelle Z verbunden, die mit der Variable F gewichtet ist. Über die Variable F, die mit beliebigen Formeln belegt werden kann, erhält t_1 Zugriff auf das aktuelle Ziel und verwendet es gleichzeitig im Wächter als Negation. Auf diese Weise kann t_1 nur dann aktiviert werden, wenn das aktuelle Ziel nicht erfüllt ist, d.h. wenn eine entsprechende Marke nicht auf A vorhanden ist. Tritt durch fortlaufendes Schalten schließlich der Fall ein, dass auf A tatsächlich die auf Z geforderte Marke vorhanden ist, bleibt t_1 inaktiv. Stattdessen ist in diesem Fall die Transition t_2 aktiviert, weil nur dann das variable Prädikat $A(z)$ erfüllt ist und die Formel im Wächter von t_2 gilt. Die Aktivierung von t_1 und t_2 hängt also von der Beschaffenheit der Markierung von A ab, obwohl keine direkte Verbindung zwischen t_2 und A existiert. Schaltet t_2, wird das aktuelle Ziel von Z entfernt und ein neues Ziel (der Nachfolger von z) erzeugt. Daraufhin kann t_1 wieder fortlaufend schalten, bis die gewünschte Marke auf A vorliegt.

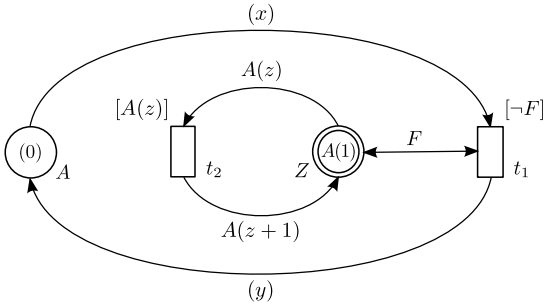


Abb. 1: Ein Ziel/Transitions-Netz mit einer gesonderten Stelle Z für dynamische Ziele.

Die Verwendung von variablen Prädikaten in Ziel/Transitions-Netzen wirkt sich auch auf die Ausdrucksmöglichkeit aus. Ein häufiges Problem bei der Modellierung mit Petri-Netzen ist der Wunsch, die Aktiviertheit einer Transition mit einem so genannten Nulltest zu koppeln. Dabei soll eine Transition nur dann schalten können, wenn eine oder mehrere gegebene Stellen *sauber* sind, d.h. keine Marken enthalten. In beschränkten Netzen kann ein solcher Nulltest durch die Einführung von *Komplementstellen* erfolgen (siehe z.B. [PW03]), aber für unbeschränkte Netze ist dies im Allgemeinen nicht möglich. In Ziel/-Transitions-Netzen kann ein solcher Nulltest sehr einfach formuliert werden. Abbildung 2 zeigt zwei Beispiele.

Die Beispiele zeigen, wie sich Systemzustände in Ziel/Transitions-Netzen mit Formeln beschreiben lassen und wie entsprechende Formeln auf der Modell-Ebene mit dem Modell-Verhalten gekoppelt werden können. Die Formel-Marken können dabei wie gewöhnliche

Marken im Netz konsumiert und produziert werden. Im Folgenden sollen diese Ideen formal definiert werden.

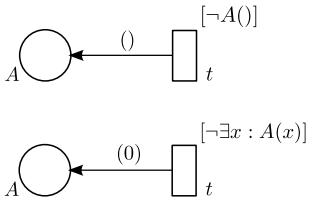


Abb. 2: Oben: Ein Ziel/Transitions-Netz mit einem Nulltest für das 0-stellige Prädikat A. Unten: Nulltest eines 1-stelligen Prädikates. Die Kanten sind für den Nulltest nicht relevant.

3 Algebraische Grundlagen

In diesem Abschnitt werden die algebraischen Grundlagen, die für eine formale Definition von Ziel/Transitions-Netzen notwendig sind, eingeführt.

$\mathbb{N} = \{0,1,2,...\}$ bezeichne die Menge der natürlichen Zahlen und $\mathbb{B} = \{\top, \bot\}$ die Menge der Wahrheitswerte. Mit R^+ sei die transitive Hülle und mit R^* die reflexiv-transitive Hülle einer (zweistelligen) Relation $R : A \times B$ bezeichnet. Ein Wort $a_1...a_n$ bzw. ein Tupel $(a_1,...,a_n)$ über A ist eine endliche geordnete Folge von Elementen aus A, so dass $a_i \in A$ für alle $i \in \{1..n\}$. ε bezeichne das leere Wort, A^* bezeichne die Menge aller Wörter über A und $A^+ = A \setminus \{\varepsilon\}$ die Menge aller Wörter über A ohne das leere Wort. Für eine beliebige Menge I bezeichne $(A_i)_{i\in I}$ eine Familie von Mengen, so dass A_i für alle $i \in I$ eine Menge ist und $A_i \cap A_j = \emptyset$ für alle $i, j \in I$ mit $i \neq j$. Für eine Familie $A = (A_i)_{i\in I}$ bezeichne A ebenso die Vereinigung $\bigcup_{i\in I} A_i$.

Signaturen und Strukturen

In Ziel/Transitions-Netzen werden die Netz-Elemente mit algebraischen Ausdrücken beschriftet. Aus diesem Grund müssen zunächst die notwendigen Begriffe wie Signatur und Struktur formal eingeführt werden, bevor die entsprechenden syntaktischen und semantischen Konzepte definiert werden können:

Eine *Signatur* $\Sigma = (S, \Omega, \Pi)$ besteht aus einer endlichen Menge S von *Sorten*, einer Familie $\Omega = (\Omega_w)_{w\in S^+}$ von *Operationssymbolen* und einer Familie $\Pi = (\Pi_w)_{w\in S^*}$ von *Prädikatensymbolen*. Eine Σ-*Struktur* A_Σ für eine Signatur Σ besteht aus einer Menge A_s für jede Sorte $s \in S$, deren Elemente *Individuen* genannt werden, einer Funktion $f_\omega : A_{s_1} \times ... \times A_{s_n} \to A_{s_{n+1}}$ für jedes Operationssymbol $\omega \in \Omega_{s_1...s_n s_{n+1}}$ und einer Relation $r_\pi : A_{s_1} \times ... \times A_{s_n}$ für jedes Prädikatensymbol $\pi \in \Pi_{s_1...s_n}$. Ein Operationssymbol $\omega \in \Omega_s$ mit $s \in S$ heißt *Konstantensymbol* der Sorte s, die entsprechende Funktion $f_\omega : A_s$ heißt *Konstante*.

Die Signatur ist syntaktischer Natur und legt Bezeichner für Sorten, Operationen und Prädikate fest. Auf ihr können Ausdrücke wie Terme und Formeln erklärt werden. Die Struktur ist semantischer Natur und assoziiert die Bezeichner in der Signatur mit konkreten

Mengen, Funktionen und Relationen. Auf diese Weise können die entsprechenden Ausdrücke eindeutig ausgewertet werden. Aus Gründen der Flexibilität werden bei der Modellierung mit Ziel/Transitions-Netzen grundsätzlich mehrsortige Signaturen bzw. Strukturen berücksichtigt.

Im Folgenden soll davon ausgegangen werden, dass für jedes Individuum $a \in A_s$ einer Sorte $s \in S$ genau ein Konstantensymbol $\omega \in \Omega_s$ mit $f_\omega = a$ existiert, d.h. jedes Individuum kann syntaktisch durch genau ein Konstantensymbol der Signatur und semantisch durch genau eine Konstante der Struktur repräsentiert werden. Darüber hinaus wird für die Beispiele von Ziel/Transitions-Netzen in dieser Arbeit die Signatur und Struktur der natürlichen Zahlen mit den gängigen Operationen vorausgesetzt.

Terme und Formeln

Die syntaktischen Regeln zur Bildung von Ausdrücken in Ziel/Transitions-Netzen sind durch Terme und Formeln gegeben:

Sei $\Sigma = (S, \Omega, \Pi)$ eine Signatur. Dann ist $X = (X_w)_{w \in S}$ eine Familie von *Variablen* für Σ mit einer Variablenmenge X_s für jede Sorte $s \in S$.

1. Die Menge der *Terme* einer Sorte $s \in S$ ist wie folgt gegeben:

   (a) Jede Variable $x \in X_s$ ist ein Term der Sorte s.

   (b) Wenn $\omega \in \Omega_{s_1 \ldots s_n s}$ ein Operationssymbol ist und t_i für alle $i \in \{1..n\}$ ein Term der Sorte s_i ist, dann ist $\omega(t_1, \ldots, t_n)$ ein Term der Sorte s.

2. Die Menge der *Formeln* ist wie folgt gegeben:

   (a) Wenn t_1 und t_2 Terme sind, dann ist $(t_1 = t_2)$ eine Formel.

   (b) Wenn $\pi \in \Pi_{s_1 \ldots s_n}$ ein Prädikatensymbol ist und t_i für alle $i \in \{1..n\}$ ein Term der Sorte s_i ist, dann ist $\pi(t_1, \ldots, t_n)$ eine Formel.

   (c) Wenn F_1 und F_2 Formeln sind und $x \in X$ eine Variable ist, dann sind auch $(\neg F_1)$, $(F_1 \wedge F_2)$, $(F_1 \vee F_2)$, $(F_1 \rightarrow F_2)$, $(F_1 \leftrightarrow F_2)$, $(\exists x : F_1)$ und $(\forall x : F_1)$ Formeln.

Ein Vorkommen einer Variablen x in einer Formel F heißt genau dann *frei*, wenn x in keiner Teilformel von F der Form $\forall x : G$ oder $\exists x : G$ vorkommt. Jedes Vorkommen einer Variablen x in einem Term t heißt *frei*. Ein Ausdruck (Term oder Formel) heißt *geschlossen*, wenn keine freien Variablen darin vorkommen, sonst *offen*.

Es ist möglich, auf Klammern in Ausdrücken zu verzichten, wo sie nicht unbedingt notwendig sind. Darüber hinaus ist neben der Präfix-Notation auch die Infix-Notation zulässig, wenn sie die Lesbarkeit untersützt, z.B. in mathematischen Ausdrücken.

Auswertung und Vereinfachung

Die Interpretation eines Ausdrucks hängt von der gewählten Struktur ab und ist im Folgenden nur für geschlossene Ausdrücke erklärt. Die freien Variablen in offenen Termen und Formeln müssen somit zunächst geeignet substituiert werden, bevor die Ausdrücke ausgewertet werden können:

Sei Σ eine Signatur und $X = (X_w)_{w \in S}$ eine Familie von Variablen für Σ. Eine Abbildung $\alpha = (x_1 \mapsto c_1, ..., x_n \mapsto c_n)$, die Variablen x_i auf Konstantensymbole c_i der selben Sorte abbildet, heißt *Belegung*. Für einen beliebigen Ausdruck e bezeichne $e : (x_1 \mapsto c_1, ..., x_n \mapsto c_n)$ den Ausdruck, der durch die Ersetzung jeder freien Variable x_i in e durch c_i entsteht. Die *Auswertung* A_Σ von Ausdrücken ist eine Abbildung, die Terme auf Individuen und Formeln auf Wahrheitswerte abbildet:

1. Für einen geschlossenen Term $t = \omega(t_1, ..., t_n)$ mit $\omega \in \Omega$ ist $A_\Sigma(t) = f_\omega(A_\Sigma(t_1), ..., A_\Sigma(t_n))$.
2. Für eine geschlossene Formel F ist $A_\Sigma(F)$ wie folgt festgelegt:
   (a) Für $F = (t_1 = t_2)$ ist $A_\Sigma(F) = \top$, wenn $A_\Sigma(t_1) = A_\Sigma(t_2)$, sonst $A_\Sigma(F) = \bot$.
   (b) Für $F = \pi(t_1, ..., t_n)$ mit $\pi \in \Pi$ ist $A_\Sigma(F) = \top$, wenn $(A_\Sigma(t_1), ..., A_\Sigma(t_n)) \in r_\pi$, sonst $A_\Sigma(F) = \bot$.
   (c) Für $F = \neg F_1$ ist $A_\Sigma(F) = \top$, wenn $A_\Sigma(F_1) = \bot$, sonst $A_\Sigma(F) = \bot$.
   (d) Für $F = F_1 \wedge F_2$ ist $A_\Sigma(F) = \top$, wenn $A_\Sigma(F_1) = A_\Sigma(F_2) = \top$, sonst $A_\Sigma(F) = \bot$.
   (e) Für $F = F_1 \vee F_2$ ist $A_\Sigma(F) = \top$, wenn $A_\Sigma(\neg((\neg F_1) \wedge (\neg F_2))) = \top$, sonst $A_\Sigma(F) = \bot$.
   (f) Für $F = F_1 \to F_2$ ist $A_\Sigma(F) = \top$, wenn $A_\Sigma(\neg(F_1 \vee F_2)) = \top$, sonst $A_\Sigma(F) = \bot$.
   (g) Für $F = F_1 \leftrightarrow F_2$ ist $A_\Sigma(F) = \top$, wenn $A_\Sigma(F_1 \to F_2) = A_\Sigma(F_2 \to F_1) = \top$, sonst $A_\Sigma(F) = \bot$.
   (h) Für $F = \exists x : F_1$ mit $x \in X_s$ und $s \in S$ ist $A_\Sigma(F) = \top$, wenn ein Konstantensymbol a der Sorte s existiert, so dass $A_\Sigma(F_1 : (x \mapsto a)) = \top$, sonst $A_\Sigma(F) = \bot$.
   (i) Für $F = \forall x : F_1$ mit $x \in X_s$ und $s \in S$ ist $A_\Sigma(F) = \top$, wenn $A_\Sigma(\neg\exists x : \neg F_1) = \top$, sonst $A_\Sigma(F) = \bot$.

Neben der Auswertung von Ausdrücken wird auch der Begriff der Vereinfachung für Ziel/-Transitions-Netze benötigt. Die Vereinfachung von Ausdrücken bezeichnet die syntaktische Ersetzung von Termen durch semantisch äquivalente Konstantensymbole:

Sei t ein geschlossener Term und F eine geschlossene Formel. Die *Vereinfachung* von t ist gegeben durch $\widehat{A_\Sigma}(t) = c$ für ein Konstantensymbol c mit $A_\Sigma(t) = A_\Sigma(c)$. Mit $\widehat{A_\Sigma}(F)$ sei die Vereinfachung der Formel F bezeichnet, in der alle Terme vereinfacht wurden.

Da vorausgesetzt wird, dass zu jedem Individuum genau ein Konstantensymbol existiert, ist die Vereinfachung von Ausdrücken für alle Terme und Formeln eindeutig.

Multimengen

Von grundlegender Bedeutung für die Definition von Ziel/Transitions-Netzen ist auch der Begriff der Multimenge, der sowohl für Kantengewichte als auch für Markierungen verwendet wird. Multimengen sind besondere Mengen, in denen das mehrfache Vorkommen eines Elementes möglich ist, d.h. jedem Element wird eine Häufigkeit zugeordnet:

Sei A eine beliebige nichtleere Menge. Eine *Multimenge* über A ist eine Abbildung $m : A \to \mathbb{N}$, die jedem Element aus A eine *Häufigkeit* zuordnet. $\mathbb{N}^A$ bezeichne die Menge aller Multimengen über A. Für ein $a \in A$ gilt $a \in m$ genau dann, wenn $m(a) > 0$. m heißt endlich genau dann, wenn $m(a) > 0$ nur für endlich viele $a \in A$ gilt. 0 bezeichnet die leere Multimenge mit $0(a) = 0$ für alle $a \in A$. Für zwei Multimengen $m_1, m_2 \in \mathbb{N}^A$ gilt $m_2 \leq m_1$ genau dann, wenn $m_2(a) \leq m_1(a)$ für alle $a \in A$. Die Addition ist gegeben durch $(m_1 + m_2)(a) = m_1(a) + m_2(a)$ für alle $a \in A$. Für $m_2 \leq m_1$ ist die Subtraktion gegeben durch $(m_1 - m_2)(a) = m_1(a) - m_2(a)$ für alle $a \in A$. Die Multiplikation einer natürlichen Zahl $n \in \mathbb{N}$ mit einer Multimenge $m \in \mathbb{N}^A$ ist gegeben durch $(n \cdot m)(a) = n \cdot m(a)$ für alle $a \in A$.

Für die Schreibweise von Multimengen wird die Mengen- oder die Produktsummennotation verwendet. So ist für die Multimenge $m = (a_1 \mapsto 2, a_2 \mapsto 1, a_3 \mapsto 0, a_4 \mapsto 2)$ über der Menge $A = \{a_1, a_2, a_3, a_4\}$ sowohl $\{a_1, a_1, a_2, a_4, a_4\}$ als auch $2a_1 + a_2 + 2a_4$ zulässig. Eine Funktion f, die auf den Elementen einer Menge A definiert ist, wird bei Bedarf auf Tupel, Mengen und Multimengen über A erweitert, indem f auf jedes Element des Tupels, der Menge oder der Multimenge angewandt wird.

4 Ziel/Transitions-Netze

Bevor die Klasse der Ziel/Transitions-Netze eingeführt werden kann, ist der Begriff der Signatur um eine Familie von so genannten *variablen* Prädikatensymbolen zu erweitern. Im Gegensatz zu den in der Signatur Σ definierten Prädikatensymbolen, deren Extension durch eine Relation in einer Σ-Struktur festgelegt wird, werden variable Prädikatensymbole anhand von Markierungen interpretiert. Die Extension eines variablen Prädikates ist gegenüber der Menge der möglichen Markierungen eines Ziel/Transitions-Netzes also variabel.

Definition 1 (Erweiterte Signatur, Erweiterte Formel) *Sei $\Sigma = (S, \Omega, \Pi)$ eine Signatur und $X = (X_w)_{w \in S}$ eine Variablenmenge für Σ. Die* Erweiterte Signatur *von Σ besteht außerdem aus einer Familie $\widehat{\Pi} = (\widehat{\Pi}_a)_{a \in S^*}$ von* variablen Prädikatensymbolen *mit $\widehat{\Pi} \cap \Pi = \emptyset$*

und einer Variablenmenge $\widehat{X}$ mit $\widehat{X} \cap X = \emptyset$. Eine erweiterte Formel *ist eine Formel, in der variable Prädikatensymbole aus $\widehat{\Pi}$ und Variablen aus $\widehat{X}$ als Teilformeln enthalten sein können. Für ein variables Prädikat $\widehat{\pi} \in \widehat{\Pi}$ und eine erweiterte Formel $\widehat{F}$ gilt $\widehat{\pi} \in \widehat{F}$ genau dann, wenn $\widehat{\pi}$ in $\widehat{F}$ enthalten ist.*

Da eine Variable $\widehat{x} \in \widehat{X}$ nicht durch Quantoren gebunden werden kann, heißt jedes Vorkommen von $\widehat{x}$ in einer erweiterten Formel frei. Darüber hinaus kann $\widehat{x}$ mit erweiterten Formeln substituiert werden. Im Folgenden sei mit dem Begriff Formel stets eine erweiterte Formel gemeint.

Definition 2 (Ziel/Transitions-Netz) *Ein* Ziel/Transitions-Netz $ZT = (N, A_N, M_0)$ *besteht aus folgenden Bestandteilen:*

1. $N = (P_N, T_N, F_N)$ *ist ein* Netz *mit:*

   (a) P_N *ist eine nichtleere, endliche Menge von* Stellen. $\widehat{P}_N \subseteq P_N$ *bezeichnet die Menge der* Ziel-Stellen *und $P_N \setminus \widehat{P}_N$ die Menge der* Prädikat-Stellen.

   (b) T_N *ist eine nichtleere, endliche Menge von* Transitionen, *so dass $T_N \cap P_N = \emptyset$ und $T_N \cup P_N \neq \emptyset$.*

   (c) $F_N \subseteq (P_N \times T_N) \cup (T_N \times P_N)$ *ist eine Menge von* Kanten.

   Der Vorbereich einer Stelle bzw. einer Transition $x \in P_N \cup T_N$ ist durch $F_N x = \{y \mid (y,x) \in F_N\}$, der Nachbereich durch $xF_N = \{y \mid (x,y) \in F_N\}$ gegeben. Für eine Kante $f = (p,t)$ oder $f = (t,p)$ mit $p \in P_N$ und $t \in T_N$ bezeichne $P_N(f) = p$ die Stelle und $T_N(f) = t$ die Transition der verbindenden Kante.

2. $A_N = (A_\Sigma, A_P, A_T, A_F)$ *ist eine* Beschriftung *mit:*

   (a) A_Σ *ist eine Σ-Struktur für eine erweiterte Signatur Σ.*

   (b) A_P *ist eine bijektive Abbildung, die jeder Prädikat-Stelle ein variables Prädikatensymbol zuordnet.*

   (c) A_T *ist eine Abbildung, die jeder Transition eine erweiterte Formel, genannt* Wächter, *zuordnet.*

   (d) A_F *ist eine Abbildung, die jeder Kante f mit $P_N(f) \in P_N \setminus \widehat{P}_N$ eine endliche, nichtleere Multimenge über Tupel von Termen und jeder Kante f mit $P_N(f) \in \widehat{P}_N$ eine endliche, nichtleere Multimenge über Formeln zuordnet. Für eine Kante f mit $P_N(f) \in P_N \setminus \widehat{P}_N$ und ein Tupel $(t_1, ..., t_n) \in A_F(f)$ sind $t_1, ..., t_n$ genau dann Terme der Sorten $s_1, ..., s_n$, wenn $A_P(P_N(f)) \in \Pi_{s_1,...,s_n}$. Für ein $f \notin F_N$ gilt $A_F(f) = 0$.*

3. M_0 *ist eine* (Anfangs-) Markierung, *die jeder Prädikat-Stelle* $p \in P_N \setminus \widehat{P}_N$ *eine endliche Multimenge über Tupel von Individuen und jeder Ziel-Stelle* $\widehat{p} \in \widehat{P}_N$ *eine endliche Multimenge über geschlossenen Formeln zuordnet. Für eine Prädikat-Stelle* $p \in P_N \setminus \widehat{P}_N$ *und ein Tupel* $(d_1, ..., d_n) \in M_0(p)$ *sind* $d_1, ..., d_n$ *genau dann Individuen aus* $A_{s_1}, ..., A_{s_n}$, *wenn* $A_P(p) \in \prod_{s_1,...,s_n}$.

Die Struktur eines Ziel/Transitions-Netzes wird durch ein Netz aus Stellen, Transitionen und Kanten festgelegt. Dabei wird die Menge der Stellen in eine Menge von Ziel-Stellen und in eine Menge von Prädikat-Stellen unterteilt. Während Prädikat-Stellen wie in Prädikat/Transitions-Netzen Tupel von Individuen als Marken enthalten, werden Ziel-Stellen mit geschlossenen Formeln markiert. Jede Prädikat-Stelle wird durch eine Bijektion mit einem variablen Prädikat assoziiert und jedes Tupel, das die Stelle markiert, muss die Stelligkeit des entsprechenden Prädikates respektieren. Kanten, die mit Prädikat-Stellen verbunden sind, werden mit nichtleeren Multimengen über Tupel von Termen gewichtet. Sie berücksichtigen ebenfalls die Stelligkeiten des entsprechenden Prädikates. Kanten, die mit Ziel-Stellen verbunden sind, werden mit nichtleeren Multimengen von Formeln gewichtet. Transitionen können ebenfalls Formeln (Wächter) zugeordnet werden. Sie beschreiben zusätzliche Restriktionen für die Aktiviertheit einer Transitioneninstanz (siehe Definition 3).

Die graphische Repräsentation von Ziel/Transitions-Netzen lehnt sich an die übliche Notation für Petri-Netze an: Stellen werden als Kreise, Transitionen als Rechtecke und Kanten als gerichtete Pfeile gezeichnet. Die Prädikat-Stellen werden mit den variablen Prädikatensymbolen beschriftet, während die Ziel-Stellen zur Unterscheidung durch einen doppelten Rand hervorgehoben werden. Der Wächter einer Transition wird in eckigen Klammern notiert. Wenn er lediglich von einer Tautologie repräsentiert wird (der Wächter ist immer erfüllt), kann auf eine Darstellung verzichtet werden.

Für die Schaltregel von Ziel/Transitions-Netzen müssen die Regeln zur Auswertung der variablen Prädikatensymbole festlegt werden. Wie bereits angedeutet, hängt die Extension eines variablen Prädikates vom Zustand des Ziel/Transitions-Netzes ab und nicht von einer in der Struktur zugeordneten Relation: Ein variables Prädikat ist genau dann erfüllt, wenn die Argumente des Prädikates in Form eines Tupels die assoziierte Stelle markieren. Umgekehrt bedeutet dies, dass die Negation des Prädikates genau dann erfüllt ist, wenn eine entsprechende Marke nicht vorhanden ist.

Wie in Prädikat/Transitions-Netzen reicht es darüber hinaus nicht aus, die Begriffe Aktiviertheit und Folgemarkierung auf einer Transition zu erklären. Für die Auswertung von Kantengewichten und Wächtern wird eine Belegung der Variablen vorausgesetzt, um etwaige Terme und Formeln zu schließen. Aus diesem Grund wird der für Prädikat/Transitions-Netze verwendete Begriff der *Transitioneninstanz* auf Ziel/Transitions-Netze übertragen. Eine Transitioneninstanz wird unter einer Markierung für aktiviert erklärt, wenn der Wächter der Transition erfüllt ist (für die gegebene Markierung und Belegung) und genug Marken im Vorbereich vorhanden sind, um allen ausgewerteten Kantengewichten zu genügen. Das Schalten einer aktivierten Transitioneninstanz konsumiert die identifizierten Tupel und Formeln von den Stellen des Vorbereichs und produziert neue Tupeln und

Formeln auf den Stellen des Nachbereichs. Dabei ist zu beachten, dass die Gewichte von Kanten an Ziel-Stellen im Gegensatz zu den Gewichten von Kanten an Prädikat-Stellen nicht vollständig durch die gegebene Struktur ausgewertet werden. Zunächst werden die freien Variablen in den Formeln substituiert. Anschließend werden alle Terme vereinfacht.

Definition 3 (Folgemarkierung) *Sei $ZT = (N, A_N, M_0)$ ein Ziel/Transitions-Netz, $t \in T_N$ eine Transition und M eine Markierung von ZT.*

1. *Sei F eine erweiterte Formel. Für die Auswertung von F bei M wird $A_\Sigma(F)$ um folgenden Fall erweitert: Für $F = \widehat{\pi}(t_1, ..., t_n)$ mit $\widehat{\pi} \in \widehat{\Pi}$ ist $A_\Sigma(F) = \top$, wenn $(A_\Sigma(t_1), ..., A_\Sigma(t_n)) \in M(A_P^{-1}(\widehat{\pi}))$, sonst $A_\Sigma(F) = \bot$.*

2. *Sei $\chi(t)$ die Menge aller freien Variablen, die im Wächter von t und in allen Kanten $f = (p,t)$ und $f = (t,p)$ für beliebige $p \in P_N$ vorkommen. Darüber hinaus sei α eine Belegung, die jeder Variablen $x \in X \cap \chi(t)$ einer Sorte $s \in S$ ein Konstantensymbol der Sorte s und jeder Variablen $\widehat{x} \in \widehat{X} \cap \chi(t)$ eine geschlossene Formel zuordnet. Dann heißt $t : \alpha$* Transitioneninstanz.

3. *Sei $t : \alpha$ eine Transitioneninstanz. $t : \alpha$ heißt* aktiviert *unter M und kann schalten genau dann, wenn $A_\Sigma(A_T(t) : \alpha) = \top$ bei M und*

   (a) $A_\Sigma(A_F(p,t) : \alpha) \leq M(p)$ *für alle* $p \in P_N \setminus \widehat{P}_N$,

   (b) $\widehat{A}_\Sigma(A_F(\widehat{p},t) : \alpha) \leq M(\widehat{p})$ *für alle* $\widehat{p} \in \widehat{P}_N$.

4. *Sei $t : \alpha$ eine aktivierte Transitioneninstanz in M. Das Schalten von $t : \alpha$ in M erzeugt eine* Folgemarkierung *M' nach folgender Vorschrift:*

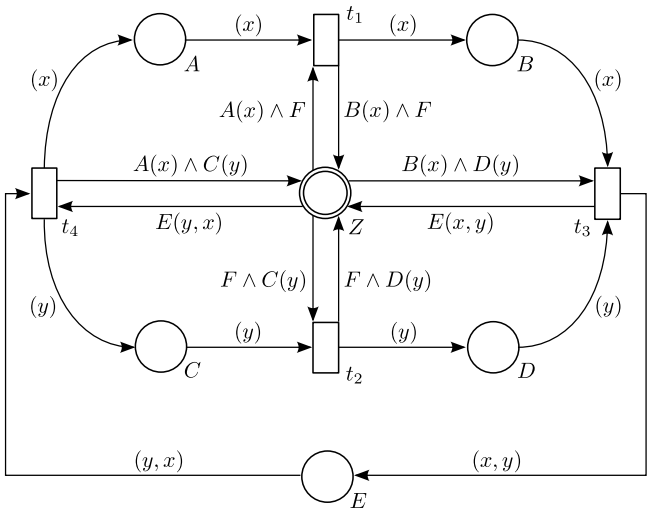
   (a) $M'(p) = M(p) - A_\Sigma(A_F(p,t) : \alpha) + A_\Sigma(A_F(t,p) : \alpha)$ *für alle* $p \in P_N \setminus \widehat{P}_N$.

   (b) $M'(\widehat{p}) = M(\widehat{p}) - \widehat{A}_\Sigma(A_F(\widehat{p},t) : \alpha) + \widehat{A}_\Sigma(A_F(t,\widehat{p}) : \alpha)$ *für alle* $\widehat{p} \in \widehat{P}_N$.

   Das Schalten einer Transitioneninstanz $t : \alpha$ in einer Markierung M zu einer Folgemarkierung M' wird mit $M \xrightarrow{t:\alpha} M'$ oder kurz $M \xrightarrow{t} M'$ notiert. R_{ZT} ist die Menge aller erreichbaren Markierungen von ZT.

5. *Der* Erreichbarkeitsgraph *$G_{ZT} = (V,E)$ von ZT ist gegeben durch die Knotenmenge $V = R_{ZT}$ und die Kantenmenge $E = \{(M, t : \alpha, M') \mid M \xrightarrow{t:\alpha} M'\}$. Ein Tupel $(M, t : \alpha, M')$ beschreibt dabei eine Kante von M nach M', die mit $t : \alpha$ beschriftet ist.*

Grundbegriffe der Petri-Netz-Theorie wie Nebenläufigkeit und Konflikt (von Transitioneninstanzen) lassen sich in gewohnter Weise auf Ziel/Transitions-Netze übertragen.



	Markierung						*Folgemarkierungen*
	A	B	C	D	E	Z	
M_0	(1)		(0)			$A(1) \wedge C(0)$	$M_0 \xrightarrow{t_1:\alpha_1} M_1,\ M_0 \xrightarrow{t_2:\alpha_2} M_2$
M_1		(1)	(0)			$B(1) \wedge C(0)$	$M_1 \xrightarrow{t_2:\alpha_3} M_3$
M_2	(1)			(0)		$A(1) \wedge D(0)$	$M_2 \xrightarrow{t_1:\alpha_4} M_3$
M_3		(1)		(0)		$B(1) \wedge D(0)$	$M_3 \xrightarrow{t_3:\alpha_5} M_4$
M_4					$(1,0)$	$E(1,0)$	$M_4 \xrightarrow{t_4:\alpha_6} M_5$
M_5	(0)		(1)			$A(0) \wedge C(1)$	$M_5 \xrightarrow{t_1:\alpha_7} M_6,\ M_5 \xrightarrow{t_2:\alpha_8} M_7$
M_6		(0)	(1)			$B(0) \wedge C(1)$	$M_6 \xrightarrow{t_2:\alpha_9} M_8$
M_7	(0)			(1)		$A(0) \wedge D(1)$	$M_7 \xrightarrow{t_1:\alpha_{10}} M_8$
M_8		(0)		(1)		$B(0) \wedge D(1)$	$M_8 \xrightarrow{t_3:\alpha_6} M_9$
M_9					$(0,1)$	$E(0,1)$	$M_9 \xrightarrow{t_4:\alpha_5} M_0$

	Belegung		
	x	y	F
α_1	1		$C(0)$
α_2		0	$A(1)$
α_3		0	$B(1)$
α_4	1		$D(0)$
α_5	1	0	
α_6	0	1	
α_7	0		$C(1)$
α_8		1	$A(0)$
α_9		1	$B(0)$
α_{10}	0		$D(1)$

Abb. 3: Ein Ziel/Transitions-Netz mit einer vollständigen Darstellung aller erreichbaren Markierungen und Transitioneninstanzen

In Abbildung 3 ist ein weiteres Beispiel für ein einfaches Ziel/Transitions-Netz gegeben, das die Aktivierung von Transitionen und die Berechnung von Folgemarkierungen verdeutlichen soll. In der Anfangsmarkierung sei $M_0(A) = (1)$, $M_0(C) = (0)$ und $M_0(Z) = A(1) \wedge C(0)$. Die restlichen Stellen sind mit der leeren Multimenge markiert, d.h. sie enthalten keine Marken. Sobald die beiden Anfangsmarken von A und C mit dem Schalten von t_1 und t_2 auf B und D abgelegt wurden, ist t_3 aktiviert. Wenn t_3 schaltet, wird eine zusammengesetzte Marke auf E erzeugt, die wiederum t_4 aktiviert. Mit dem Schalten von t_4 werden die Anfangsmarken daraufhin wieder in umgekehrter Reihenfolge auf A und C abgelegt. Anschließend können alle Transitionen erneut schalten, bis die Anfangsmarkierung des Netzes wieder erreicht ist. Darüber hinaus aktualisiert jede Transition die auf Z vorhandene Formel, so dass sie in jedem Zustand die aktuelle Markierung der Prädikat-Stellen in Form einer Konjunktion repräsentiert. Zu diesem Zweck wird bei Bedarf auf eine Variable F zugegriffen, die mit entsprechenden Teilformeln belegt werden kann. Auf

der Ziel-Stelle Z kann somit stets der aktuelle Zustand des Netzes abgelesen werden. Die erreichbaren Markierungen und die entsprechenden Transitioneninstanzen des Netzes sind ebenfalls in Abbildung 3 dargestellt.

Das Beispiel zeigt, dass Zustände eines Systems sehr einfach durch entsprechende Formeln auf einer Ziel-Stelle beschrieben werden können. Eine derartige Stelle kann für die Modellierung von dynamischen Zielen eines Agenten verwendet werden, indem jedes Ziel durch eine Formel auf der Stelle repräsentiert wird. Durch den üblichen Konsum und die Produktion von Marken beim Schalten des Netzes, können Ziele erstellt, gelöscht und geändert werden. Darüber hinaus lässt sich das Verhalten des Systems durch ihre Verwendung in den Wächtern von Transitionen beeinflussen. Ein konkretes Beispiel für einen einfachen Agenten mit dynamischen Zielen und dem entsprechenden Ziel/Transitions-Netz wird im nächsten Abschnitt dargestellt.

5 Mächtigkeit

Offensichtlich sind Prädikat/Transitions-Netze ein Spezialfall von Ziel/Transitions-Netzen mit $\widehat{P}_N = \emptyset$. Darüber hinaus werden keine erweiterten Formeln in Wächtern verwendet und alle Tupel in den Kantenbeschriftungen enthalten ausschließlich Variablen. Jedes Prädikat/Transitions-Netz kann somit als Ziel/Transitions-Netz dargesellt werden. Die Umkehrung gilt nicht, wie der folgende Abschnitt zeigt.

Prädikat/Transitions-Netze besitzen nicht die selbe Mächtigkeit wie Turingmaschinen, d.h. mit ihnen können nicht alle berechenbaren Funktionen berechnet werden. Das Problem kann auf die Unfähigkeit von Prädikat/Transitions-Netze zurückgeführt werden, einen Nulltest auf einer Stelle durchzuführen. Es ist zwar möglich, durch Anliegen einer Transition im Nachbereich zu prüfen, ob sich *mindestens* eine Marke auf einer Stelle befindet. Es ist im Allgemeinen aber nicht möglich, zu prüfen, ob sich *keine* Marke auf der Stelle befindet. Eine Transition, die bei einer *sauberen* Stelle aktiviert ist, ist auch bei jeder anderen Anzahl von Marken auf der Stelle aktiviert. Es existieren Erweiterungen wie z.B. Inhibitor-Netze, um diesem Problem zu begegnen. In Inhibitor-Netzen wird durch eine Abbildung $I : T_N \to 2^{P_N}$ für jede Transition eine Menge von Inhibitorkanten festlegt (siehe z.B. [PW03]). Inhibitorkanten verlaufen dabei stets von einer Stelle zu einer Transition und werden mit einem Kreis statt mit einer Pfeilspitze gezeichnet. Mit Inhibitorkanten wird die Aktiviertheit einer Transition $t \in T_N$ unter einer Markierung M eingeschränkt, indem zusätzlich verlangt wird, dass alle Stellen in $I(t)$ keine Marken enthalten dürfen, d.h. $\forall p \in I(t) : M(p) = 0$.

Wie bereits in Abschnitt 2 angedeutet, lassen sich Nulltests ebenfalls mit Ziel/Transitions-Netzen realisieren. Dazu werden entsprechende Wächter mit variablen Prädikaten eingeführt, die die selben Restriktionen beschreiben wie Inhibitorkanten in Inhibitor-Netzen. Ziel/Transitions-Netze können das Verhalten von Inhibitor-Netzen also simulieren. Ein Beispiel ist in Abbildung 4 dargestellt.

Theorem 1 *Zu jedem Prädikat/Transitions-Netz H mit Inhibitorkanten existiert ein verhaltensgleiches Ziel/Transitions-Netz $ZT(H)$, d.h. H und $ZT(H)$ haben den selben Erreichbarkeitsgraphen.*

Beweis: $ZT(H)$ wird mit dem selben Netz, den selben Beschriftungen und der selben Anfangsmarkierung wie H konstruiert. Für jeden Inhibitorbogen von einer Stelle $p \in I(t)$ zu einer Transition $t \in T_N$ im Inhibitor-Netz H wird eine Formel $G_p = \exists x_1 :, ..., \exists x_n : A_P(p)(x_1, ..., x_n)$ konstruiert, so dass $x_1, ..., x_n$ genau dann Variablen aus $X_{s_1}, ..., X_{s_n}$ sind, wenn $A_P(p) \in \Pi_{s_1,...,s_n}$. Der Wächter von t im Ziel/Transitions-Netz $ZT(H)$ kann dann durch $A_T(t) = \bigwedge_{p \in I(t)}(\neg G_p)$ erweitert werden. Befindet sich in einer beliebigen Markierung M mindestens ein Tupel auf eine Stelle $p \in I(t)$ im Inhibitor-Netz H, dann ist das Konjunktionsglied $\neg G_p$ im Wächter von t des Ziel/Transitions-Netzes $ZT(H)$ nicht erfüllt und t kann nicht aktiviert unter M. Der Wächter $A_T(t)$ in $ZT(H)$ repräsentiert somit genau die Restriktion, die die entsprechenden Inhibitorkanten von t in H beschreiben. Alle Zustände und Zustandsübergänge von H und $ZT(H)$ sind daher identisch. □

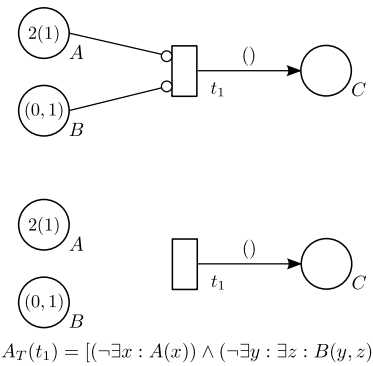


Abb. 4: Oben: Ein Inhibitor-Netz mit zwei Inhibitorkanten. Unten: Ein verhaltensgleiches Ziel/Transitions-Netz ohne Inhibitorkanten. In beiden Fällen ist t_1 nicht aktiviert, da sowohl die Stelle A als auch die Stelle B Marken enthält.

Korollar 1 *Ziel/Transitions-Netze sind turingmächtig.*

Beweis: Folgt aus Theorem 1 und dem Beweis, dass Inhibitor-Netze im Allgemeinen turingmächtig sind (siehe z.B. [PW03, S. 166]). □

6 Anwendungsbeispiel

In diesem Abschnitt soll ein einfacher Agent mit dynamischen Zielen als Ziel/Transitions-Netz modelliert werden. Als Beispiel dient eine einfache Variante des klassischen BlocksWorld-Szenarios aus [Woo02]. In diesem Szenario existieren unterschiedliche Klötzchen, die aufeinander gestapelt werden können. Die Aufgabe des Agenten besteht darin, durch Auf- und Abstellen von Klötzchen eine vordefinierte Anordnung zu erzeugen. Dabei soll er jedoch das unnötige Umstellen von Klötzchen vermeiden, d.h. die Lösung soll

nicht darin bestehen, zunächst alle vorhandenen Stapel abzubauen. Darüber hinaus wird in diesem Beispiel davon ausgegangen, dass jeder Stapel aus nicht mehr als zwei Klötzchen bestehen kann.

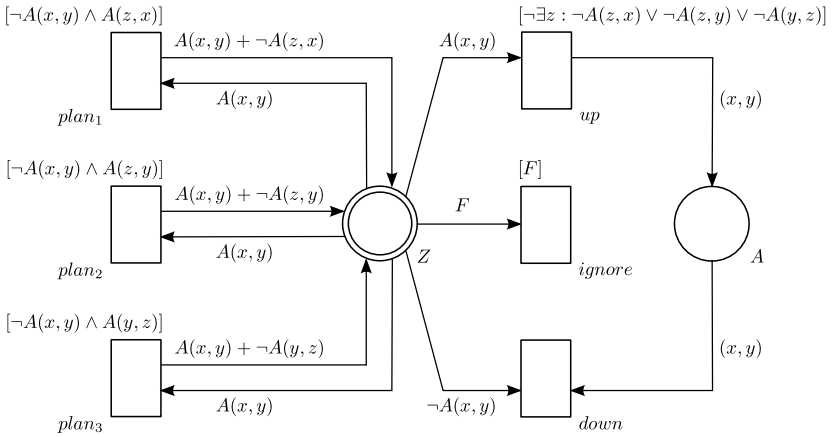


Abb. 5: Ein BlocksWorld-Agent als Ziel/Transitions-Netz.

Abbildung 5 zeigt ein Ziel/Transitions-Netz, das einen BlocksWorld-Agenten und seine Umgebung modelliert. Die Umgebung des Agenten wird durch die Prädikat-Stelle A repräsentiert, in der die aktuelle Anordnung der Klötzchen fixiert ist. Ein Tupel (x,y) bedeutet, dass das Klötzchen x auf dem Klötzchen y gestapelt wurde. Existiert für ein Klötzchen x kein entsprechendes Tupel, dann befindet sich x auf dem Boden. Die (Teil-)Ziele des Agenten werden durch positive und negative Literale auf der Ziel-Stelle Z festgehalten und beschreiben die gewünschte Anordnung der Klötzchen. Positive Literale legen fest, welche Klötzchen aufeinander zu stapeln sind. Negative Literale legen fest, welche Klötzchen nicht aufeinander gestapelt werden dürfen. Als Zielvorgabe sind aber auch unvollständige Konfigurationen zulässig. Das Aufstellen eines Klötzchens findet durch Schalten der Transition *up* statt. Sie entfernt ein Literal der Form $A(x,y)$ von der Ziel-Stelle Z und stellt das Klötzchen x auf das Klötzchen y, indem das Tupel (x,y) auf A produziert wird. Der Wächter von *up* garantiert, dass ein Stapelvorgang nur dann durchgeführt werden kann, wenn x und y nicht verstellt sind und y nicht bereits auf ein anderes Klötzchen gestellt wurde. Die Transition *down* baut einen vorhandenen Stapel (x,y) wieder ab, wenn ein Literal der Form $\neg A(x,y)$ auf Z vorliegt. Dabei wird das Literal mit dem Schalten von *down* ebenfalls wieder von Z entfernt. Sollte ein beliebiges Ziel auf Z bereits erfüllt sein, dann kann es vom Agenten ignoriert und mit dem Schalten der Transition *ignore* gelöscht werden. Die Transitionen $plan_1$, $plan_2$ und $plan_3$ repräsentieren mentale Prozesse des Agenten. Sie sind immer dann aktiviert, wenn ein Ziel auf Z einen Stapel (x,y) verlangt, der Stapel aber zunächst nicht erzeugt werden kann. Dies ist der Fall, wenn ein Klötzchen z das Klötzchen x oder y verstellt ($plan_1$ oder $plan_2$) oder wenn das Klötzchen y bereits auf ein Klötzchen z gestellt wurde ($plan_3$). Die Wächter der Transitionen realisieren die notwendigen Fallunterscheidungen. Darüber hinaus setzen sie durch das Konjunktionsglied $\neg A(x,y)$ voraus, dass der geforderte Stapel noch nicht existiert. Wenn $plan_1$, $plan_2$ und $plan_3$ schalten, werden neue Ziele auf der Ziel-Stelle Z erzeugt, um x und y frei zu stellen. Auf diese Weise kann ein BlocksWorld-Agent ausgehend von einer beliebigen

Anfangs-Konfiguration der Klötzchen in A nacheinander alle notwendigen Schritte durchführen, um die Ziel-Konfiguration in Z zu erreichen. Abbildung 6 zeigt einen möglichen System-Ablauf für die Anfangs-Konfiguration $M_0(G) = (1,2) + (3,4) + (5,6) + (7,8)$ und die Ziel-Konfiguration $M_0(Z) = A(4,1) + A(6,5) + \neg A(3,2)$.

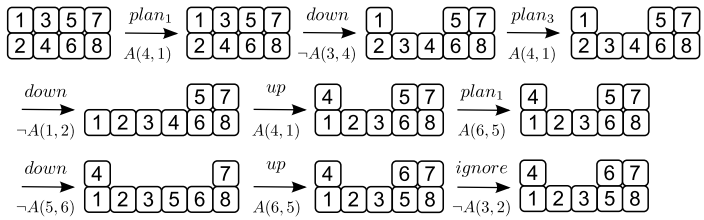


Abb. 6: Ein Ablauf eines BlocksWorld-Agenten. Die gezeigten Klötzchen entsprechen der Markierung der Prädikat-Stelle A. Neben den Transitionen sind auch die Literale der Ziel-Stelle Z dargestellt, die in jedem Schaltvorgang berücksichtigt wurden.

Das Beispiel in diesem Abschnitt zeigt einige Elemente, die den Unterschied zu klassischen Prädikat/Transitions-Netzen verdeutlichen. So ist die Unterscheidung zwischen dem Aktivwerden der Transitionen $plan_1$, $plan_2$, $plan_3$ und up auf entsprechende Wächter zurückzuführen, die die Beschaffenheit der Markierung von A abfragen. Diese Abfragen werden durch die Verwendung von variablen Prädikaten realisiert und sind in dieser Form nicht in Prädikat/Transitions-Netzen möglich. Dies gilt ebenfalls für die Darstellung der positiven und negativen Literale (Ziele). Um diese Mechanismen in Prädikat/Transitions-Netzen zu realisieren und einzubetten, sind komplexere Netz-Strukturen notwendig, d.h. weitere Stellen, Transitionen und Kanten werden benötigt. In Ziel/Transitions-Netze können die entsprechenden Abfragen und Ziele intuitiv und kompakt modelliert werden.

7 Ausblick

In dieser Arbeit wurde mit der Klasse der Ziel/Transitions-Netze ein Ansatz vorgeschlagen, um dynamische Ziele von Agenten mit Petri-Netzen zu modellieren. Auf diese Weise lassen sich die Vorteile von Petri-Netzen auch bei der Modellierung von Agentensystemen nutzen. Für die Darstellung von ziel-orientiertem Verhalten in Agentensystemen soll in den kommenden Arbeiten eine entsprechende Ablaufsemantik für Ziel/Transitions-Netze entwickelt werden, die auf einer kausalen Semantik (Prozesse, siehe z.B. [DFO97]) basiert. Mit der Einführung einer geeigneten Ablaufsemantik wird ein zentraler Vorteil von Ziel/-Transitions-Netzen deutlich: Ziele und Zustände können in einem Zusammenhang dargestellt werden, so dass ihre Wechselwirkung untersucht werden kann. Auf diese Weise ist es möglich, ziel-orientiertes Verhalten von Agenten nachzuweisen. Dies ist insbesondere für die Validierung von Agenten-Modellen von Bedeutung.

Literaturverzeichnis

[BMO00] B. Bauer, J. Müller und J. Odell. Agent UML: A Formalism for Specifying Multiagent Software Systems. In P. Ciancarini und M. J. Wooldridge, Hrsg., *Agent-Oriented Software Engineering: First International Workshop*, Lecture Notes in Computer Science, Volume 1957, Seiten 91–104. Springer, 2000.

[CL90] P. R. Cohen und H. J. Levesque. Intention Is Choice with Commitment. *Artificial Intelligence*, 42(2-3):213–261, 1990.

[DFO97] J. Desel, T. Freytag und A. Oberweis. Causal-semantic-based simulation and validation of high-level Petri nets. In A. R. Kaylan und A. Lehmann, Hrsg., *Proceedings of 11th European Simulation Multiconference*, Seiten 826–831, 1997.

[DHK01] R. Depke, R. Heckel und J. Malte Küster. Agent-Oriented Modeling with Graph Transformation. In P. Ciancarini und Michael Wooldridge, Hrsg., *Agent-Oriented Software Engineering: First International Workshop*, Jgg. 1957 of *Lecture Notes in Computer Science*, Seiten 105–120. Springer, 2001.

[Gen87] H. J. Genrich. Predicate/Transition Nets. In W. Brauer, W. Reisig und G. Rozenberg, Hrsg., *Petri Nets: Central Models and Their Properties. Advances in Petri Nets*, Lecture Notes in Computer Science, Volume 254, Seiten 207–247. Springer, 1987.

[GL81] H. J. Genrich und K. Lautenbach. System Modelling with High-Level Petri Nets. *Theoretical Computer Science*, 13:109–136, 1981.

[HKHK+07] K. Hölscher, R. Klempien-Hinrichs, P. Knirsch, H.-J. Kreowski und S. Kuske. Autonomous Units: Basic Concepts and Semantic Foundation. In M. Hülsmann und K. Windt, Hrsg., *Understanding Autonomous Cooperation and Control in Logistics - The Impact of Autonomy on Management, Information, Communication and Material Flow*, Seiten 103–120. Springer, 2007.

[KR91] L.P. Kaelbling und S.J Rosenschein. Action and planing in embedded agents. In P. Maes, Hrsg., *Designing Autonomous Agents: Theory and Practice from Biology to Engineering and Back*, Seiten 35–48. MIT Press, 1991.

[MW97] D. Moldt und F. Wienberg. Multi-Agent-Systems based on Coloured Petri Nets. In P. Azema und G. Balbo, Hrsg., *Application and Theory of Petri Nets 1997: 18th International Conference*, Jgg. 1248 of *Lecture Notes in Computer Science*, Seiten 241–249. Springer, 1997.

[PW03] L. Priese und H. Wimmel. *Theoretische Informatik: Petri-Netze*. Spinger, 2003.

[RG95] A. S. Rao und M. P. Georgeff. BDI agents: From theory to practice. In V. Lesser und L. Gasser, Hrsg., *Proceedings of the First International Conference on Multi-Agent Systems*, Seiten 312–319. MIT Press, 1995.

[Sho93] Y. Shoham. Agent-oriented Programming. *Artificial Intelligence*, 60(1):51–92, 1993.

[Woo02] M. Wooldridge. *An Introduction to MultiAgent Systems*. Wiley, 2002.

[XVIY03] D. Xu, R. Volz, T. Ioerger und J. Yen. Modeling and Analyzing Multi-Agent Behaviors Using Predicate Transition Nets. *International Journal of Software Engineering and Knowledge Engineering*, 13:103–124, 2003.

Integration automatisch generierter und manuell konstruierter Prozessmodelle

Susanne Leist[1] und Wolfgang Lichtenegger[2]

Abstract: Die Modellierung von Prozessen wird als wichtiges Aufgabengebiet in Wissenschaft und Praxis gesehen. Obwohl zahlreiche Methoden zur manuellen Prozessmodellierung und Algorithmen zum Process Mining entwickelt wurden, bestehen immer noch Probleme bei der praktischen Anwendung. Ein Ansatz diesen Problemen zu begegnen, besteht in der Integration manuell konstruierter und automatisch generierter Prozessmodelle. Auf diese Weise lässt sich die Korrektheit des Modells erhöhen sowie der Erstellungs- bzw. Aktualisierungsaufwand reduzieren. Die Durchführung der Integration manuell konstruierter und automatisch generierter Prozessmodelle ist noch Gegenstand der Forschung. Im vorliegenden Beitrag werden deshalb der Integrationsvorgang analysiert, Einflussfaktoren identifiziert, Ansatzpunkte zur systematischen Durchführung der Integration aufgezeigt und ein erster Ansatz zur Integration auf Basis eines Fallbeispiels entwickelt.

1 Einleitung

Das Geschäftsprozessmanagement ist für Unternehmen nicht nur in Zeiten einer wirtschaftlichen Krise ein bedeutendes Aufgabengebiet. Dabei richtet sich aktuell der Schwerpunkt des Interesses, blickt man beispielsweise auf die Themen und Referate von in der Praxis wichtiger Tagungen (z. B. Business Process Management Conference Europe 2009, Gartner Business Process Management Summit 2009), vor allem auf die Analyse, aktive Steuerung und Verbesserung bestehender Prozesse. Doch setzten gerade diese Aufgaben voraus, dass die bestehenden Prozesse in guter Qualität dokumentiert wurden. Mit einem aktuellen und ausreichend detailliert beschriebenen Ist-Prozessmodell lässt sich beispielsweise die Erfüllung von Anforderungen des Sarbanes Oxley Acts nachweisen, oder die Grundlage für eine Prozesssteuerung schaffen [Kü05, S. 961; Re09, S. 334]. Somit bleibt die Modellierung der Ist-Prozesse für die Unternehmen nach wie vor von hoher Relevanz.

Nicht nur aus diesem Grunde wird die Modellierung der Prozesse schon seit langer Zeit intensiv diskutiert, und es wurden zahlreiche Konzepte und Methoden entwickelt, die hierbei unterstützen. Trotzdem charakterisieren Unternehmen die gegenwärtige Reife der Geschäftsprozesse häufig als unzureichend (siehe [ARW08, S. 297]). Dies zeigt sich oft auch darin, dass nicht mehr aktuelle oder nur unvollständige Prozessmodelle vorliegen. Einen wichtigen Beitrag zur Aktualität und Korrektheit der Modelle kann das Process Mining [Aa03b], i. S. der automatischen Generierung von Prozessmodellen, leisten. Das

[1] Lehrstuhl für Business Engineering, Universität Regensburg, Universitätsstraße 31, 93040 Regensburg, susanne.leist@wiwi.uni-regensburg.de
[2] Lehrstuhl für Business Engineering, Universität Regensburg, Universitätsstraße 31, 93040 Regensburg, wolfgang.lichtenegger@ wiwi.uni-regensburg.de

Process Mining stützt sich dabei auf Ereignisse, die von den Anwendungssystemen des Unternehmens erzeugt werden. Jedoch ist es häufig, wie auch im Fallbeispiel (siehe Abschnitt fünf) gezeigt, nicht möglich, den Prozess auf dieser Basis durchgängig, d. h. von seiner Initiierung bis zu seinem Abschluss, zu generieren. Für die durchgängige Erfassung des Prozesses ist somit eine Integration automatisch generierter und manuell konstruierter Modelle notwendig.

Das Ziel des Beitrags ist es, die Problemstellung der Integration manuell konstruierter und automatisch generierter Prozessmodelle zu strukturieren sowie Ansätze zu deren Lösung zu geben. Hierzu werden im nachfolgenden zweiten Abschnitt die beiden Modellierungstechniken (manuelle Konstruktion und automatische Generierung) vorgestellt und die Vorteilhaftigkeit der Integration begründet. Im dritten Abschnitt werden Herausforderungen bei der Integration identifiziert und deren Grundlagen in einem morphologischen Kasten zusammengestellt. Anschließend wird in Abschnitt vier die bestehende Literatur nach vorhandenen Lösungsansätzen untersucht. Da keiner der bestehenden Ansätze den Integrationsprozess vollständig unterstützt, wird in Abschnitt fünf der eigene Ansatz skizziert. Die Schlussbetrachtung in Abschnitt sechs fasst die wichtigsten Ergebnisse zusammen und stellt den weiteren Forschungsbedarf vor.

2 Thematische Grundlagen

Prozessmodelle werden in Unternehmen sehr vielseitig eingesetzt. Beispielsweise übernehmen sie eine wichtige Transferfunktion, da mit ihnen die Anforderungen aus der Unternehmensstrategie für die betrieblichen Abläufe umgesetzt werden. Zudem ermöglichen sie, Restriktionen und Potenziale der IT-Infrastruktur für die Unternehmensstrategie zu verdeutlichen (vgl. [Fr94, S. 230-231; Ös95, S. 20-21]). Darüber hinaus können sie für viele weitere Zwecke, wie beispielsweise das Qualitätsmanagement, bei der Einführung von Standardsoftware, dem Compliance-Management oder bei Sourcing-Entscheidungen, und nicht zuletzt für die Prozessoptimierung verwendet werden [RSD05, S. 51-57].

Für viele der genannten Einsatzzwecke werden Modelle im Ist-Zustand verwendet. Da erst durch die Kenntnis des Ist-Zustandes die nötige Transparenz geschaffen wird, um beispielsweise Grundlagen für Steuerungsentscheidungen zu legen oder um den Nachweis der Erfüllung von Anforderungen aus dem Compliance Management zu erbringen. Eine wichtige Voraussetzung ist dabei, dass die Prozessmodelle aktuell sind und die Abläufe des Unternehmens korrekt und vollständig darstellen. Die Abläufe sind vollständig beschrieben, wenn die Prozesse durchgängig vom unternehmensexternen Leistungsnehmer (v. a. den Endkunden) bis hin zu den unternehmensexternen Leistungserbringern (Lieferanten oder Tochterunternehmen) erfasst werden (end-to-end). Die mit dieser Voraussetzung verbundenen Probleme werden im folgenden Abschnitt betrachtet. Unterschieden werden dabei die manuelle Konstruktion und die automatische Generierung von Prozessmodellen als grundlegende Modellierungstechniken.

2.1 Manuelle Konstruktion der Prozesse

Zur Unterstützung der manuellen Konstruktion von Prozessmodellen werden seit den 90er Jahren Ansätze in der Literatur vermehrt publiziert (siehe [FS95; KNS92]). Neben diesen Ansätzen wurden auch viele Modellierungssprachen entwickelt, die ebenfalls bei der Konstruktion helfen sollen. Trotz dieser zahlreichen Bemühungen besteht nach wie vor eine wesentliche Problemstellung darin, ein gleiches Verständnis über das Modell zwischen Modellierer, Fachexperten und Modellnutzer herzustellen [Le06, S. 83]. Schwierigkeiten ergeben sich insbesondere daraus, dass die Fachexperten und Modellnutzer gewohnt sind, in natürlicher Sprache zu kommunizieren, während die Modellierer künstliche, konstruierte Sprachen verwenden. Da die Modellnutzer die Anforderungen an die Prozessmodelle bestimmen, die Fachexperten das Basiswissen über die Prozesse zur Verfügung stellen und das Wissen über die Modellierung von den Modellierern beigesteuert wird, ist eine intensive Kommunikation zwischen diesen Gruppen erforderlich. Neben den damit verbundenen hohen Aufwand [SL05, S. 155], wird als weitere Problemstellung die möglicherweise idealisierte Sicht auf die Prozesse durch die Fachexperten gesehen [Aa07, S. 18], die zur Modellierung nicht korrekter Abläufe führt.

Veränderungen am Markt, neue Anforderungen der Kunden oder Geschäftspartner wie auch neue gesetzliche Vorgaben sind nur einige Beispiele für Entwicklungen, die meist auch zu einer Anpassung der Abläufe und damit der Prozesse des Unternehmens führen [BK05, S.3]. In Konsequenz ist es daher notwendig nach der ersten Erstellung eines Prozessmodells, dieses stetig zu aktualisieren. Oben genannte Probleme sind bei Aktualisierungen zwar meist vergleichsweise geringer, da sich die Anpassungen in der Regel nur auf einen Teilausschnitt des Prozesses beziehen. Als neues Problem hinzu kommt jedoch, wie zeitnah Änderungen in den Prozessmodellen dokumentiert werden und ob Seiteneffekte der Änderungen erkannt und berücksichtigt werden.

Die Nachteile der manuellen Modellierung lassen sich zusammenfassend im hohen Aufwand bei der ersten Erstellung und auch bei den stetigen Aktualisierungen sehen. Im Weiteren können die Aktualität und die Korrektheit des Modells beeinträchtigt sein. Ein nicht korrektes Modell kann dabei aufgrund der idealisierten Sicht des Fachexperten oder durch Kommunikationsprobleme zwischen Modellnutzer, Fachexperte und Modellierer entstehen.

2.2 Automatische Generierung der Prozesse

Eine wichtige Aufgabe des Process Minings ist die automatische Modellgenerierung auf Basis von in der Vergangenheit ausgeführten Prozessinstanzen [AW04, S. 232]. Grundlage sind auf Ereignissen (z.B. „eine Aufgabe wurde abgeschlossen“, „eine Nachricht wurde ausgetauscht“) aufbauende Eventlogs, die für jede Prozessinstanz (z.B. „der Kreditantrag von Herrn Schmid wurde geprüft“) mit einem Zeitstempel aufgezeichnet werden. Aus den erfassten Eventlogs wird anschließend mit Hilfe von Process Mining-Algorithmen automatisch ein Modell abgeleitet, das wiedergibt, wie der Prozess von den Modellnutzern und Anwendungssystemen tatsächlich ausgeführt wurde [Aa05, S. 199].

Ist Process Mining einmal für einen Prozess implementiert, lassen sich auf Knopfdruck aktuelle Prozessmodelle ableiten. Allerdings setzt das Process Mining voraus, dass eine ausreichende Anzahl von Eventlogs erfasst werden kann, die den Kontrollfluss sinnvoll beschreiben. Dabei wird ein Process Mining-Algorithmus den Prozess in Abhängigkeit der gesetzten Start- und Endereignisse zwar immer durchgängig, aber nicht unbedingt auch vollständig erfassen. Zudem müssen die beobachteten Instanzen repräsentativ sein und es muss eine ausreichend große Anzahl möglicher Durchläufe aufgezeichnet sein [Aa05, S. 201]. Weitere Herausforderungen ergeben sich bei der Generierung von nicht trivialen Konstrukten, beispielsweise im Eventlog verborgenen Aktivitäten sowie von Oder-Konstrukten mit vorgegebener Entscheidung. Ebenso ist der Umgang mit verrauschten Daten (fehlende oder fehlerhafte Werte) problematisch. [AW04, S. 237-242] Lassen sich diese Herausforderungen nicht beheben, kann das Process Mining keine korrekten oder vollständigen Modelle erzeugen.

2.3 Gegenüberstellung und Kombination

Die Kombination beider Modellierungstechniken kann in mehrfacher Hinsicht bei der Modellierung von Nutzen sein. So können bspw. über den Vergleich von einem mit Process Mining generierten Modell mit einem manuell konstruierten Modell wichtige Erkenntnisse über gedachte und tatsächliche Ausführung der Aufgaben durch die Modellnutzer und Anwendungssysteme gewonnen werden [Aa07, S. 16-17].

	Vorteile	Nachteile
Manuelle Konstruktion	Alle Aufgaben eines Prozesses können erfasst und in beliebiger Tiefe modelliert werden.	• Hoher Aufwand bei der Erstellung und Aktualisierung des Prozessmodells • Korrektheit schwer erreichbar • Aktualität im Zeitablauf schwer einhaltbar
Automatische Generierung	• Kaum Aufwand bei der wiederholten Generierung des Prozessmodells, d.h. die Modelle sind leicht aktualisierbar • Tatsächliche Ausführung der Aufgaben des Prozesses anhand der Instanzen wird ermittelt, d.h. die Modelle sind korrekt • Die Modelle werden im Rahmen der definierten Start- und Endereignisse immer durchgängig erfasst	• Aufwand bei der ersten Implementierung • Die Aufgaben eines Prozesses können nur über Eventlogs erfasst werden, d.h. die Vollständigkeit ist nicht sichergestellt. • Die Anzahl der betrachteten Instanzen muss repräsentativ sein. • Die Anzahl der betrachteten Durchläufe muss ausreichend hoch sein.

Tab. 1: Gegenüberstellung der grundlegenden Modellierungstechniken

Darüber hinaus und im Folgenden betrachtet, können beide Modellierungstechniken ergänzend eingesetzt werden. Dies ist naheliegend, da Process Mining Nachteile der manuellen Konstruktion beheben kann. Und umgekehrt die manuelle Konstruktion auch dort angewendet werden kann, wenn das Process Mining keine korrekten oder vollstän-

digen Modelle generieren kann (siehe Tabelle 1). Ziel einer solchen Integration ist es, den Aufwand der manuellen Konstruktion bei der Modellierung zu reduzieren sowie die Korrektheit, Aktualität und Vollständigkeit des Modells zu erhöhen. Dabei werden die Abschnitte eines Prozesses mit Process Mining Modelle generiert, die über Eventlogs erfasst werden können. Für die anderen Abschnitte werden in Zusammenarbeit von Modellierer, Fachexperte und Modellnutzer die Abläufe erhoben. Offen ist dabei jedoch, wie die manuell konstruierten und automatisch generierten Modelle (der Prozessausschnitte) in ein konsistentes Modell integriert werden können. Dies wird im Folgenden untersucht.

3 Herausforderungen bei der Integration der Modelle

Zielsetzung der Integration ist, aus dem manuell konstruierten und dem automatisch generierten (Ausgangs-)Modellen ein konsistentes (Ziel-)Modell des Ist-Zustandes zu entwickeln. Die Güte dieses Zielmodells ist nicht nur davon abhängig, wie die Ausgangsmodelle integriert werden (z.B. verbunden oder vereinigt), sondern auch von deren Beschaffenheit. Dementsprechend lassen sich die ersten Herausforderungen aus den Eigenschaften der Ausgangsmodelle ableiten.

- Die erste auffälligste Eigenschaft eines Modells ergibt sich aus der verwendeten *Modellierungssprache*, da die Konstrukte der Modellierungssprache im Wesentlichen die Ausdrucksmächtigkeit des Modells begrenzen. Herausforderungen bei der Integration ergeben sich vor allem, wenn die Ausgangsmodelle in unterschiedlichen Sprachen konstruiert oder generiert werden und deshalb die Transformation eines Modells erfordern. Beispielsweise werden Petri Netze häufig als Modellierungssprache von Process Mining Algorithmen verwendet [TTM08, S. 7], während die EPK bei der manuellen Konstruktion sehr weit verbreitet ist [Pe09, S. 10].
- Bei automatisch generierten Prozessmodellen ist im Weiteren zu unterscheiden mit welchen Algorithmen diese erstellt wurden (*Process Mining Algorithmus)*. Zahlreiche Ansätze zur Generierung von Prozessmodellen werden im ProM Framework [Do05; Pr09b], einem Open Source Tool für Process Mining Forschung, unterstützt. Bei der oben angesprochenen Herausforderung der Transformation der Modellierungssprachen können sich weitere Probleme ergeben, da manche Algorithmen zur Einhaltung der Sprachsyntax fiktive Ereignisse bzw. Funktionen erstellen.
- Auch die für die *Bezeichnung* der Modellkonstrukte verwendete Sprache kann unterschiedlich sein. Hier können sich insbesondere Herausforderungen durch auftretende Konflikte (u.a. [BLN86, S. 344-347] und [Ro96, S. 187-224]) ergeben. Ein Konflikt zwischen zwei Modellen tritt dann auf, wenn ein gleicher Realitätsausschnitt unterschiedlich abgebildet wird. Dazu gehören beispielsweise Namenskonflikte (Verwendung homonymer oder synonymer Bezeichnungen), Typkonflikte (ein Informationsobjekt wird in den beiden Ausgangsmodellen in unterschiedlichen Typen dargestellt, bspw. als Ereignis oder Funktion) und Struktur-

konflikte (gleiche Sachverhalte werden semantisch unterschiedlich modelliert, z. B. durch Verwendung unterschiedlicher Operatoren) [Ro96, S. 187; 207; 216]. Diese Konflikte können durch ein Fachbegriffsmodell (oder Glossar) sowie durch Konventionen, die die Verwendung der Konstrukte festlegen, weitgehend vermieden werden.

- Darüber hinaus ist zu berücksichtigen nach welchem Prinzip die Prozessmodelle entwickelt wurden *(Prinzip der Modellierung)*. So schlagen die Modellierungsmethoden verschiedene Anhaltspunkte vor, an denen Prozesse und Teilprozesse abgegrenzt werden. Zur manuellen Konstruktion wird beispielsweise von Österle die Ausrichtung auf Leistungen und Objekte vorgeschlagen [Ös95, S. 87-88], während Ferstl und Sinz Regeln ausgerichtet an den Transaktionen und Objekten [FS95, S. 214] vorgeben [Le06, S. 289]. Dagegen sind automatisch generierte Modelle an die Verfügbarkeit von Eventlogs gebunden und entsprechend nach ihnen ausgerichtet [AW04, S. 232]. Herausforderungen ergeben sich bei der Integration durch die Angleichung der Modelle, die durch Verwendung unterschiedlicher Modellierungsprinzipien entstanden sind.
- Letztlich kann auch der *Detaillierungsgrad*, in dem die beiden Ausgangsmodelle vorliegen, unterschiedlich sein. Hieraus können sich Konflikte ergeben und die Herausforderung besteht, eine Angleichung der Modelle vorzunehmen.

Neben diesen modellbasierten Herausforderungen, die die Voraussetzungen der Integration beeinflussen, sind weitere zu berücksichtigen, die sich durch die Integration selbst ergeben. Im Fokus stehen hierbei Aspekte, die konkret die Ausführung der Integration beeinflussen und bspw. unterschiedliche Aktivitäten zur Integration notwendig machen. Entsprechend ergeben sich weitere Herausforderungen für den Modellintegrator.

- Ganz maßgeblich wird das Vorgehen der Integration dadurch beeinflusst, ob die Ausgangmodelle bereits generiert bzw. konstruiert sind *(generelles Vorgehen der Integration)*. Liegen bereits generierte und manuell konstruierte Prozessmodelle vor, werden sie rekonstruktiv integriert. Werden die Anforderungen der Integration bereits bei der Erstellung der Prozessmodelle einbezogen, so ist das Vorgehen konstruktiv. [Ro96, S. 172]
- In diesem Beitrag wird die Integration manuell konstruierter und automatisch generierter Modellen betrachtet *(Art der Ausgangsmodelle)*. Jedoch grundsätzlich denkbar ist auch die Integration von nur automatisch generierten oder nur manuell konstruierten Modellen, wie zum Beispiel beim Aufbau einer Prozessarchitektur.
- Auch die Integration selbst kann in unterschiedlicher Art durchgeführt werden *(Art der Integration)*: die Ausgangsmodelle werden verbunden oder vereinigt [Ro99, S. 5; Ju06, S. 38-39].
- Die *Richtung der Integration* kann zwischen horizontal und vertikal differenziert werden [Me07, S. 5], je nachdem ob die zu integrierenden Ausgangsmodelle auf

gleicher Detaillierungsstufe (horizontal) oder auf unterschiedlichen Detaillierungsstufen (vertikal) zu integrieren sind.

- Die vertikale Integration kann unterschiedlichen, grundlegenden Prinzipien folgen (*Prinzip zur vertikalen Integration)*. So werden drei Typen von Beziehungen [Br03, S. 77] zwischen den Ausgangsmodellen auf abstrakter und detaillierter Ebene definiert: Generalisierung/Spezialisierung, Aggregation/Disaggregation, Generizität/ Instanziierung.

- Das *Prinzip der horizontalen Integration* beschreibt, welche Beziehung Prozessteile zu einander haben können. Sie kann eine Sequenz, Parallelität oder Entscheidung sein [Aa03a, S. 10-17].

Merkmal	**Ausprägungen**					
Modellierungssprache	EPK	BPMN	Stellen-Transitions-Netz	Heuristisches Netz	…	
Process Mining Algorithmus	Alpha+	Genetic Mining	Fuzzy Miner	Region Mining	…	
Bezeichnung	Keine Konvention		Fachbegriffsmodell		Konventionen	
Prinzip der Modellierung	Objekte	Leistungen	Transaktionen		Verfügbare Eventlogs	
Detaillierungsgrad	Übereinstimmender Detaillierungsgrad		Abweichender Detaillierungsgrad			
Generelles Vorgehen der Integration	Konstruktiv		Rekonstruktiv			
Art der Ausgangsmodelle	Manuell konstruierte Modelle		Manuell konstruiertes und generiertes Modell		Automatisch generierte Modelle	
Art der Integration	Verbinden		Vereinigen			
Richtung der Integration (Prinzip der horizontalen bzw. vertikalen Integration)	Horizontal			Vertikal		
	Sequenz	Parallelität	Entscheidung	Generalisierung/ Spezialisierung	Aggregation/ Disaggregation	Generizität/Instanziierung

Tab. 2: morphologischer Kasten der Einflussfaktoren

Sowohl die aufgezählten Modelleigenschaften wie auch die unterschiedlichen Aspekte der Integration werden in einem morphologischen Kasten (siehe Tabelle 2) zusammengefasst und jeweils um mögliche Ausprägungen ergänzt. Dabei wird nicht der Anspruch erhoben, die Ausprägungen vollständig zu erfassen, was insbesondere bei den Modellierungssprachen zu sehen ist. Der morphologische Kasten gibt eine erste Übersicht und

strukturiert in einem ersten Ansatz die Vielfalt an Herausforderungen, die bei der Integration zu berücksichtigen sind. Auffallend hierbei ist, dass alle genannten Herausforderungen bereits bekannt sind. Das originäre Problem besteht daher in der Integration dieser Herausforderungen.

Der morphologische Kasten dient gleichzeitig als Grundlage, um unterschiedliche Integrationsszenarien ableiten zu können. Beispielsweise können automatisch generierte und manuell konstruierte Ausgangsmodelle rekonstruktiv integriert werden. Dabei kann es sich zudem um Ausgangsmodelle handeln, die jeweils mit Petri Netzen dargestellt wurden und die verbunden werden sollen, usw. Ein solches Szenario unterscheidet sich im Hinblick auf die damit verbundenen Herausforderungen bzw. möglicherweise auftretenden Konflikte grundsätzlich von einem Szenario indem die Ausgangsmodelle konstruktiv integriert werden sollen. Im letztgenannten Szenario können Konflikte durch vorausschauende Definition von Konventionen vermieden werden.

4 Verwandte Arbeiten

In der Literatur finden sich zahlreiche Ansätze, um den genannten Herausforderungen zu begegnen. Die meisten Beiträge stammen dabei aus den Themenbereichen der Prozessmodellintegration und der Prozessmodellierung. Aufsätze im Process Mining beschäftigen sich vor allem mit den Algorithmen zur Generierung der Prozessmodelle [TTM08, S.11; CW98; AAW05], jedoch nicht mit der Integration der Modelle. Da kein Ansatz den Integrationsprozess umfassend unterstützt, wird im Folgenden eine kurze Übersicht über Techniken der bestehenden Ansätzen gegeben, die zumindest teilweise Unterstützung bieten.

Als erste Herausforderung wurde die Transformation der Modellierungssprachen der Ausgangsmodelle (*Modellierungssprache* und *Process Mining Algorithmus*) aufgeführt. Gefunden werden Ansätze die direkt Modellierungssprachen auf graphischer Ebene transformieren [Jo07] oder Modellierungssprachen in ausführbare Sprachen übersetzen [Me08]. Hier von besonderem Interesse sind Ansätze, die EPK-Modelle in Petri-Netze (u. u.) überführen [De09; Lo09]. Solche sind auch im ProM Framework realisiert [Pr09a].

Die Verwendung unterschiedlicher Bezeichnungen *(Bezeichnung)* in den Ausgangsmodellen führt zu zahlreichen Konflikten, die insbesondere im Themenbereich der Qualitätssicherung der Modellierung behandelt werden. Eine Vielzahl von Ansätzen zeigen hier Möglichkeiten z. B. durch Vorgabe von Konventionen auf, um solche Konflikte zu vermeiden [Sc98, S. 189-194; Ro96, S. 187-224]. Darüber hinaus lassen sich Ansätze im Bereich der Schemaintegration [BLN86] finden. Auch dort werden Vorschläge entwickelt, die sich jedoch auf die Integration von Datenmodellen beziehen und deshalb zunächst noch auf die Problemstellung übertragen werden müssen.

Die Angleichung von Modellen, die mit unterschiedlichem *Prinzip der Modellierung* erstellt wurden, wird in der Literatur nicht behandelt.

Demgegenüber lassen sich leichter Vorschläge für die Angleichung von Modellen mit unterschiedlichem *Detaillierungsgrad* finden. Die Vorschläge sind zwar mit der Intention entstanden, dem Modellierer bei der Detaillierung seiner Modelle zu unterstützen. Die Zerlegungsregeln von Ferstl und Sinz [FS95, S. 203-204] oder die Kriterien zur Ableitung von Aufgaben aus Prozessen von Österle [Ös95, S. 86-94] lassen sich jedoch auch für die möglicherweise notwendige Abstraktion oder Detaillierung von Prozessmodellen verwenden.

Das *Vorgehen der Integration* manuell konstruierter und automatisch generierter Modelle *(Art der Ausgangsmodelle)* wird in der Literatur bislang noch nicht behandelt. Zu finden sind Ansätze, die konstruktiv und rekonstruktiv manuell konstruierte Prozessmodelle integrieren und dabei auch die *Art der Integration* festlegen: [Pr95, S. 257-259; Ro96, S. 255-267; SL05, S. 170-171; Ku06, S. 138]

Neben diesen der Problemstellung sehr nahe kommenden Ansätzen können weitere Anhaltspunkte zur *Richtung der Integration* und insbesondere zur vertikalen Integration (*Prinzip der vertikalen Integration)* aus folgenden Ansätzen entnommen werden: [Be07, S. 267; S. 271; Ma99, S. 429-431; S. 435-436; Br03, S. 269-312; He09, S. 85-86].

Zusammenfassend lässt sich feststellen, dass es in der Literatur eine Vielzahl von Ansätzen gibt, die bei der Bewältigung einzelner Herausforderungen unterstützen können. Zudem finden sich Ansätze, die auch die Integration von Modellen fokussieren. Jedoch deckt kein Ansatz alle genannten Herausforderungen ab und kein Ansatz behandelt die Integration manuell konstruierter und automatisch generierter Modelle. Aufgrund der Vielzahl der Ausprägungen, die im morphologischen Kasten abgebildet sind, erscheint es nicht erstrebenswert einen Ansatz zu entwickeln, der alle Herausforderungen bewältigt. Viel sinnvoller ist hier für relevante, mit dem morphologischen Kasten abgrenzbare Szenarien je einen adäquaten Ansatz zu entwickeln. Die Auswahl der relevanten Szenarien kann sich an der Bedeutung der Szenarien für die Praxis orientieren.

5 Fallbeispiel

Das Fallbeispiel stellt einen Ansatz zur Integration vor, der für ein bestimmtes Szenario entwickelt wurde. Der Ansatz wird an mehreren Beispielen nach der Design Science Forschungsmethode [Ta90, S. 42-43; He04, S. 89] erarbeitet. Dazu werden aus den oben genannten Herausforderungen jeweils Anforderungen für ein Integrationsszenario abgeleitet. Es wird ein Ansatz zur Lösung entworfen, der, soweit möglich, bestehende Arbeiten integriert. Anschließend wird der Ansatz in der Praxis eingesetzt. Auch wenn die folgenden Ausführungen sich nur auf den Lieferantenwechselprozess bei einem Unternehmen in der Energiewirtschaft beziehen, erfolgt die Anwendung an weiteren Fallbeispielen, um den Ansatz zu evaluieren.

Das Unternehmen bietet Endkunden u. a. die Unterstützung beim Wechsel zu einem neuen Energielieferanten an. Zum Nachweis der Einhaltung gesetzlicher Vorgaben [Bu06, S.10] werden regelmäßig Ist-Prozessmodelle manuell konstruiert, die u. a. als

EPKs dargestellt sind. Da auch Eventlogs für einen Prozessteil vorliegen, ergibt sich die Möglichkeit der automatischen Generierung. Durch Integration generierter Modelle und bestehender EPKs wird ein positiver Effekt auf die Wirtschaftlichkeit der Modellierung erwartet.

Die folgenden Anforderungen beschreiben das Szenario:

- Das zu entwickelnde Integrationsvorgehen muss für die Modellierungssprachen EPK und heuristisches Netz geeignet sein.
- Durch die Verwendung des Genetischen Process Mining Algorithmus [AWA07], der im ProM Framework realisiert ist, sind die Anforderungen der Transformation eines heuristischen Netzes in eine EPK und die Behandlung fiktiver Start-, Auslöse- und Endereignissen sowie fiktiver Start- und Endfunktion zu berücksichtigen.
- Die Abgrenzung der Ausgangsmodelle erfolgt auf Basis verfügbarer Eventlogs.
- Es sind Namens-, Typ- und Strukturkonflikte sowie Konflikte mit unterschiedlichen Detaillierungsgraden in den Ausgangsmodellen zu bewältigen.
- Die Prozessmodelle liegen vor, so dass von einer rekonstruktiven Integration auszugehen ist.
- Da gesetzliche Vorgaben keine alternativen Prozessabläufe erlauben, kann die Beziehung der Ausgangsmodelle als zeitlich nachgelagert und damit als Sequenz definiert werden. Die Prozessteile werden entsprechend verbunden.
- Weitere Anforderungen an die Richtung der Integration (horizontal) und die Art der Ausgangsmodelle (manuell konstruiert und automatisch generiert) ergeben sich offensichtlich aus dem Szenario.

Für dieses Szenario, welches aufgrund der Verbreitung der EPK [Pe09, S.10] eine hohe Praxisrelevanz besitzt, wurde ein erster Integrationsansatz entwickelt. Dieser kann auf Arbeiten von Rosemann zur Strukturintegration von Prozessmodellen [Ro96, S. 153-275] aufbauen. Dort werden u. a. jedoch nicht die Anforderungen behandelt, die sich aus der Integration automatisch generierter und manuell konstruierter Ausgangsmodelle, aus der Verwendung des Genetischen Process Mining Algorithmus, oder aus der Transformation eines heuristischen Netzes in eine EPK ergeben.

Das Vorgehen der horizontalen Integration für dieses Szenario wird im Folgenden an zwei bereits vorliegenden Prozessteilen dargestellt. Der Prozessteil „Akquisition durchführen“ wurde manuell als EPK konstruiert (siehe linker Teil der Abbildung 1). Der Prozessteil „Lieferantenwechsel ausführen“ wurde, aufgrund seiner Fähigkeit nicht triviale Konstrukte zu generieren, mit dem Genetischen Process Mining Algorithmus, unter Verwendung des ProM Frameworks, generiert. Aus Geheimhaltungsgründen wurden die zu Grunde liegenden Daten gefiltert und vereinfacht. Daneben wurde den verwendeten 6090 Prozessinstanzen eine fiktive Endfunktion hinzugefügt, da eine einzige Endfunktion die Generierung erleichtert. Eine fiktive Startfunktion wird nicht benötigt, da jede Prozessinstanz mit der Funktion „Geschäftsvorfall aufbauen“ beginnt. Das generierte

Prozessmodell erreicht nach 750 Populationen einen Fitness-Wert [AWA07, S. 263] von 0,98289 und liegt als heuristisches Netz vor (siehe rechter Teil der Abbildung 1). Dieses könnte nun zur Identifikation von Verbesserungspotenzial verwendet werden, was allerdings nicht im Fokus dieses Beitrags ist.

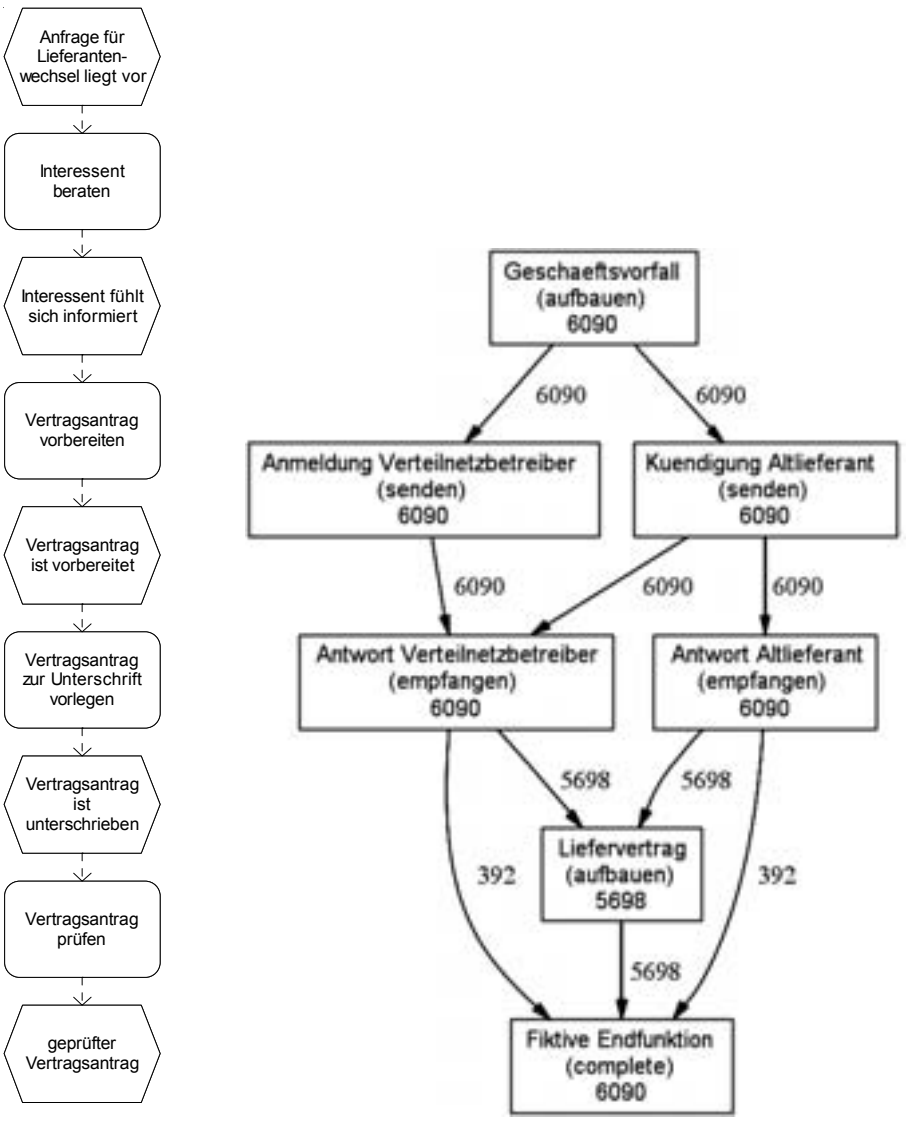


Abb. 1: Ausgangsmodelle als EPK und heuristisches Netz

Wie in der manuell konstruierten EPK (siehe linker Teil der Abbildung 1) zu erkennen ist, wurden bei der Bezeichnung der Ereignisse vorhandene Konventionen nicht voll-

ständig eingehalten. So sind die Ereignisse „Interessent fühlt sich informiert“ und „geprüfter Vertragsantrag“ nicht in der Form Objekt + sein + Verb im Partizip Perfekt. Ebenso ist zu erkennen, dass der Detaillierungsgrad der Ausgangsmodelle voneinander abweicht. Die Abgrenzung der Prozessteile erfolgte disjunkt und nach verfügbaren Eventlogs. Somit bilden die Prozessteile eine Sequenz, da gesetzliche Vorgaben keine andere Beziehung erlauben.

Der in diesem Beitrag vorgeschlagene Ansatz integriert die zwei Ausgangsmodelle horizontal mit der Zielsprache EPK. Als erste Aktivität erfolgt dabei die Transformation des heuristischen Netzes in eine EPK mit anschließender Überprüfung der Transformation. Für die Transformation kann das in ProM verfügbare Transformations-Plug-in „Heuristic net to EPC“ [Do05, S. 449; Pr09a] verwendet werden. Für die Überprüfung der Transformation wurde eine neue Technik entwickelt. Dabei wird die transformierte EPK zunächst im AML-Format aus dem ProM Framework exportiert und anschließend in das Aris Toolset importiert. Es folgt eine automatisierte grafische Aufbereitung mit dem Layoutassistenten. Weiterhin vorhandene Verletzungen von Layoutkonventionen, wie z. B. Kantenüberschneidungen, werden manuell aufgelöst.

Die zweite Aktivität stellt die Konfliktbehandlung dar. Hierbei sind Namens-, Typ- und Strukturkonflikte, Konflikte mit abweichenden Detaillierungsgraden sowie mit fiktiven Ereignissen und Funkionen aufzulösen. Eine bestimmte Ordnung der Konfliktbehandlung kann nicht festgelegt werden, da sich diese gegenseitig beeinflussen und deshalb von konkreten Ausgangsmodellen abhängig sind. Die Behandlung von Namens-, Typ- und Strukturkonflikten kann von Rosemann [Ro96, S.187-224] übernommen werden. So werden die oben genannten Ereignisse entsprechend den Namenskonventionen in „Interessent ist informiert“ und „Vertragsantrag ist geprüft“ geändert.

Weitere Techniken zur Konfliktbehandlung wurden selbst entwickelt. So können abweichende Detaillierungsgrade bspw. dadurch aufgelöst werden, dass die manuell konstruierte EPK auf die Detaillierungsstufe der transformierten EPK gebracht wird. Im Fallbeispiel ergibt sich eine EPK mit der Funktion „Akquisition durchführen“ und zugehörigem Start- und Endereignis.

Für die Behandlung der fiktiven Endfunktion und der fiktiven Ereignissen in der transformierten EPK wird die vollständige Entfernung und anschließende, für die EPK-Syntax notwendige, Ergänzung semantisch sinnvoller Ereignissen vorgeschlagen. Es ist darauf zu achten, dass als Ergebnis der Konfliktbehandlung zwei sequentielle Prozessteile mit lexikalisch übereinstimmenden End- und Startereignis entstehen, welches aus der manuell konstruierten EPK übernommen wird („Vertragsantrag ist geprüft“).

Anschließend folgt als dritte Aktivität die Integration im Sinne des Verbindens, welche an Rosemann [Ro96, S. 257] angelehnt wird. Aufgrund zeitlich nachgelagerter Prozessteile und der lexikalischen Identität von End- und Startereignissen können die Teilmodelle durch eine sequentielle Verbindung mittels Prozesswegweiser integriert werden.

Als abschließende Aktivität vier wird das Zielmodell überprüft und ggf. modifiziert, damit beispielsweise festgelegte Konventionen zur Sicherstellung der Qualität des Mo-

dells eingehalten werden. Maßnahmen hierzu können von Rosemann [Ro96, S. 274-275] übernommen werden.

Schließlich liegt das Ergebnis der rekonstruktiven Integration (siehe Abbildung 2) vor. Die beiden Prozessteile sind als EPK auf dem gleichen Abstraktionsniveau modelliert. Aufgetretene Konflikte sind aufgelöst. Die Prozessteile haben ein lexikalisch übereinstimmendes End- bzw. Startereignis und sind über Prozesswegweiser miteinander integriert.

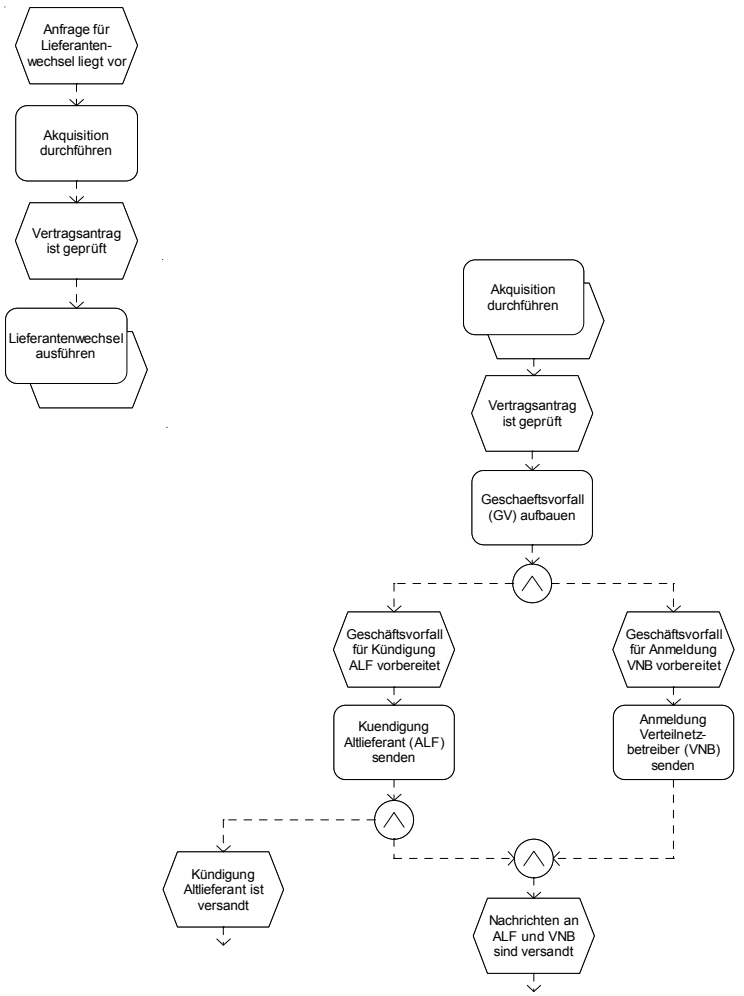


Abb. 2: Integriertes Zielmodell

6 Schlussbetrachtung

Der vorliegende Beitrag greift die Modellierung der Prozesse im Ist-Zustand als Themenstellungen heraus. Dabei wird von der These ausgegangen, dass durch Integration manuell konstruierter und automatisch generierter Modelle die Korrektheit, Aktualität und Vollständigkeit des Prozessmodells erhöht werden und der Erstellungs- bzw. Aktualisierungsaufwand reduziert werden kann. Die Schwerpunkte des Beitrags liegen darin, den Integrationsvorgang zu analysieren, Einflussfaktoren zu identifizieren, Ansatzpunkte zur systematischen Durchführung der Integration zu finden und einen ersten Integrationsansatz zu entwickeln. Die identifizierten Einflussfaktoren werden in einem morphologischen Kasten zusammengetragen. Die Untersuchung zeigte, dass die Integration der Modelle aufgrund der vielen Einflussfaktoren ein komplexer Vorgang ist und dass unterschiedliche Integrationsszenarios differenziert werden müssen. Der morphologische Kasten bildet eine strukturierte Grundlage zur Abgrenzung dieser Szenarios. Die Analyse bestehender Ansätze zeigte, dass keine der untersuchten Methoden die Problemstellung vollständig lösen kann, sich jedoch einzelne Techniken der bestehenden Methoden eignen, um Teilprobleme zu lösen und damit wiederverwendet werden können. Für ein ausgewähltes Integrationsszenario wurde ein erster Ansatz entwickelt und am Beispiel des Lieferantenwechselprozesses eines Unternehmens in der Energiewirtschaft durchgeführt. Der Ansatz wird aktuell an mehreren Anwendungsfällen validiert. Besondere Herausforderungen ergeben sich dabei insbesondere bei der Transformation der Modellierungssprachen und der Konfliktbehandlung bei unterschiedlichen Bezeichnungen.

Neben der weiteren Validierung des vorgestellten Ansatzes werden zukünftige Arbeiten vor allem in der Abgrenzung der Szenarien sowie der Entwicklung von szenariobasierten Integrationsmethoden gesehen. Die unterschiedlichen Pfade durch den morphologischen Kasten können hier als gute Grundlage dienen. Zudem sollte die Frage, für welche Bereiche einer Prozessarchitektur eher manuell konstruierte oder eher automatisch generierte Prozessmodelle zu erstellen sind, näher untersucht werden und Auswahlkriterien bereitgestellt werden.

Eine weitere künftige Aufgabe besteht darin, die Automatisierbarkeit des entwickelten Ansatzes zu prüfen. Um den Erstellungs- und Aktualisierungsaufwand zu reduzieren, sollte die Integration zumindest in Teilen automatisiert werden. Erste Schritte werden hierzu bereits durch die Entwicklung des Ansatzes geleistet, der den Integrationsvorgang strukturiert und einzelne Integrationsschritte (soweit möglich) formalisiert.

Literaturverzeichnis

[Aa05] Aalst, W.M.P. van der: Business Alignment: Using Process Mining as a Tool for Delta Analysis and Conformance Testing. In: Requirements Engineering Journal, 10, 3, 2005; S. 198-211.

[Aa07] Aalst, W.M.P. van der: Trends in Business Process Analysis - From Verification to Process Mining. In (Cardoso, J. et al. Hrsg): Proceedings of the 9th International Con-

ference on Enterprise Information Systems (ICEIS 2007), Medeira, Portugal, 2007, 12-22.

[AAW05] Aalst, W.M.P. van der et al.: Genetic Process Mining. In (Ciardo, G., Darondeau, P. Hrsg): 26th International Conference on Applications and Theory of Petri Nets (ICATPN 2005), LNCS 3536, 2005, 48-69.

[Aa03a] Aalst, W.M.P. van der et al.: Workflow Patterns In: Distributed and Parallel Databases, 14, 1, 2003; S. 5-51.

[Aa03b] Aalst, W.M.P. van der et al.: Workflow mining: a survey of issues and approaches. In: Data & Knowledge Engineering, 47, 2, 2003; S. 237-267.

[AW04] Aalst, W.M.P. van der; Weijters, A.J.M.M.: Process Mining: A Research Agenda. In: Computers in Industry, 53, 3, 2004; S. 231-244.

[ARW08] Aier, S. et al.: Unternehmensarchitektur – Literaturüberblick und Stand der Praxis. In: Wirtschaftsinformatik, 50, 4, 2008; S. 292-304.

[AWA07] Alves de Medeiros, A.K. et al.: Genetic Process Mining: An Experimental Evaluation. In: Data Mining and Knowledge Discovery, 14, 2, 2007; S. 245-304.

[BLN86] Batini, C. et al.: A Comparative Analysis of Methodologies for Database Schema Integration. In: ACM Computing Surveys, 18, 4, 1986; S. 323-364.

[Be07] Becker, J. et al.: Bausteinbasierte Modellierung von Prozesslandschaften mit der PICTURE-Methode am Beispiel der Universitätsverwaltung Münster. In: Wirtschaftsinformatik, 49, 4, 2007; S. 267-279.

[BK05] Becker, J.; Kahn, D.: Der Prozess im Fokus. In (Becker, J. et al. Hrsg.): Prozessmanagement - Ein Leitfaden zur prozessorientierten Organisationsgestaltung, 5. Auflage, Springer, Berlin et al. 2005; S. 3-16.

[Bu06] Bundesnetzagentur: Beschluss BK6-06-009. 2006, http://www.bundesnetzagentur.de/media/archive/10893.pdf (Zugriff: 22.09.2008).

[Br03] Brocke, J. vom: Referenzmodellierung. Gestaltung und Verteilung von Konstruktionsprozessen. Logos, Berlin, 2003.

[CW98] Cook, J.E.; Wolf, A.L.: Discovering models of software processes from event-based data. In: ACM Transactions on Software Engineering and Methodology, 7, 3, 1998; S. 215-249.

[De09] Dehnert, J.: Making epcs fit for workflow management. In: EMISA FORUM, 23, 1, 2009; S. 12-26.

[Do05] Dongen, B.F. van et al.: The ProM framework: a new era in process mining tool support. In (Ciardo, G., Darondeau, P. Hrsg.): 26th International Conference on Applications and Theory of Petri Nets (ICATPN 2005), LNCS 3536, Springer Heidelberg 2005; S. 444-454.

[FS95] Ferstl, O.K.; Sinz, E.J.: Der Ansatz des Semantischen Objektmodells (SOM) zur Modellierung von Geschäftsprozessen. In: Wirtschaftsinformatik, 37, 3, 1995; S. 209-220.

[Fr94] Frank, U.: Multiperspektivische Unternehmensmodellierung: theoretischer Hintergrund und Entwurf einer objektorientierten Entwicklungsumgebung. Oldenburg, München, 1994.

[He09] Heinrich, B. et al.: The process map as an instrument to standardize processes: design and application at a financial service provider. In: Information Systems and E-Business Management, 7, 1, 2009; S. 81-102.

[He04] Hevner, A.R. et al.: Design Science In Information Systems Research. In: MIS Quarterly, 28, 1, 2004; S. 75-105.

[Jo07] Johannsen, F.: Transformation von Prozessmodellen: Bewertung XML-basierter Ansätze. Salzwasser-Verlag, Bremen, 2007.

[Ju06] Jung, R.: Architekturen zur Datenintegration - Gestaltungsempfehlungen auf der Basis fachkonzeptueller Anforderungen. Dt. Univ.-Verl., Wiesbaden, 2006.

[KNS92] Keller, G. et al.: Semantische Prozeßmodellierung auf der Grundlage „Ereignisgesteuerter Prozeßketten (EPK)“. In (Scheer, A.-W. Hrsg.): Veröffentlichungen des Instituts für Wirtschaftsinformatik - Heft 89, Saarbrücken 1992; S.

[Kü05] Küng, P. et al.: Business Process Monitoring & Measurement in a Large Bank: Challenges and selected Approaches. Proceedings of the 16th International Workshop on Database and Expert Systems Applications (DEXA’05), Copenhagen, 2005.

[Ku06] Kupsch, F.: Framework zur dezentralen Integration systemübergreifender Geschäftsprozesse. Eul, Lohmar, 2006.

[Le06] Leist-Galanos, S.: Methoden zur Unternehmensmodellierung - Vergleich, Anwendungen und Diskussion der Integrationspotentiale. Logos, Berlin, 2006.

[Lo09] Lohmann, N. et al.: Petri Net Transformations for Business Processes - A Survey. In: Transactions on Petri Nets and Other Models of Concurrency II, LNCS Volume 5460, 2009; S. 46-63.

[Ma99] Malone, T.W. et al.: Tools for Inventing Organizations: Toward a Handbook of Organizational Processes. In: Management Science, 45, 3, 1999; S. 425-443.

[Me08] Mendling, J.: Metrics for Process Models: Empirical Foundations of Verification, Error Prediction, and Guidelines for Correctness. Springer, 2008.

[Me07] Mertens, P.: Integrierte Informationsverarbeitung 1 - Operative Systeme in der Industrie 16., überarbeitete Auflage, Gabler, Wiesbaden, 2007.

[Ös95] Österle, H.: Business Engineering - Prozeß- und Systementwicklung. 2., verbesserte Auflage, Springer, Berlin et al., 1995.

[Pe09] Peyret, H. et al.: The Forrester Wave: Business Process Analysis, EA Tools, And IT Planning, Q1 2009. Forrester Research, Inc., Cambridge, 2009.

[Pr95] Priemer, J.: Entscheidungen über die Einsetzbarkeit von Software anhand formaler Modelle. Pro Universitate, Sinzheim, 1995.

[Pr09a] ProM: Conversion Plug-ins contained in ProM. 2009, http://is.tm.tue.nl/trac/prom/query?component=Management&milestone=Conversion&order=priority (Zugriff: 06.10.09).

[Pr09b] ProM: Mining Plug-ins contained in ProM. 2009, http://is.tm.tue.nl/trac/prom/query?component=Management&milestone=Mining&order=priority (Zugriff: 2009-08-20).

[Re09] Recker, J. et al.: Business Process Modeling - A Comparative Analysis. In: Journal of the Association for Information Systemes, 10, 2, 2009; S. 333-363.

[Ro96] Rosemann, M.: Komplexitätsmanagement in Prozeßmodellen - Methodenspezifische Gestaltungsempfehlungen für die Informationsmodellierung. Gabler, Wiesbaden, 1996.

[Ro99] Rosemann, M.: Gegenstand und Aufgaben des Integrationsmanagements. In (Scheer, A.-W. et al. Hrsg.): Integrationsmanagement - Arbeitsbericht des Instituts für Wirtschaftsinformatik Nr.65, Münster 1999; S. 5-18.

[RSD05] Rosemann, M. et al.: Vorbereitung der Prozessmodellierung. In (Becker, J. et al. Hrsg.): Prozessmanagement - Ein Leitfaden zur prozessorientierten Organisationsgestaltung, 5. Auflage, Springer, Berlin et al. 2005; S. 45-103

[Sc98] Schütte: Grundsätze ordnungsmässiger Referenzmodellierung: Konstruktion konfigurations- und anpassungsorientierter Modelle. Gabler, Wiesbaden, 1998.

[SL05] Schwegmann, A.; Laske, M.: Istmodellierung und Istanalyse. In (Becker, Jörg et al. Hrsg.): Prozessmanagement - Ein Leitfaden zur prozessorientierten Organisationsgestaltung, 5. Auflage, Springer, Berlin et al. 2005; S. 155-184.

[Ta90] Takeda, H. et al.: Modeling Design Processes In: AI Magazine, 11, 4, 1990; S. 37-48.

[TTM08] Tiwari, A. et al.: A review of business process mining - state-of-the-art and future trends. In: Business Process Management Journal, 14, 1, 2008; S. 5-22.

Erhöhte Abbildungstreue von Geschäftsprozessmodellen durch Kontextsensitivität

Daniel Wagner[1] und Otto K. Ferstl[2]

Abstract: Individuen sind im alltäglichen Leben externen Einflüssen ausgesetzt und reagieren unmittelbar auf diese Einflüsse oder passen nur ihr weiteres Verhalten an die jeweils neue Situation an. In gleicher Weise wirken auf Unternehmen Einflüsse, die eine direkte Reaktion oder nur eine Anpassung ihrer Geschäftsprozesse und IT-Systeme erfordern. Geschäftsprozesse unterliegen somit grundsätzlich zwei Arten von Einflussgrößen: primäre, vorhersehbare Einflussgrößen wie z. B. ankommende Aufträge, die eine unmittelbare Reaktion der Prozesse bewirken, oder sekundäre Einflüsse, im Folgenden „Kontext" genannt, die eine Verhaltensänderung bzw. Anpassung der Prozesse bewirken. Herkömmliche Modelle von Geschäftsprozessen erfassen häufig nur primäre Einflussgrößen. Die zusätzliche Erfassung sekundärer Einflussgrößen gewinnt aufgrund der steigenden Komplexität und Dynamik von Geschäftsprozessen an Bedeutung. Geschäftsprozessmodelle müssen als Modell der Realität präziser werden, d. h. eine erhöhte Abbildungstreue erreichen. Entsprechendes gilt für die Geschäftsprozessmodelle unterstützenden Anwendungssysteme. Der Beitrag zeigt einen Ansatz zur Modellierung kontextsensitiver Geschäftsprozesse und zur Spezifikation kontextsensitiver Anwendungssysteme. Die methodische Grundlage bildet das Semantische Objektmodell (SOM).

1 Einleitung

Geschäftsprozessmodelle (GP-Modelle) erfassen die Geschäftsprozesse der realen Welt gemäß den Vorgaben der Modellierungsziele. Formalziele, die bei der Modellierung eines Ausschnittes der Realität verfolgt werden, sind z. B. Konstruktionsadäquanz, Wirtschaftlichkeit, systematischer Aufbau, Klarheit oder Vergleichbarkeit. Der Fokus dieses Beitrages liegt darauf, das Ziel der Richtigkeit von Modellen durch Erhöhung der Abbildungstreue zu unterstützen (Sprachadäquanz) [Sc98, S. 111 ff.]. Mittel zur Erhöhung der Abbildungstreue ist die Einbeziehung von Kontextinformationen in die Modellierung. Herkömmliche Geschäftsprozessmodelle ohne Berücksichtigung von Kontext erfassen primäre Einflussgrößen und Verhaltensmuster realer Geschäftsprozesse und abstrahieren von weiteren sekundären Einflussgrößen und Merkmalen. Zum einen, um die Komplexität der Modelle zu begrenzen, und zum anderen aufgrund der Tatsache, dass die Modellierungssprachen für Geschäftsprozessmodelle (Metamodelle) in vielen Fällen keine Modellierungselemente für sekundäre Merkmale vorsehen. Primäre Einflussgrößen wie z. B. ankommende Aufträge bewirken unmittelbare Reaktionen der Prozesse, die Geschäftsprozessmodelle beschreiben dann den Ablauf der Reaktionen. Sekundäre Einflüsse, im Folgenden als „Kontext" bezeichnet, beeinflussen den Ablauf solcher Reaktionen,

[1] forFLEX – Dienstorientierte IT-Systeme für hochflexible Geschäftsprozesse, Feldkirchenstraße 21, 96045 Bamberg, daniel.wagner@uni-bamberg.de
[2] forFLEX – Dienstorientierte IT-Systeme für hochflexible Geschäftsprozesse, Feldkirchenstraße 21, 96045 Bamberg, otto.ferstl@uni-bamberg.de

d. h. führen zu Verhaltensänderungen bzw. Anpassungen der Prozesse, aber bewirken keine unmittelbare Reaktion.

Die Einbeziehung von Kontext hat zum Ziel, die Präzision der Modellierung zu erhöhen. Dies gilt insbesondere für Geschäftsprozesse, die eine hohe Struktur- und Verhaltensflexibilität benötigen. Deren Verhaltens- und Strukturvielfalt wird in herkömmlichen Geschäftsprozessmodellen häufig nicht ausreichend berücksichtigt, d. h. der Flexibilitätsbedarf wird nicht sichtbar. Kontextsensitive Geschäftsprozessmodelle ermöglichen, den Flexibilitätsbedarf durch Erfassung der Verhaltens- und Strukturvielfalt darzustellen und eignen sich auch zur Modellierung hochflexibler Geschäftsprozesse (hGP)[3] [Pü09].

Forschungsbedarf besteht hinsichtlich einer Methodik zur Modellierung von Kontext in Geschäftsprozessmodellen und Anwendungssystemen (AwS). Im vorliegenden Beitrag wird auf Grundlage der Methodik des Semantischen Objektmodells (SOM-Methodik, [FS08, S. 197 ff.]) ein Ansatz zur Modellierung von Kontext in Geschäftsprozessmodellen dargestellt, der Systementwickler in die Lage versetzt, die Präzision bei der Abbildung der Realität in Geschäftsprozessmodellen und der anschließenden Anwendungssystemspezifikation zu erhöhen. Hierzu zeigt Abschnitt 2 die Notwendigkeit der Berücksichtigung von Kontext auf. Abschnitt 3 diskutiert Arbeiten zu diesem Themenkomplex. Die Modellierung von Kontext in GP-Modellen zeigt Abschnitt 4 auf. Abschnitt 5 beschreibt eine Kontextmodellierungssystematik, welche die wesentlichen Fälle der Kontextsensitivierung von GP-Modellen umfasst. Die Anreicherung der Anwendungssystemspezifikation um Kontextinformationen zeigt Abschnitt 6. Einen zusammenfassenden Ausblick gibt Abschnitt 7.

2 Notwendigkeit der Berücksichtigung von Kontext in der Geschäftsprozessmodellierung

Ein Modell ist ein System, welches ein anderes System zielorientiert abbildet und dabei eine struktur- und verhaltenstreue Abbildung realisiert [FS08, S. 22 ff.]. Eine zentrale Aufgabe eines Modells ist es, ein „adäquates Abbild der betrachteten Wirklichkeit“ [Ko61, S. 321] zu schaffen. Je nach Modellierungs- und Untersuchungsziel werden die als relevant betrachteten Ausschnitte der Realität im Modell repräsentiert. Dies führt häufig dazu, dass die Abgrenzung zu eng erfolgt und somit der Umwelt realer Systeme, die häufig nicht zu vernachlässigende Einflussfaktoren beinhaltet, ein zu geringer Stellenwert beigemessen wird. Die von KOSIOL geforderte adäquate Abbildung der betrachteten Wirklichkeit ist häufig nicht gegeben. Dies führt zu einer suboptimalen Abbildungstreue und Generierung von Mehrdeutigkeiten im GP-Modell. Es gilt also den „Ausschnitt der Wirklichkeit“ [Di73] möglichst so zu wählen, dass nicht nur die offensichtlichen primären Einflussfaktoren auf Komponenten des Systems berücksichtigt

[3] hGP weisen im Gegensatz zu herkömmlichen Geschäftsprozessen mindestens eines der folgenden Merkmale auf: (1) Unvollständige Planbarkeit vor Ausführung des GP, (2) Überlappung von Planung und Ausführung des GP, sowie (3) Kontextsensitivität des GP. Solche GP treten bspw. im Gesundheitswesen bei der Behandlung eines Patienten mit unklarer Diagnose oder beim Bau industrieller Anlagen auf.

werden, sondern auch sekundäre Einflussfaktoren, die hier als „Kontext“ von Geschäftsprozessen bezeichnet werden. In der um Kontextmodellierungselemente erweiterten SOM-GP-Modellierung dienen Kontextinformationen dazu, mehrdeutige Situationen aufzulösen und in eindeutige Situationen überzuführen. In systemtheoretischer Sprechweise liegt eine mehrdeutige Situation vor, wenn die Beziehung zwischen einer unabhängigen Größe X und einer abhängigen Y, die allgemein in einer Relation $R_{XY} \subseteq X \times Y$ beschrieben wird, nicht funktional oder eindeutig ist, d. h. nicht in der Form $f: X \rightarrow Y$ dargestellt werden kann. Durch die Hinzunahme eines Parameters K kann eine solche Relation in eine Funktion $f_a: X \times K \rightarrow Y$ transformiert werden [Fe79, S. 12 ff.]. K wird hier als Kontext der Beziehung interpretiert. In kontextsensitiven Geschäftsprozessmodellen wird die Parametermenge K durch Kontextfaktoren repräsentiert.

3 Verwandte Arbeiten

Verschiedene Bereiche der (Wirtschafts-) Informatik untersuchen seit etwa 15 Jahren kontextsensitive Systeme[4]. In der Literatur liegen demzufolge zahlreiche Definitionen von Kontext in Bezug auf die Entwicklung von IT-Systemen vor. Siehe hierzu [Sc94], [Br96], [Br97], [WJH97], [FF98], [Ro98], [Ab99], [RPM99] oder [Ja01]. Es ist bemerkenswert, dass eine eindeutige Definition des Kontextbegriffes schwer fällt. Dies resultiert im Wesentlichen daraus, dass diese Definitionen sehr von der jeweiligen Domäne, also vom Kontext des Kontextes, abhängen [Mc87]. Die Motivation für die Kontextbetrachtung ist vielschichtig, lässt sich aber stets auf die Auflösung oder Beherrschung mehrdeutiger Situationen zurückführen. Nachfolgend werden einige dieser Bereiche exemplarisch dargestellt. Das Information Retrieval verwendet Kontextinformationen im Wesentlichen dazu, die Ermittlung des Informationsbedarfes zu präzisieren und daraus das Informationsbedürfnis eines Anwenders abzuleiten. Verwendete Informationen aus dem Kontext sind beispielsweise die Rolle des Informationssuchenden oder dessen aktuelle Aufgabe [Mo06, S. 4], [Is07, S. 2]. In den Bereichen Ubiquitous und Mobile Computing wird durch das Auswerten zur Verfügung stehender Kontextinformationen auf Ziele und Aktivitäten des Anwenders geschlossen, um ihn zu jeder Zeit und an jedem Ort mit relevanten Informationen zu versorgen [Co05]. Dies soll durch kontextbasierte Anpassung mobiler Terminals und Anwendungen erreicht werden [BC04]. Ein Unterziel ist hierbei die Minimierung der Benutzereingaben durch Berücksichtigung von Kontextinformationen [SLF03, S. 1]. Im Business Process Management ist die Betrachtung von Kontext unter anderem den steigenden Flexibilitätsanforderungen von Geschäftsprozessen geschuldet. Wenige Autoren haben bislang Ansätze veröffentlicht, die die Bedeutung von Kontextinformationen und die Potenziale kontextsensitiver Geschäftsprozesse aufzeigen [RRF08], [SN07], [HBR09]. Bisher ausstehend ist ein methodenbasierter Ansatz zur Ergänzung von Geschäftsprozessmodellen um Kontextinformationen und die ent-

[4] Ausgenommen ist die Diskussion kontextfreier bzw. kontextsensitiver Grammatiken und Programmiersprachen, welche bereits vor Jahrzenten Gegenstand der Diskussion waren, und deren Grundzüge im Wesentlichen NOAM CHOMSKY zuzuschreiben sind. [Ba09, S. 162])

sprechende Propagierung dieser Informationen in die Anwendungssystemspezifikation. In der Unified Modeling Language (UML), in Ereignisgesteuerten Prozessketten (EPK) und der Business Process Modeling Notation (BPMN) können Kontextinformationen beispielsweise über textuelle Annotationen eingefügt werden [Pl09]. Sie werden jedoch weder als Teil der Modellierungsmethodik berücksichtigt, noch methodenbasiert vom GP-Modell in die AwS-Spezifikation propagiert.

4 Modellierung von Kontext in Geschäftsprozessmodellen

Zur Modellierung von Geschäftsprozessen wird im vorliegenden Beitrag die Methodik des Semantische Objektmodells verwendet. Die zentralen Elemente von SOM-GP-Modellen sind betriebliche Objekte und Transaktionen als Verbünde von eng bzw. lose gekoppelten Aufgaben sowie Zielbeziehungen. Zur Modellierung der Struktursicht auf Geschäftsprozesse wird das Interaktionsschema (IAS), zur Modellierung der Verhaltenssicht das Vorgangs-Ereignis-Schema (VES) verwendet [FS08, S. 195 ff.]. Im IAS wird das „Straßennetz“, welches der Ausführung des Prozesses zu Grunde liegt, modelliert. Die Verhaltenssicht, also der Prozessablauf, wird separat im VES modelliert. Anhand eines knappen Beispiels soll die GP-Modellierung veranschaulicht werden. Modelliert wird aus Struktur- (Abb. 1a) und Verhaltenssicht (Abb. 1b) der Geschäftsprozess „Warendistribution“, woran die betrieblichen Objekte *Großhändler* (Diskursweltobjekt) und *Kunde* (Umweltobjekt) beteiligt sind. Ein *Großhändler* bewirbt seine Produkte (A: Anbahnung), woraufhin ein *Kunde* einen Auftrag in Form einer *Bestellung* erteilt (V: Vereinbarung). Der *Großhändler* nimmt im Anschluss die *Lieferung* vor (D: Durchführung).

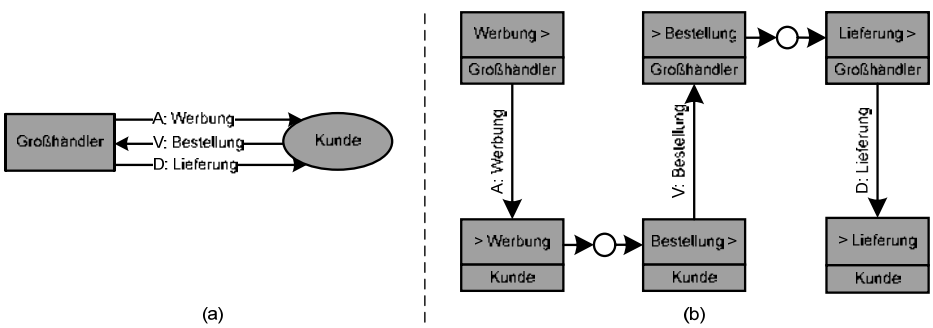


Abb. 1: Struktursicht und Verhaltenssicht des Geschäftsprozesses "Warendistribution"

Das VES visualisiert die Reihenfolgebeziehung der beteiligten Aufgaben. Hierbei liegt eine Petri-Netz-Semantik zu Grunde. Jede Transaktion umfasst eine sendende Aufgabe (Aufgabe>) und eine empfangende Aufgabe (>Aufgabe). Die Transaktion *A: Werbung* besteht beispielsweise aus der synchronen Durchführung der Aufgabe *Werbung*>, in der das betriebliche Objekt *Großhändler* Werbung versendet, und der Aufgabe *>Werbung*, in der *Kunde* die Werbung empfängt. Zwischen den Aufgaben innerhalb eines betrieblichen Objektes regeln Ereignisse die Reihenfolge der Aufgaben. Jede Aufgabe kann um

Vor- und Nachbedingungen in Form logischer Ausdrücke ergänzt werden. Vorbedingungen beschreiben, wann eine Aufgabe durchzuführen ist, Nachbedingungen steuern, falls eine Aufgabe mehrere Nachfolger hat, welche Nachfolger zu starten sind. Das hier dargelegte Beispiel berücksichtigt keine Kontextfaktoren; es ist demnach nicht kontextsensitiv. Dies entspricht jedoch nicht zwangsweise der Realität. So ist der modellierte Geschäftsprozess durchaus beeinflusst von Umweltbedingungen, deren Auswirkungen im gezeigten Modell nicht berücksichtigt sind. Es existieren Mehrdeutigkeiten beispielsweise dahingehend, ob ein Kunde persönlich im Ladenlokal oder per Webapplikation eine Bestellung aufgibt, oder auf welchem Wege das Unternehmen die bestellten Produkte versendet. Die Auflösung solcher Mehrdeutigkeiten kann durch die Einbeziehung von Kontext in die Modellierung erfolgen. Im Folgenden wird erläutert, wie das Semantische Objektmodell erweitert wird, um kontextsensitive Geschäftsprozesse modellieren zu können.

Die Modellierung von Kontext in SOM-GP-Modellen erfolgt über Annotationen von betrieblichen Objekten und Transaktionen. Hierfür wird die Kontextwolke als neues Modellierungselement eingeführt und mit diesen Modellierungselementen verbunden. Die geringfügige Erhöhung der Komplexität durch ein zusätzliches Modellierungselement kann akzeptiert werden, da das SOM-Metamodell nur sehr wenige Modellierungselemente vorsieht. Der Zugewinn an Abbildungstreue rechtfertigt die etwas höhere Komplexität der GP-Modelle.

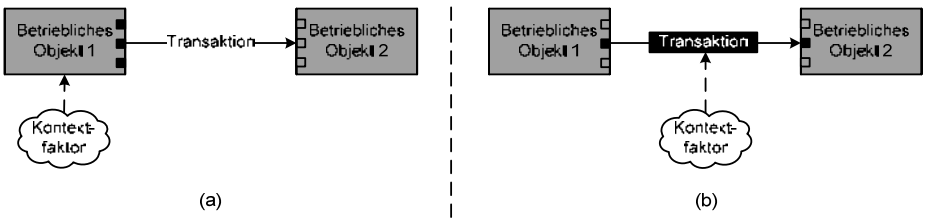


Abb. 2: Kontextsensitive Modellierungselemente

Als Kontext von betrieblichen Objekten oder Transaktionen werden Einflussgrößen aus deren Umgebung bezeichnet, die nicht bereits in Transaktionen modelliert sind. Ein Geschäftsprozess ist kontextsensitiv, wenn er zusätzlich zu den in Transaktionen modellierten primären Einflussgrößen weitere sekundäre GP-externe Einflussgrößen in seinem Verhalten und seiner Struktur berücksichtigt. Ein Anwendungssystem ist kontextsensitiv wenn es kontextsensitive Geschäftsprozesse unterstützt. Einflussfaktoren auf Geschäftsprozesse können somit alternativ primär als Transaktionen einschließlich Umweltobjekten oder sekundär als Kontext modelliert werden. Ob ein Einflussfaktor als Transaktion oder als Kontextfaktor modelliert wird, hängt davon ab, ob er eine Reaktion des Geschäftsprozesses initiiert oder den Ablauf und die Struktur des Prozesses nur beeinflusst. Zudem ist es bei der Modellierung als Transaktion nötig, den Sender des Einflusses genauer zu kennen, d. h. ihn mittels eines Objektes (Diskurs- oder Umweltobjekt) zu repräsentieren. Ist der Verursacher eines Einflusses unbekannt oder soll dieser absichtlich als unbekannt oder irrelevant modelliert werden, bietet sich die Modellierung in

Form eines Kontextfaktors an. Abb. 2 zeigt die Wirkung von Kontextfaktoren auf betriebliche Objekte (Abb. 2a) oder Transaktionen bzw. deren Aufgaben (Abb. 2b).

In einer kontextsensitiven Transaktion sind auch die zugehörige Sende- und Empfangsaufgabe kontextsensitiv. Um die einflussausübenden Kontextfaktoren darstellen und ihre Wirkungsweise modellieren zu können, wird die Symbolik des VES (Abb. 3a) erweitert. In den Vor- und Nachbedingungen wird auf die Kontextfaktoren einer Aufgabe Bezug genommen. Die Formulierung der Vor- und Nachbedingungen als logische Ausdrücke erfolgt in Pseudo-Code (z. B.: „Ort_des_Kunden = zu Hause“). Die Notation einer kontextsensitiven Aufgabe als Teil einer kontextsensitiven Transaktion zeigt Abb. 3b.

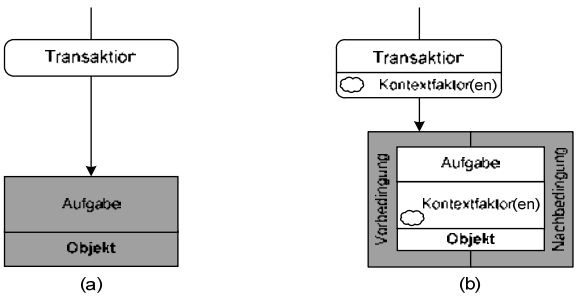


Abb. 3: Herkömmliche (a) und kontextsensitive (b) Aufgabe und Transaktion im VES

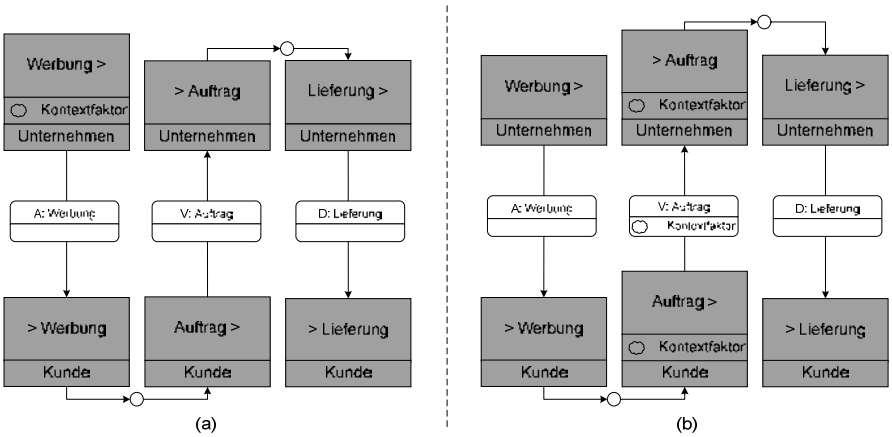


Abb. 4: Kontextsensitive Aufgabe (a) und kontextsensitive Transaktion (b) im VES

Im VES werden kontextsensitive Aufgaben eines betrieblichen Objektes um den/die entsprechenden Kontextfaktor(en) erweitert. Abb. 4 zeigt die beiden grundlegenden Modellierungsfälle. Abb. 4a zeigt einen Geschäftsprozess, dessen einziger kontextsensitiver Bestandteil die Aufgabe zur Erstellung eines Werbeangebotes (*Werbung>*) ist. Abb. 4b hingegen visualisiert eine kontextsensitive Transaktion. Während im vorherge-

henden Fall nur eine einzelne Aufgabe kontextsensitiv ist, sind bei kontextsensitiven Transaktionen grundsätzlich die Sende- und Empfangsaufgabe kontextsensitiv.

5 Kontextmodellierungssystematik

Zur Verdeutlichung der in Abschnitt 4 vorgeschlagenen Modellierungsmethodik für kontextsensitive GP werden im Folgenden wesentliche Fälle der Modellierung von Kontext präsentiert. Kontextsensitives Systemelement ist entweder ein betriebliches Objekt oder eine betriebliche Transaktion. Die Einwirkung des Kontextfaktors auf das Systemelement, d. h. die dadurch erfasste Mehrdeutigkeit einer Situation, wird mittels Variantenbildung oder Parametrisierung behandelt. Es werden folglich vier grundlegende Fälle unterschieden. Die Entscheidung wie ein Kontextfaktor behandelt wird, hängt zum einen von den zu erwartenden Kontextfaktorwerten, und zum anderen von den Modellierungszielen ab. Allgemein gilt, dass Parametrisierung immer dann empfehlenswert ist, wenn aus den Kontextfaktorwerten zu viele Varianten resultieren, um sie einzeln im Modell darzustellen. Abb. 5 zeigt die vier resultierenden Fälle der Kontextmodellierung. Sie werden im Folgenden mithilfe von Beispielen dargestellt.

		Kontextsensitives Systemelement	
		Betriebliches Objekt	Betriebliche Transaktion
Methode der Kontextbehandlung	Variantenbildung	Fall 1: Kontextsensitiver User-Support	Fall 3: Kontextsensitive Distribution
	Parametrisierung	Fall 2: Kontextsensitive Angebotserstellung	Fall 4: Kontextsensitives Multikanalbanking

Abb. 5: Kontextmodellierungssystematik

Fall 1: Kontextsensitiver User-Support

Abb. 6a und 6b beschreiben die Interaktion zwischen einem *User* und einem *Support-Provider* aus Struktursicht. Die vom *Support-Provider* erbrachte Unterstützungsleistung ist jedoch nicht für jeden User und zu jedem Zeitpunkt gleich, sondern hängt vom vereinbarten Service-Level-Agreement ab. Die ursprünglich mehrdeutige Situation wird durch den Einbezug des Kontextfaktors *Service-Level-Agreement* aufgelöst.

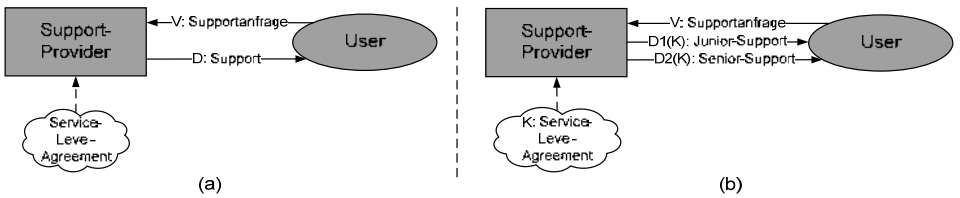


Abb. 6: Kontextsensitiver User-Support (Interaktionsschema, Variantenbildung)

Der Kontextfaktor kann die Werte *Junior-Support* und *Senior-Support* annehmen. Die Variabilität besteht in diesem Fall ausschließlich hinsichtlich der zu erbringenden Supportleistung durch den Support-Provider. Die Empfangsaufgabe auf Seiten des Users ist nicht kontextsensitiv. Aufgrund des vorhandenen Domänenwissens ist bekannt, dass die Ausprägungen des Kontextfaktors binär sind. Daher eignet sich zur weiteren Modellierung des Geschäftsprozesses das Mittel der Variantenbildung. Die Struktur und das Verhalten dieses kontextsensitiven Geschäftsprozesses zeigen Abb. 6b und 7. Der Kontextfaktor wird gekennzeichnet mit „*K*“, um auch in komplexeren Modellen die zur jeweiligen Variantenbildung gehörige Selektorvariable identifizieren zu können. Im vorliegenden Beispiel werden aus der ursprünglichen Leistungserbringung *D: Support* zwei Varianten generiert: *D1(K): Junior-Support* und *D2(K): Senior-Support*. Diese Notation lässt unmittelbar erkennen, dass die jeweilige Durchführungstransaktion vom Kontextfaktor *K* abhängt. Im VES stellen Vorbedingungen sicher, dass nur eine der beiden kontextsensitiven Aufgaben ausgeführt wird.

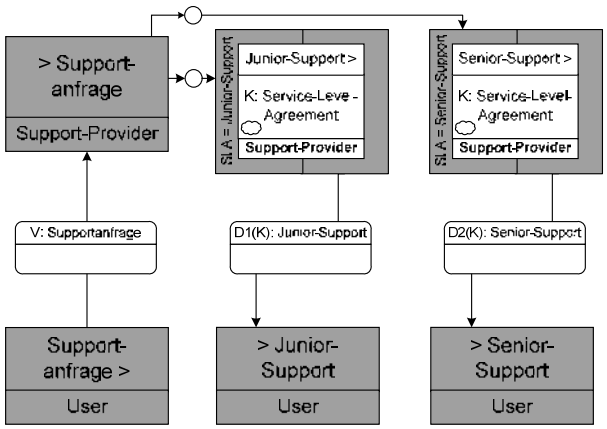


Abb. 7: Kontextsensitives VES mit Vorbedingungen

Fall 2: Kontextsensitive Angebotserstellung

Ein *Ölhändler* beachtet bei der Erstellung von Angeboten das aktuell am Markt herrschende Preisniveau (Abb. 8a). Um den Kontextfaktor und dessen beliebig große Ausprägungsmenge von möglichen Kontextfaktorwerten adäquat behandeln zu können, wird hier die Möglichkeit der Aufgabenparametrisierung angewandt. Aus dem korrespondie-

renden VES (Abb. 8b) ist abzulesen, dass ausschließlich die Erstellungsaufgabe *Angebot>* kontextsensitiv ist. Die tatsächliche Parametrisierung des Lösungsverfahrens erfolgt auf der AwS-Ebene. Dies ist beispielsweise durch Integration eines Web-Services denkbar, der das AwS zur Erstellung von Angeboten mit dem aktuellen Preisniveau versorgt (siehe Abschnitt 6).

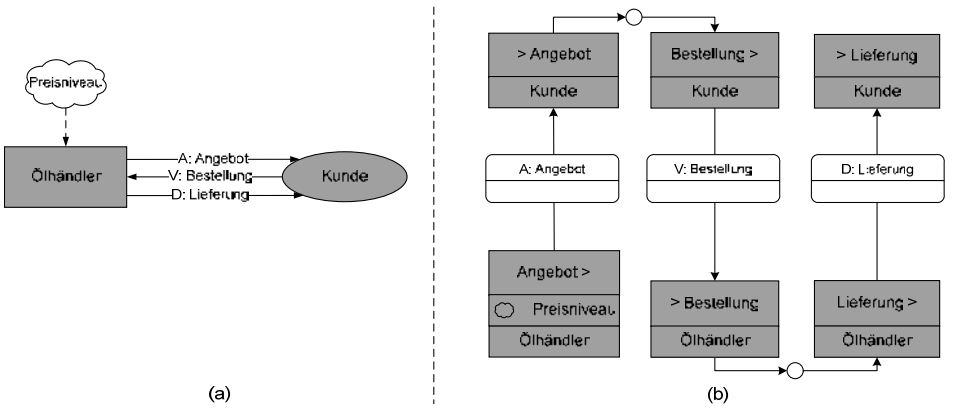


Abbildung 8: Kontextsensitive Angebotserstellung

Fall 3: Kontextsensitive Distribution

Ein Kunde befindet sich in einem Möbelhaus, welches vor Ort Bestellungen über elektronische Terminals aufnimmt und Produkte zudem über einen Onlineshop vertreibt. Das Bestellabwicklungssystem erhält Bestellungen aus beiden Vertriebskanälen. Vor der Auslieferung wird entschieden, ob der Kunde sich im Ladenlokal befindet, und die Ware aus dem Lager an die Abholstation gebracht, oder ob eine Spedition mit dem Transport der Ware zum Wohnort des Kunden beauftragt wird. Den aktuellen Aufenthaltsort des Kunden bezieht das Bestellabwicklungssystem z. B. über einen Web-Service, welcher vom Mobilfunkanbieter des Kunden zur Verfügung gestellt wird. Die Besonderheit dieses Szenarios ist, dass Erstellungs- und Empfangsaufgabe auf den jeweiligen Kontext abgestimmt werden müssen (Abb. 9a). Durch Zerlegung des Objektes *Warendistribution* in die Objekte *Warenbereitstellung* und *Spedition* kann die Kontextsensitivität der Transaktion *D: Übergabe* durch Variantenbildung berücksichtigt werden (siehe Abb. 9b). Somit sind nur noch die aus der Zerlegung resultierenden Aufgaben kontextsensitiv, nicht aber die entsprechenden resultierenden Transaktionen (siehe Abb. 10).

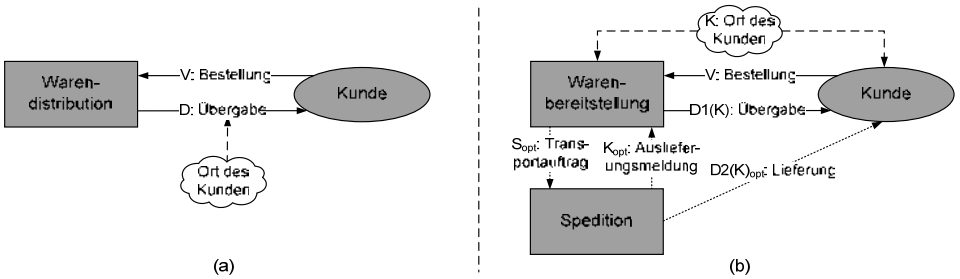


Abb. 9: Kontextsensitive Warendistribution

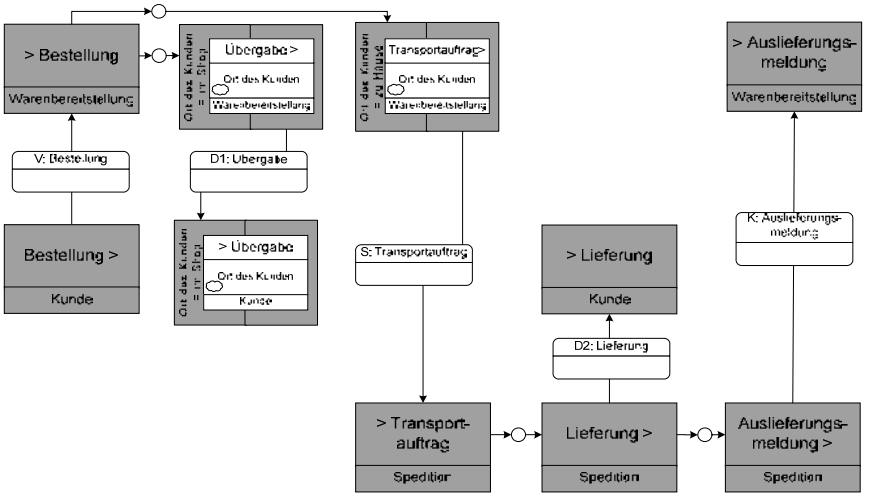


Abb. 10: Kontextsensitives VES Warendistribution

Fall 4: Kontextsensitives Multikanalbanking

Wenn es nicht möglich ist die Kontextsensitivität einer Transaktion mittels Variantenbildung adäquat abzubilden, kann – wie bei betrieblichen Objekten; siehe Fall 2 – Parametrisierung eingesetzt werden. Dadurch treten weder im IAS noch im VES Varianten auf. Obgleich in der Realität vermutlich wenige Situationen existieren, in welchen die Erstellungs- und/oder Empfangsaufgaben betrieblicher Transaktionen derart großer Variabilität unterliegen, dass ein Explizieren in *n* Varianten nicht praktikabel ist, wird dieser Fall aus Gründen der Vollständigkeit dennoch betrachtet. Abb. 11a und Abb. 11b zeigen das GP-Modell des Prozesses der Steuerbescheinigungsanforderung und -zusendung aus Struktur- und Verhaltenssicht. Für die Anforderung des Steuerbescheides stehen dem Kunden im Rahmen des Multikanalbankings die unterschiedlichsten Wege zur Verfügung [APS05]. Die Wahl des Übermittlungskanals wird als Kontextfaktor repräsentiert, welcher die Erstellungs- und Empfangsaufgabe parametrisiert. Diese Variabilität wirkt

sich dementsprechend auch auf AwS-Ebene aus. Erstellungs- und Empfangsaufgabe müssen beispielsweise bei Anforderung der Steuerbescheinigung über das Call Center der Bank anders behandelt werden als bei einer über das Online-Banking-Portal angeforderten Bescheinigung.

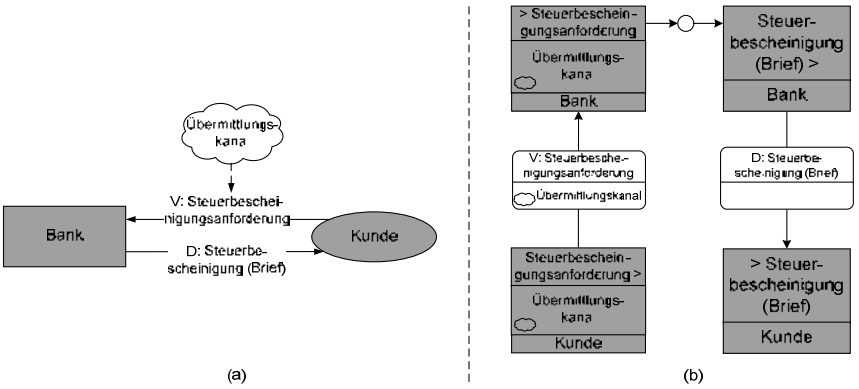


Abb. 11: Kontextsensitives Multikanalbanking (Struktur- und Verhaltenssicht)

6 Kontextsensitive Geschäftsprozessmodelle als Grundlage für kontextsensitive Anwendungssysteme

Im vorhergehenden Abschnitt wurden die Notwendigkeit der Berücksichtigung von Kontext in GP-Modellen und dessen Modellierung erläutert. Anwendungssysteme können kontextsensitive Geschäftsprozesse nur dann entsprechend unterstützen, wenn sie selbst in der Lage sind, die relevanten Kontextfaktorwerte zu ermitteln und zu verarbeiten. Die kontextsensitiven Bestandteile eines AwS müssen hierfür mit entsprechenden Sensoren ausgestattet sein, um während der Prozessausführung die benötigten Kontextfaktorwerte aus der Umwelt zu ermitteln. Das Vorhandensein dieser Werte ist eine Voraussetzung für entsprechende Variantenwahlen oder Parametrisierungen des im AwS implementierten Lösungsverfahrens.

Im Rahmen der SOM-Methodik werden nach der Modellierung der GP-Ebene mittels IAS und VES zur Spezifikation des AwS das Konzeptuelle Objektschema (KOS) und das Vorgangsobjektschema (VOS) metamodellbasiert aus GP-Modellen abgeleitet [FS08, S. 221 ff.]. Diskurs- und Umweltobjekte sowie Leistungsspezifikationen werden in Konzeptuelle Objekttypen abgebildet. Sie sind existenzunabhängig und stehen in der linken Spalte eines Konzeptuellen Objektschemas (Abb. 12a). Aus jeder Transaktion zwischen zwei betrieblichen Objekten des Interaktionsschemas werden weitere Konzeptuelle Objekttypen abgeleitet, die von den erstgenannten Konzeptuellen Objekttypen existenzabhängig sind und rechts davon angeordnet werden. Zusätzlich zu den Ergebnissen des IAS werden die Reihenfolgebeziehungen aus dem Vorgangs-Ereignis-Schema ausgewertet. Sequenzen von Transaktionen führen zu weiteren Existenzabhängigkeiten

zwischen Konzeptuellen Objekttypen. Das in Abschnitt 4 eingeführte Beispiel „Warendistribution“ (siehe Abb. 1) führt zu dem in Abb. 12a abgebildeten Konzeptuellen Objektschema.

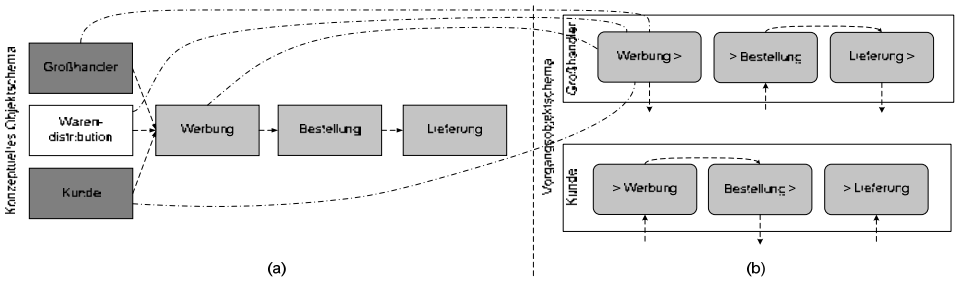


Abb. 12: Anwendungssystemspezifikation in KOS und VOS

Komplementär zum Konzeptuellen Objektschema vervollständigt das Vorgangsobjektschema die Anwendungssystemspezifikation (Abb. 12b). Ein Vorgangsobjektschema umfasst die Aufgaben eines oder mehrerer betrieblicher Objekte. Es besteht aus Vorgangsobjekttypen (VOT), welche direkt aus den jeweiligen Sende- und Empfangsaufgaben eines VES abgeleitet werden. Vorgangsobjekttypen beschreiben das Zusammenwirken Konzeptueller Objekttypen bei der Durchführung von Aufgaben. In Abb. 12 ist exemplarisch die Interaktion zwischen dem VOT *Werbung>* und den korrespondierenden KOTs mittels Strich-Punkt-Linien visualisiert. Das aus dem VES des Beispiels „Warendistribution“ abgeleitete Vorgangsobjektschema zeigt Abb. 12b. Konzeptuelles Objektschema und Vorgangsobjektschema bilden zusammen die fachliche Spezifikation eines Anwendungssystems und dienen als Grundlage für die systemtechnische Implementierung. Kontextfaktoren werden auf der GP-Ebene an betriebliche Objekte, Aufgaben oder Transaktionen annotiert. Auf der AwS-Ebene erfolgt die Kontextsensitivierung analog mittels Annotationen an den entsprechenden Stellen einer AwS-Spezifikation. In einem Konzeptuellen Objektschema werden KOTs, entsprechend dem zur Bildung des Konzeptuellen Objektschemas verwendeten VES, zu kontextsensitiven KOTs (Abb. 13a). Bezüglich der Vorgangsobjekte wird analog vorgegangen. Aus dem kontextsensitiven VES kann direkt das kontextsensitive Vorgangsobjektschema (Abb. 13b) abgeleitet werden. Kontextsensitive Aufgaben aus dem VES werden im Vorgangsobjektschema durch entsprechende kontextsensitive Vorgangsobjekttypen modelliert. Das AwS wird durch den Einbezug von Kontext in die Lage versetzt, mehrdeutige Situationen adäquat zu unterstützen.

Web-Services sind ein mögliches Instrument, um den Wert von Kontextfaktoren aus der Umwelt zu ermitteln. Diverse Anbieter[5] stellen bereits heute ein umfangreiches Sortiment an Web-Services zur Verfügung. Für die kontextsensitive Angebotserstellung eines Ölhändlers (siehe Abschnitt 5) wird beispielsweise die Ermittlung des Angebotspreises für eine bestimmte Menge Öl mit dem aktuellen Preisniveau parametrisiert. Hierfür ruft

[5] Siehe z. B.: http://www.webservicex.net, http://www.xwebservices.com, http://www.fraudlabs.com, http://www.xignite.com

der kontextsensitive KOT *Angebot* über eine *Acquire*-Methode (*Acquire_Preisniveau*) einen Web-Service auf. Der Web-Service liefert das für die Berechnung des Angebotspreises nötige Ölpreisniveau zurück. Abb. 14 zeigt das Konzeptuelle Objektschema inklusive der Definition des Konzeptuellen Objekttyps *Angebot*. Dieser beinhaltet zum einen Attribute und zum anderen Methoden zum Lesen oder Verändern dieser Attribute. Die Menge der Attribute aller Konzeptuellen Objekttypen beschreibt den Zustand des Anwendungssystems, der durch die entsprechenden Set-Methoden verändert wird. Die Get- und Set-Methoden werden von Vorgangsobjekten aufgerufen. Zusätzlich ist eine Acquire-Methode zur Herstellung einer Verbindung zu einem Web-Service vorhanden. Im vorliegenden Fall wird dem Konzeptuellen Objekttyp erlaubt, eines seiner Attribute, nämlich das aktuelle Ölpreis-Niveau (*$_Ölpreis_Niveau:float*), durch den Aufruf des Web-Services zu verändern. Die Acquire-Methode wird aufgerufen, sobald der zugehörige Vorgangsobjekttyp *Angebot>* (siehe Abb. 13b) die KOT-Methode *Angebot.Get_Ölpreis_Niveau(Angebot_ID)* aufruft.

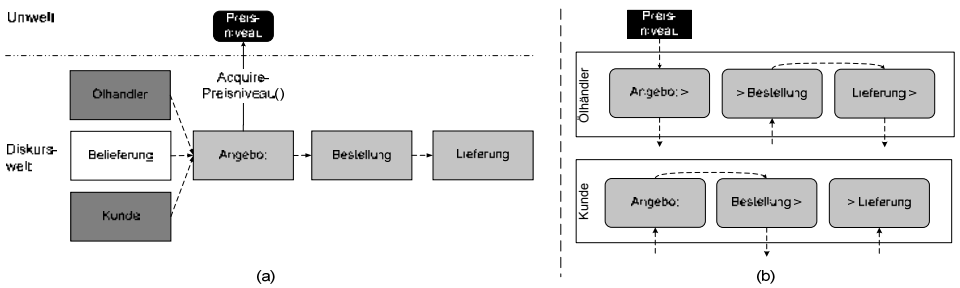


Abb. 13: Kontextsensitive Anwendungssystemspezifikation in KOS und VOS am Beispiel „Kontextsensitive Preisbildung“

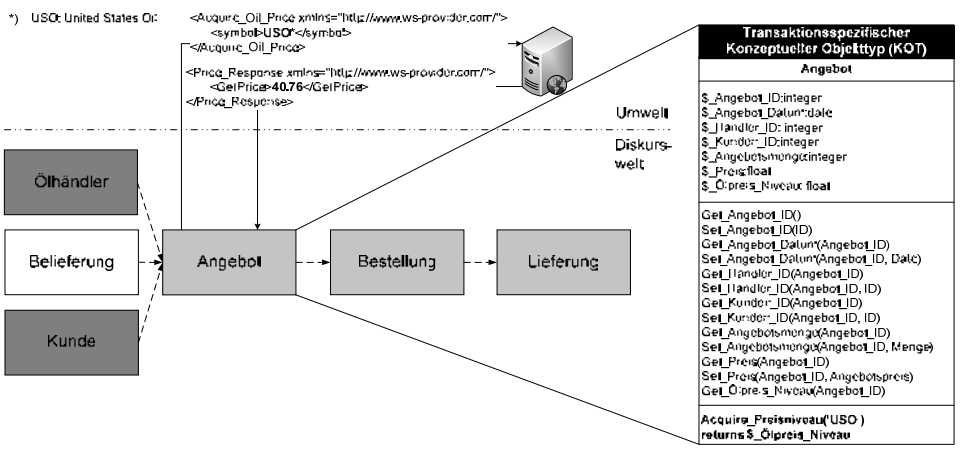


Abb. 14: Beschaffung des Ölpreisniveaus über einen Web-Service

7 Zusammenfassung und Ausblick

Komplexer werdende Geschäftsprozesse mit hohen Flexibilitätsanforderungen verlangen auch bei ihrer Modellierung komplexer werdende Geschäftsprozessmodelle. Einer der Treiber höherer Komplexität ist der Kontext von Geschäftsprozessen, der zunehmend mehr in den Modellen berücksichtigt werden muss. In diesem Beitrag wird eine Erweiterung der SOM-Methodik vorgestellt, die unterschiedliche Kontextformen aufgreift und dafür Modellkonstrukte anbietet. Die Integration in das Modellierungskonzept der SOM-Methodik erlaubt, die Kontextinformationen auch auf die Ebene der Anwendungssystemspezifikation zu propagieren.

Die Nutzung dieser Erweiterung der SOM-Methodik bietet den Vorteil höherer Präzision der Modellierung durch erhöhte Abbildungstreue, stellt aber auch erhöhte Anforderungen an die Modellierung und deren Umsetzung in Anwendungssystemen. Bei der Modellierung von Geschäftsprozessen müssen die Kontextfaktoren identifiziert werden. Aus den bisher durchgeführten Fallstudien ist erkennbar, dass die Erkennung sekundärer Einflussfaktoren als Kontext schwieriger als die Erkennung der offensichtlichen primären Einflussfaktoren sein kann. Bei der Umsetzung in Anwendungssystemen ist Kontext bei deren Entwicklung und Betrieb zu berücksichtigen. In der Entwicklungsphase bedeutet die Berücksichtigung von Kontext wie bei der Geschäftsprozessmodellierung erhöhte Komplexität, in der Betriebsphase müssen Sensoren für die Erfassung von Kontextfaktorwerten verfügbar sein. In den Beispielen wurden ein Sensor für die Erkennung des Ortes eines Kunden oder die Ermittlung des Ölpreisniveaus genannt. Schwieriger ist die Erfassung und Verarbeitung von Kontextfaktoren wie Konjunktur oder Kaufstimmung z. B. bei der Modellierung von Vertriebsprozessen.

Komplexer werdende Geschäftsprozesse und ein höherer GP-Automatisierungsgrad werden die Berücksichtigung von Kontext bei der Modellierung von Geschäftsprozessen und der Entwicklung von Anwendungssystemen erzwingen. Es geht also weniger um die Frage, ob Kontext in Geschäftsprozessmodellen und AwS-Spezifikationen zu berücksichtigen ist, sondern um die Frage der geeigneten Modellierung und Erfassung. In weiteren Arbeiten werden daher die hier vorgestellten Modellkonstrukte mit intensiver Beteiligung von Praxispartnern in Fallstudien untersucht und evaluiert.

Literaturverzeichnis

[Ab99] Abowd, G. D. et al.: Towards a Better Understanding of Context and Context-Awareness. In (Gellersen, H.-W. Hrsg.): HUC ’99: Proceedings of the 1st international symposium on Handheld and Ubiquitous Computing. Springer, Berlin, 1999; S. 304–307.

[APS05] Ammon, R. von; Pausch, W.; Schimmer, M.: Realization of Service-Oriented Architecture (SOA) Using Enterprise Portal Platforms Taking the Example of Multi-Channel Sales in Banking Domains. In (Ferstl, O. K.; Sinz, E. J.; Eckert, S.; Isselhorst, T. Hrsg.): Wirtschaftsinformatik 2005. eEconomy, eGovernment, eSociety. Physica-Verlag Heidelberg, Heidelberg, 2005; S. 1503–1518.

[Ba09] Bauer, F. L.: Historische Notizen zur Informatik. Springer-Verlag, Berlin, Heidelberg, 2009.

[BC04] Biegel, G.; Cahill, V.: A Framework for Developing Mobile, Context-aware Applications. In (IEEE Computer Society Hrsg.): Proceedings of the Second IEEE International Conference on Pervasive Computing and Communications (PerCom 2004), 2004; S. 361–365.

[Br96] Brown, P. J.: The Stick-e Document: a Framework for Creating Context-aware Applications. In EPODD - Electronic Publishing, Origination, Dissemination and Design, 1996, 8; S. 259–272.

[Br97] Brown, P. J.: Context-Aware Applications: From the Laboratory to the Marketplace. In Institute of Electrical and Electronics Engineers, 1997, 4; S. 58–64.

[Co05] Coutaz, J. et al.: Context is Key. In Communications of the ACM, 2005, 48; S. 49–53.

[Di73] Dinkelbach, W.: Modell - ein isomorphes Abbild der Wirklichkeit? In (Grochla, E.; Szyperski, N.; Grochla-Szyperski Hrsg.): Modell- und computergestützte Unternehmungsplanung. Gabler, Wiesbaden, 1973; S. 152–162.

[Fe79] Ferstl, O. K.: Konstruktion und Analyse von Simulationsmodellen. In (Angermann, A. Hrsg.): Beiträge zur Datenverarbeitung und Unternehmensforschung. Anton Hain Meisenheim GmbH, Königstein/Ts., 1979.

[FF98] Franklin, D.; Flachsbart, J.: All gadget and no representation makes jack a dull environment. In (AAAI Press Hrsg.): Spring Symposium on Intelligent Environments. AAAI TR SS-98-02, 1998; S. 155 160.

[FS08] Ferstl, O. K.; Sinz, E. J.: Grundlagen der Wirtschaftsinformatik. Oldenbourg, München, 2008.

[HBR09] Hallerbach, A.; Bauer, T.; Reichert, M.: Issues in Modeling Process Variants with Provop. In (Ardagna, D.; Mecella, M.; Yang, J. Hrsg.): Business process management workshops. Springer, Berlin, New York, 2009; S. 56–67.

[Is07] Isselhorst, T.: Modellierung von Kontext für Führungsinformationssysteme. WiKu-Verl., Duisburg, 2007.

[Ja01] Jackson, M.: Problem frames. Analysing and structuring software development problems. Addison-Wesley [u.a.], Harlow, 2001.

[Ko61] Kosiol, E.: Modellanalyse als Grundlage unternehmerischer Entscheidungen. In Zeitschrift für handelswissenschaftliche Forschung (ZfbF), 1961, 13; S. 318–334.

[Mc87] McCarthy, J.: Generality in artificial intelligence. In Communications of the ACM, 1987, 30; S. 257–267.

[Mo06] Morgenroth, K.: Kontextbasiertes Information-Retrieval. Modell, Konzeption und Realisierung kontextbasierter Information-Retrieval-Systeme. Logos-Verl., Berlin, 2006.

[Pl09] Ploesser, K. et al.: Learning from Context to Improve Business Processes. In BPTrends, 2009, 6; S. 1–7.

[Pü09] Pütz, C. et al.: Geschäftsprozesse in Medizinischen Versorgungszentren und ihre Flexibilitätsanforderungen – ein fallstudienbasiertes Szenario, Bamberg, 2009, http://www.forflex.de/uploads/AB/forflex-2009-001.pdf. Abruf am 2009-10-20.

[Ro98] Rodden, T. et al.: Exploiting Context in HCI Design for Mobile Systems. In (Johnson, C. Hrsg.): Proceedings of the First Workshop on Human Computer Interaction with Mobile Devices. GIST Technical Report G98-1, Glasgow, 1998.

[RPM99] Ryan, N.; Pascoe, J.; Morse, D.: Enhanced Reality Fieldwork: The Context-Aware Archaeological Assistant. In (Dingwall, L. Hrsg.): Archaeology in the age of the Internet. Computer applications and quantitative methods in archaeology. Proceedings of the 25th anniversary conference, University of Birmingham, April 1997. Archaeopress, Oxford, 1999.

[RRF08] Rosemann, M.; Recker, J. C.; Flender, C.: Contextualization of Business Processes. In International Journal of Business Process Integration and Management, 2008, 3; S. 47–60.

[Sc94] Schilit, B. N.: Disseminating Active Map Information to Mobile Hosts. In IEEE Network, 1994, 8; S. 22–32.

[Sc98] Schütte, R.: Grundsätze ordnungsmäßiger Referenzmodellierung. Konstruktion konfigurations- und anpassungsorientierter Modelle. Gabler, Wiesbaden, 1998.

[SLF03] Strang, T.; Linnhoff-Popien, C.; Frank, K.: CoOL: A Context Ontology Language to enable Contextual Interoperability. In (Stefani, J.-B.; Demeure, I.; Hagimont, D. Hrsg.): Distributed applications and interoperable systems. Proceedings of the 4th IFIP WG 6.1 international conference, Paris, France, November 17-21, 2003. Springer, Berlin, 2003; S. 236–247.

[SN07] Saidani, O.; Nurcan, S.: Towards Context Aware Business Process Modelling. Workshop on Business Process Modelling, Development and Support (BPMDS'07), Trondheim, Norway, 2007.

[WJH97] Ward, A.; Jones, A.; Hopper, A.: A new location technique for the active office. In IEEE Personal Communications, 1997, 4; S. 42-47.

Defining the Quality of Business Processes

Robert Heinrich[1] and Barbara Paech[2]

Abstract: Business process models are used to gain a joint understanding of complex processes. Often they are applied in change projects where either the supporting IT or the processes themselves or both are to be improved. So it is an important question to assess the quality of the modeled business processes. However, so far there is no standard definition of the quality of a business process. Furthermore, business process models are not tuned to capture quality aspects. The goal of our work is to collect important quality characteristics and attributes of processes and to enhance business process modeling languages with means to express these attributes. This paper is a first step in this direction. We define a first set of quality characteristics, attributes and measures for these attributes in a business process. As an example we evaluate how well these measures can be expressed in a BPMN model.

1 Introduction

Business process models aim at providing a joint understanding of business processes. Therefore, they typically cover information about structure and behavior like description of activities or decisions within the process. But they do not aim at providing quality information. Quality information, for example, is information about the reliability or the usability of a business process. That information is of high interest for organizations because business process models are often used in change projects where either the supporting IT or the processes themselves or both are to be enhanced. Furthermore, the quality of a business process highly affects the success of an organization. All the more curious is that there is no standard definition of what constitutes the quality of business processes. This is in contrast to the definition of software product quality which is standardized by ISO/IEC 9126 [ISO01].

This paper discusses how to assess the quality of business processes and business process models. There is a wide range of papers relating to business process quality, but only few try to provide a unifying basis for quality information. These typically try to adopt ISO/IEC 9126 for business processes (cf. [GD05], [HMR09]). However, the resulting set of characteristics is on a very high level of abstraction (see related work in section 5). Furthermore, it is not possible to understand what was adapted how. And it is not clear how relevant the characteristics are for practice.

In this paper we use a more systematic and detailed approach. As a first step we present a meta-model capturing and classifying the quality characteristics of a business process and their relationships. We want to use these characteristics to propose changes to

[1] Institute for Computer Science, University of Heidelberg, Im Neuenheimer Feld 326, 69120 Heidelberg, Germany, heinrich@informatik.uni-heidelberg.de
[2] Institute for Computer Science, University of Heidelberg, Im Neuenheimer Feld 326, 69120 Heidelberg, Germany, paech@informatik.uni-heidelberg.de

business process modeling notations. Furthermore, - as for the software quality characteristics - the characteristics are useful for evaluating specific processes or for defining quality requirements for specific processes. The characteristics are quite general and thus cannot be used directly for evaluation and requirements definition. Therefore, it is important to associate specific measures with them. As an intermediate step we collect attributes which describe important aspects of the characteristics that should be measured. It is *not* the goal of this paper to define a complete set of exact measures for these attributes. However, we collect important base measures which can be composed to complex measures depending on the purpose of measurement. By associating with the characteristics example attributes and measures adopted from practice problems we justify that the characteristic is relevant for practice. These measures then constitute the set of concepts which should be expressible by a business process model in order to support the assessment of the quality of the modeled process. As an example, we evaluate how the Business Process Modeling Notation (BPMN) can express these measures, as it is an up to date and very wide-spread modeling notation for business processes. It is important to note that we are *not* interested in the quality of the business process model as a model. Model qualities are discussed e.g. in [LC05], [MD09] and [MDN09].

The remainder of this paper is organized as follows. Section two introduces relevant standards and concepts. In section three quality characteristics, attributes and measures of the business process are presented. Section four evaluates how well BPMN can express the measures. Section five discusses related work and section six concludes the paper.

2 Background

This section clarifies the background of the paper by presenting standards and concepts relevant for the understanding of the paper.

There is a close relationship between software and business processes [Os87]. For the basic terminology we adapt the terminology from the software product quality standards: In analogy to the definition of data quality characteristic in ISO/IEC 25012 [ISO08] we define a *business process quality characteristic* as a category of business process quality attributes. Adapting the definition of attribute in ISO/IEC 25000 [ISO05], a *business process quality attribute* is an inherent property of a business process that can be distinguished quantitatively or qualitatively by human or automated means. Adapting the definition of data quality measure in ISO/IEC 25012 [ISO08] we define a *business process quality measure* as a variable to which a value is assigned as the result of measurement of a business process quality attribute. In the following, we use *characteristic* instead of *business process quality characteristic*, *attribute* instead of *business process quality attribute* and *measure* instead of *business process quality measure*.

While these definitions can be adapted from software quality standards, this is not that easy with the definitions of characteristics themselves. To validate whether our definitions are relevant we identified typical problems with business processes in practice such as inadequate capacity of resources or unqualified actors. Here we used as a source our own experiences [ABP10] and a process check list [Fi09]. A process check list is a collection of business process problems. Thus, it is based on the assumption that often the business process quality is threatened by similar problems in different organizations or projects. Problems represented in a process check list are typically more specific than quality characteristics. Thus, they can be adapted to attributes or measures.

3 Business Process Quality

This section presents business process quality characteristics, classifies them in a meta-model and provides attributes and measures of these characteristics. We have identified the characteristics from software product quality standards and adapted them while identifying attributes and measures based on the problems (see related work in section 5).

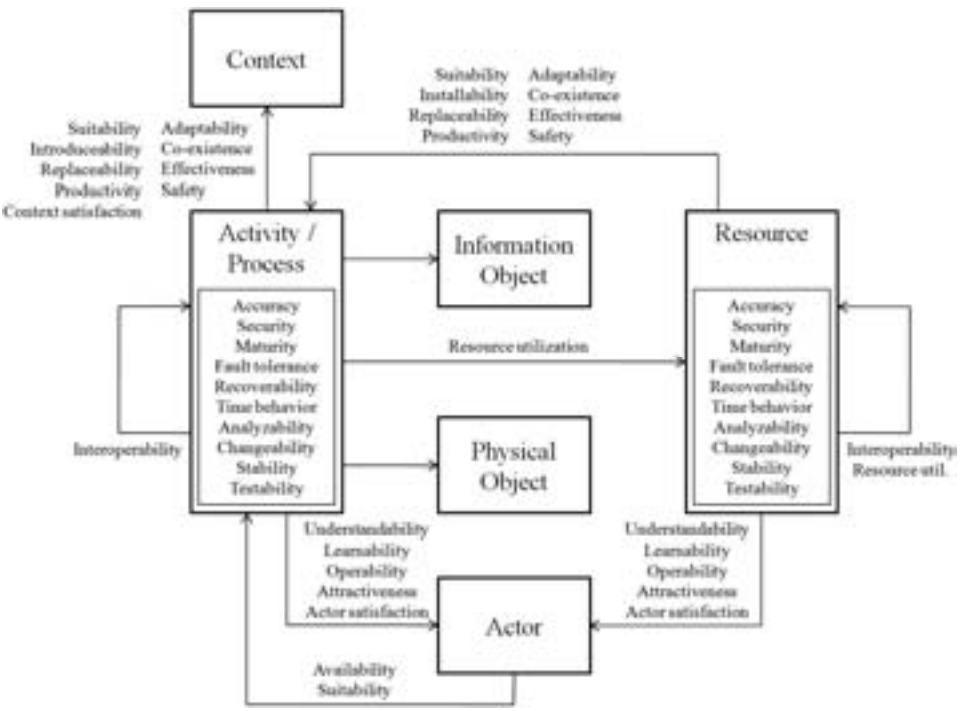


Fig. 1: Business Process Quality Characteristics Meta-Model

Business process quality refers to the components of a business process, to the process as a whole as well as to the context of the process. The context of a business process covers the conditions of use as well as the organizational environment. Components of a business process are the activities, the actors performing these activities, the information objects and physical objects handled and created by the process as well as the resources necessary for execution. So the characteristics are grouped by those categories. Figure 1 provides a meta-model of business process quality characteristics and visualizes their dependencies. The nodes correspond to the categories and the characteristics are listed either within the node or on an edge between nodes. If a characteristic is located on an edge, the assessment of that characteristic depends on information of another category, where A → B means that B must be considered to assess A. Information and physical objects are not studied in detail here (see 3.1.4). Therefore, their attributes are not shown in the figure.

3.1 Characteristics

Next we clarify the characteristics presented in Figure 1 by providing their definitions.

3.1.1 Activity Characteristics

A process consists of activities, where an activity can be atomic or can be a process itself (this means it contains sub-activities). In analogy to software we consider the documentation of the activity as a part of the activity. The following characteristics apply to activities (and by definition also to the process as a whole). We developed these characteristics based on ISO/IEC 9126-1 [ISO01]. ISO/IEC 9126-1 presents characteristics which are further subdivided into sub-characteristics. The latter are also called characteristics in the following and are listed indented below. Mostly, we took the definitions from ISO/IEC 9126-1 and changed only single words. For example, we replaced "software product" by "activity". Sometimes, we had to adapt further parts of the definition or changed the name of the characteristic. These characteristics are marked with "**(N)**" in the following and an explanation for the change is given.

Functionality is the capability of the process to provide activities which meet stated and implied needs when used under specified conditions.

> **Suitability (N)** is the capability of the activity to be appropriate for a specified context of use.
>
> *In ISO/IEC 9126-1 suitability of the software product is focused on "specified tasks and user objectives". We wanted to use uniform terms. Thus, we use "context of use" instead of tasks. Furthermore, as an actor is a process component we cover suitability for user objectives by "actor satisfaction". Moreover, the term "set of functions" is improper for activities, so we changed the wording.*
>
> **Accuracy** is the capability of the activity to provide the right or agreed results or effects with the needed degree of precision.
>
> **Interoperability (N)** is the capability of the activity to be executed before or

after one or more other specified activities.
With respect to activities the interaction corresponds to execution dependency, so we reworded the definition.
Security (N) is the capability of the activity to protect information and physical objects so that unauthorized actors or resources cannot access them and authorized actors or resources are not denied access to them.
An activity should also protect physical objects, so we extended the definition.

Reliability is the capability of the activity to maintain a specified level of performance when used under specified conditions.

Maturity is the capability of the activity to avoid failure as a result of faults in the activity.
Fault tolerance is the capability of the activity to maintain a specified level of performance in cases of faults or of infringement of its specified interface.
Recoverability is the capability of the activity to re-establish a specified level of performance and recover the information and physical objects directly affected in the case of a failure.

Usability is the capability of the activity to be understood, learned, used and attractive to the actor, when used under specified conditions.

Understandability (N) is the capability of the activity to enable the actor to understand whether it is suitable, and how it can be executed in a particular context of use.
*We replaced "*tasks and conditions of use*" as stated in ISO/IEC 9126-1 by "*context of use*" to use uniform terms.*
Learnability is the capability of the activity to enable the actor to learn its execution.
Operability is the capability of the activity to enable the actor to operate and control it.
Attractiveness is the capability of the activity to be attractive to the actor.

Efficiency (N) is the capability of the activity to provide appropriate performance, relative to the amount of resources and the actor time used, under stated conditions.
The efficiency of a software product may only depend on the amount of resources but the efficiency of a business process also depends on the actor time used, so we extended that definition.

Time behavior (N) is the capability of the activity to provide appropriate transport and processing times and throughput rates when executed under stated conditions.
In ISO/IEC 9126-1 the definition of time behavior covers response time. We removed response time because we consider that as inappropriate for activities. Moreover, we extended the definition by transport time which is typically used for business processes.
Resource utilization is the capability of the activity to use appropriate amounts and types of resources when executed under stated conditions.

Maintainability is the capability of the activity to be modified.

Analyzability is the capability of the activity to be diagnosed for deficiencies or causes of failures, or for the parts to be modified to be identified.

Changeability is the capability of the activity to enable a specified modification to be executed.

Stability is the capability of the activity to avoid unexpected effects from modifications of the activity.

Testability (N) is the capability of the activity to be validated.

In ISO/IEC 9126-1 testability requires "modified software" to be validated. We could not see any reason for this restriction.

Portability is the capability of the activity to be transferred from one context of use to another.

Adaptability (N) is the capability of the activity to be adapted for different specified contexts of use.

We shortened the definition of adaptability because the add-on "without applying actions or means other than those provided for this purpose for the software considered" *seemed unnecessarily complex*.

Introduceability (N) is the capability of the activity to be introduced in a specified context of use.

In our view "introduction" is more appropriate for business processes than "installation", so we changed the naming.

Co-existence is the capability of the activity to be executed with other independent activities in a common context of use sharing common resources.

Replaceability is the capability of the activity to be used in place of another specified activity for the same purpose in the same context of use.

Quality in use is the capability of the activity to enable specified actors to achieve specified goals with effectiveness, productivity, safety and satisfaction in specified contexts of use.

Effectiveness is the capability of the activity to enable actors to achieve specified goals with accuracy and completeness in a specified context of use.

Productivity (N) is the capability of the activity to enable actors to achieve specified goals with appropriate efforts in a specified context of use.

We changed the definition of the characteristic productivity because "enable users to expend appropriate amounts of resources in relation to the effectiveness achieved in a specified context of use" *seemed unnecessarily complex*.

Safety is the capability of the activity to achieve acceptable levels of risk of harm to people, business, process, property or the environment in a specified context of use.

Actor satisfaction (N) is the capability of the activity to fulfill a specific actor objective.

Context satisfaction (N) is the capability of the activity to fulfill a particular constraint in a specified context of use.

We split the ISO characteristic satisfaction into actor satisfaction which is

focused on a specific actor objective and context satisfaction which is focused on contextual constraints, like requirements of the customer, because we want to enable a separate consideration of these characteristics.

Compliance (N) is the capability of the activity to adhere to standards, conventions or regulations in laws and similar prescriptions.
In ISO/IEC 9126-1 for each characteristic there is a compliance sub-characteristic. We generalize these sub-characteristics using the characteristic compliance.

3.1.2 Resource Characteristics

A resource is used in a process, for example, a machine, a device, or an IT system. Activities usually need resources for execution. We use the term resource not only for a single one, but also for a whole resource landscape. The quality of these resources affects the quality of the activities. The context of use of a resource is the activity which uses the resource. Therefore, the characteristic context satisfaction is omitted for resources. As a resource is very similar to a software product one can adapt again all ISO/IEC 9126-1 definitions to resources. In the following we present the definition of resource characteristics which differ more substantially from the definition of the activity characteristic than just replacing "activity" by "resource".

Interoperability is the capability of the resource to interact with one or more specified resources.
Installability is the capability of the resource to be installed in a specified context of use.

3.1.3 Actor Characteristics

An actor performs one or more activities. The quality of a business process depends on the availability and skills of those who perform it. Next the characteristics of the category actor are listed. We developed these characteristics to capture attributes and measures from practice [Fi09] related to actors. Note that the actor characteristics differ from the resource characteristics. Thus, in analogy to related work [HMR09] we do not treat actor as a resource.

Availability is the capability of the actor to be able to perform the activity in the required unit of time.
Suitability is the capability of the actor to perform the activity well.

3.1.4 Information and Physical Object Characteristics

The quality of a business process also depends on the quality of its input and output. Input and output can be information objects as well as physical objects. ISO/IEC 25012 [ISO08] discusses data quality characteristics which can be easily adapted to information objects. Furthermore, there is more detailed work on information and data quality (cf. [BP85], [WS96], [KSW02], [Le02], [PLW02], [ES07]) which should be considered. We

could not find a standard which covers the quality of physical objects comprehensively. We suppose physical object quality characteristics can be adapted from data quality characteristics. Information and physical object quality will be the topic of a separate paper. So both categories are not within the scope of this paper.

3.2 Attributes and Measures

One goal of this paper is to refine abstract quality characteristics to specific measures. As an intermediate step we provide attributes for the characteristics. The following paragraphs do not represent a complete list of attributes and measures per characteristic but rather a collection of typical ones. One way to choose them is again the adaptation based on software product quality measurement elements in [ISO07]. However, we are even more interested to see whether the characteristics cover the process goals and problems encountered in practice. Therefore, we looked at typical business process problems represented in a process checklist [Fi09] and at measures we have developed ourselves in the medical context [ABP10]. If the ISO is not referenced as source below, we did not find an attribute for this characteristic respectively a measure for this attribute in the standard. The measures discussed in the following paragraphs are base measures which can be composed to complex measures. Typically they are not meaningful on their own (e.g. number of automated activities), but only in relation to the overall process (e.g. number of activities altogether) or the process component. We have omitted the latter base measure as it can be inferred easily. Note that we do not list all the characteristics of section 3.1 in the following paragraphs, but only the ones whose attributes and measures we found in the literature from practice. This does not mean that the other characteristics are not relevant, but yet we could not find attributes and measures for them.

Table 1 shows attributes and measures of the activity characteristics. The columns labeled with *Attr. Source* and *Measure Source* present the source if we took or adapted the attribute or measure from related work. The column labeled with BPMN is used for the evaluation of BPMN (see section 4). Measures or attributes marked with "E" focus on the behavior during the execution of the process and measures or attributes marked with "S" refer to structural process properties and documentation properties. This is also used for the evaluation of BPMN. Some of the rows are explained in detail after the table.

Characteristic / Attribute	Attr. Source	Base Measure	Measure Source		BPMN
Interoperability					
Interfaces		Number of message exchanges between activities	[ABP10]	E / S	Yes
Maturity					
Fault density	[ISO07]	Number of detected faults, activity size	[ISO07]	E	No
		Number of activities that terminate correctly	[ABP10]	E	Yes
Callback		Number of callbacks	[Fi09]	E	Yes

Characteristic / Attribute	Attr. Source	Base Measure	Measure Source		BPMN
Fault tolerance					
Exception handling		Number of handled exceptions	[ABP10]	E	Yes
Understandability					
Completeness of description	[ISO07]	Number of described activities	[ISO07], [ABP10]	S	Yes
		Number of documented process goals	[ABP10]	S	No
		Number of defined process beginnings and ends	[ABP10]	S	Yes
Variants		Number of XOR decisions	[Fi09], [ABP10]	S	Yes
Loops		Number of loops	[Fi09]	S	Yes
Parallel paths/activities		Number of parallel paths/activities	[Fi09], [ABP10]	S	Yes
Process components		Number of activities	[ABP10]	S	Yes
		Number of actors	[ABP10]	S	Yes
Process components per actor		Number of activities per actor	[ABP10]	S	Yes
Learnability					
Effectiveness of the documentation	[ISO07]	Number of activities successfully completed after viewing documentation	[ISO07]	E	No
Fit between expertise		Expertise needed for the activity, expertise of the actor	[Fi09]	S	No
Time behavior					
Mean amount of throughput	[ISO07]	Throughput, number of evaluations	[ISO07]	E	No
Processing time efficiency		Number of missing triage	[Fi09]	S	Yes
		Number of appropriately outsourced activities	[Fi09]	E	No
		Number of unnecessary activities	[Fi09]	E	No
		Number of automated activities	[ABP10]	S	No
		Number of unnecessary sequential flows between activities	[Fi09]	S	No
		Number of objects with complex handling	[Fi09]	E	No
		Number of media disruptions	[ABP10]	S	No
		Number of unnecessary repetition of activities	[ABP10]	E	No
Transport time efficiency		Number of inappropriate means of transportation	[Fi09]	E	No
		Number of inappropriate routes of transportation	[Fi09]	E	No
		Number of unnecessarily transported objects	[Fi09]	S	Yes
		Number of objects with complex handling	[Fi09]	E	No
Wait time		Number of missing groupings	[Fi09]	S	No
		Number of parallel paths with very different processing time	[Fi09]	E	No

Characteristic / Attribute	Attr. Source	Base Measure	Measure Source		BPMN
Resource utilization					
Mean occurrence of errors	[ISO07]	Number of error messages and failures, number of evaluations	[ISO07]	E	No
Capacity of the resource wrt. activity	[Fi09]	Number of cases in which a resource is not available	[Fi09]	E	No
Amount of resources		Number of resources involved	[ABP10]	S	No
Adequate resource usage	[ABP10]			E	No
Productivity					
activity time	[ISO07], [ABP10]			E	No
Actor satisfaction					
Opinion of the actor		Number of complaints by the actors	[ABP10]	E	No
Context satisfaction					
Opinion of the customer		Number of complaints by the customers	[ABP10]	E	No

Tab. 1: Characteristics, Attributes and Measures of Activity

In the following we discuss attributes and measures represented in Table 1 which may require additional explanation. The characteristic *interoperability* of an activity depends on the attribute *interface* because the interfaces determine whether an activity can be executed before or after one another. The attribute interface, for example, is assessed by the measure *number of message exchanges between activities*. The characteristic *maturity* is affected by the attribute *callback*. A callback is a question an actor needs to get answered to continue the execution of an activity. A callback, for example, is caused by an error or an ambiguity. The higher the number of callbacks, the lower is the maturity of the activity. The characteristic *learnability* is influenced by the fit between the expertise of the activity and the actor because an actor with a low expertise will have problems learning an activity which expects a high expertise. The attribute *fit between expertise* depends on the measures *expertise needed for the activity* and *expertise of the actor*. The characteristic *time behavior* is affected by the attributes *mean amount of throughput*, *processing time efficiency* and *wait time*. The mean amount of throughput depends on the measures *throughput* and *number of evaluations*. According to [ISO07] an evaluation consists of iterations with same input and same scenario. The processing time depends on the measure *number of missing triage*. Triage is a split handling of routine, moderate and problem cases into three separate activities. Triage decreases the processing time because it speeds up the handling of routine cases. Moreover, the processing time is influenced by the measure *number of media disruptions*. For example, if there is a media disruption between two resources, an actor may have to transfer information from one resource to another manually. This is time-consuming. The wait time depends on the measure *number of missing groupings*. Objects of the same or similar type should be processed in groups to avoid frequent changes of the object type. Frequent changes of the object type may increase the wait time. A missing grouping is a single-processed object which is better to be processed in a group. The measure *number of parallel paths with very different processing time* affects the wait time because the

objects on the path(s) with lower processing time have to wait until the path with the highest processing time is completed.

Table 2 shows the resource characteristics and allocates typical attributes and measures.

Characteristic / Attribute	Attr. Source	Base Measure	Measure Source	
Suitability				
Up-to-dateness	[Fi09], [ABP10]			S
Interoperability				
Interface		Number of message exchanges between resources	[ABP10]	E / S
Security				
Authentication		Number of different logins or authentication needed by one actor within one process	[ABP10]	S
Maturity				
Fault density	[ISO07]	Number of detected faults, resource size	[ISO07], [ABP10]	E
Recoverability				
Restartability	[ISO07]	Number of restarts which met required time during testing or user operation support	[ISO07]	E
		Time needed to recover the system	[ABP10]	E
Mobility of functionality in case of failures	[ABP10]			S
Understandability				
Completeness of description	[ISO07]	Number of functions described in the resource description	[ISO07]	S
Effort required for understanding	[ABP10]			E
Learnability				
Effectiveness of the user documentation	[ISO07]	Number of operations successfully completed after accessing user documentation	[ISO07]	E
Effort required for learning	[ABP10]			E
Operability				
Physical accessibility	[ISO07]	Number of functions which can be customized	[ISO07]	S
Effort required for operation	[ABP10]			E
Service (e.g. hotline)		Number of additional services	[ABP10]	S
Ergonomics of the resource	[Fi09]			E
Attractiveness				
User interface attractiveness	[ISO01], [ABP10]			S
Time behavior				
Mean response time	[ISO07]	Response time, number of evaluations	[ISO07], [ABP10]	E
Resource utilization				
Maximum memory utilization	[ISO07]	Memory utilization, number of evaluations	[ISO07]	E
Redundancy of functionality	[ABP10]			S

between resources				
Actor satisfaction				
Opinion of the actor		Number of complaints by the actors	[ABP10]	E

Tab. 2: Characteristics, Attributes and Measures of Resource

This paragraph provides further explanations on selected attributes and measures presented in Table 2. The characteristic *recoverability* depends on the attribute *mobility of functionality in case of failures*. The mobility is important to transfer the functionality from on resource to another in case of a failure of the resource, and thus to recover the functionality of the resource. The characteristic *operability* depends on the attribute *physical accessibility* because the physical accessibility is the prerequisite to operate the resource. A resource can utilize other resources, for example, to store data. Thus, the characteristic *resource utilization* is refined by the attribute *maximum memory utilization*.

The actor characteristics and related attributes and measures are summarized in Table 3.

Characteristic / Attribute	Attr. Source	Base Measure	Measure Source		BPMN
Availability					
Capacity of the actor	[Fi09]			E	No
Suitability					
Skills of the actor		Qualification, expertise, social competence, team skills, motivation, performance ability	[Fi09]	S	No

Tab. 3: Characteristics, Attributes and Measures of Actor

The characteristic *availability* depends on the *capacity of the actor* because an actor is available if he is able to provide the needed capacity at the required unit of time. The characteristic *suitability* covers all the *skills* an actor needs to perform a specific activity. Examples are qualification, expertise, social competence, team skills, motivation and performance ability.

3.3 Summary

The tables above provide insight in the adequateness of our characteristics. We checked the usefulness of the characteristics adapted from a software quality standard for business processes by comparing them with process problems from practice. The activity characteristics and the resource characteristics were sufficient for capturing most of the problems, we only had to develop two actor characteristics to cover all the problems we found in literature. However, only 9 out of the 26 characteristics defined for activities and 12 out of 25 characteristics defined for resources were needed. This might indicate that many of the adapted characteristics are not practically relevant. Moreover, it might indicate that different characteristics are relevant for activities and resources. Furthermore, as one can see, the measures derived from practice differ very much from the measures adapted from the ISO documents. While the ISO measures can often directly be derived from the definitions of the characteristics, the problems capture very

specific practical insights, e.g. measuring the usage of triages to reduce processing time. This clearly shows that the characteristic definitions are only a very first step to the understanding of process quality.

4 Evaluation of BPMN

In the previous section, we refined abstract quality characteristics to specific base measures. In this section, we study these measures wrt. a business process modeling notation. As an example, we evaluate BPMN models because BPMN is an up to date and very wide-spread modeling notation for business processes. The primary goal of BPMN is to provide a notation that is readily understandable by all business users to bridge the gap between the business process design and the process implementation [OMG09]. Business process models in BPMN are called Business Process Diagrams (BPD).

A business process model typically captures structural process properties. However, some of the base measures require knowledge of the behavior during the execution of the process, e.g. such as the time needed to perform a certain activity or the number of failures occurred while performing the activity. We consider a measure as expressible by BPMN if it is directly identifiable in the BPD. If a measure needs additional information to be identified or if there is only an indicator for the measure expressible in the BPD, we consider that measure as not identifiable in BPMN.

BPMN is a useful means for modeling business processes but there are some aspects which are not well covered. The evaluation (cf. Table 1 and Table 3) showed that BPMN is not able to express all the measures summarized in the tables above. With respect to the behavior during the execution of the process there are few measures expressible by BPMN, for example, the number of handled exceptions or the number of callbacks. As a major deficit we consider that BPMN is not able to capture time values. As was expected, BPMN is able to express a lot of the measures which capture structural process properties like number of XOR decisions, number of loops or number of activities. But there are some measures of structural properties which cannot be represented. As a major deficit with respect to structural process properties we consider that BPMN does not provide model elements to express the process components resource and physical object. Thus, measures presented in Table 2 are not expressible in the BPD. Moreover, BPMN does not directly allow modeling important information with respect to model elements like skills of an actor[3] or expertise required for an activity (cf. Table 1 and Table 3). Only modeler-defined property attributes can be used to capture this information. However, this is not possible for actors.

Altogether, 11 of the 24 structural measures and 4 of the 27 behavior measures are expressible by BPMN. Furthermore, 14 of the 40 measures captured from practice are expressible by BPMN, while only 1 of the 12 measures adapted from ISO is expressible.

[3] An actor is represented by the BPMN model elements Pool or Lane [OMG09].

5 Related Work

While there is no standard definition of what contributes to business process quality, there are several publications concerned with the definition of quality. These are discussed in the following.

As mentioned above, the process check list in [Fi09] presents a collection of problems from practice which should be avoided. We assigned these practical problems to the ISO characteristics (cf. section 3.2). Therefore, we reworded, abstracted or stated the problems more precisely because they are presented in question form in the process check list. Thus, the process check list corresponds to a subset of our attributes and measures. In the process check list the problems are considered as time and cost aspects.

[GD05] presents a model for measuring information system effects on business process quality based on the ISO/IEC 9126. The characteristics provided in this approach are covered by our (sub-)characteristics. They only represent a subset of the ISO characteristics which was used in a specific study. The paper does not explain the reasons for this choice.

A recent approach for a comprehensive definition of business process quality is given in [HMR09]. This approach associates several quality dimensions, which are based on the ISO/IEC 9126-1 and other related work, with different components of a business process. Unfortunately, it is not explained which adaptations were made why, and why some of the ISO characteristics were left out. Furthermore, the dimensions are of different granularity. Mostly, the level of abstraction of these dimensions is comparable to the level of abstraction of (sub-)characteristics. As no attributes are given, it is often difficult to understand the meaning of the dimensions. For all these reasons, it is difficult in our view to base further work on the definitions. Therefore, we used a more systematic approach.

[ABP10] gives a first idea of refining abstract quality characteristics of process quality, data handling quality and IT support quality in healthcare processes by allocating specific measures. This approach uses the terms “quality category” and “indicator” to talk about business process quality. The level of abstraction of the quality categories is comparable to the level of abstraction of characteristics and the indicators are comparable to attributes or measures. We assigned these indicators from practice to characteristics, thus they are a subset of our attributes and measures.

Beyond that, there are a lot of publications on the topic of quality metrics for business process models. [Va07] provides an overview of existing literature on that topic. Quality metrics, usually, are based on what we call base measures or a combination of these. Because this publication represents a summary of publications there are few measures discussed in detail. [Me10] focuses on a subset of our characteristics by discussing the measuring of a person’s structural understanding of a business process model. This approach presents definitions which formalize the attributes concurrency, exclusiveness,

order and repetition by describing very specific measures for these aspects. Our measures focus particularly on attributes relevant to practice.

6 Conclusion

In this paper we presented a first attempt to define the quality of a business process systematically. We introduced consistent terminology by adapting software product quality standards. We presented the characteristics coherently in a meta-model which enables an easy overview. By looking at typical process problems we derived attributes and measures for the characteristics. This seems to indicate that only part of the characteristics and the ISO-derived measures are really relevant for business processes. We discussed how many of these measures are expressible by BPMN. This seems to indicate that important notations are missing in BPMN to capture practically relevant measures.

In our view this is a good basis for further research on business process quality. First we want to apply our measures to real processes and check how much effort it is to capture these measures. This will also help in deriving relevant complex measures. Furthermore, this will provide insight in the completeness of our characteristics, attributes and measures. We do not believe that it will be possible to come up with a complete set of measures, but we aim at a set of measures which gives important feedback on quality and which can be captured with adequate effort. Second we want to study other business process modeling notations and define ways to capture our measures in a business process model. Then again it is necessary to apply the new notation to real processes and thereby assess its benefits and drawbacks.

References

[ABP10] Ammenwerth, E.; Breu, R.; Paech B.: User-Oriented Quality Assessment of IT-Supported Healthcare Processes – a Position Paper, BPM 2009 Workshops, Lecture Notes in Business Information Processing Vol. 43, Springer, pp. 617–622 (2010).

[BP85] Ballou, D. P.; Pazer, H. L.: Modeling Data and Process Quality in Multi-Input, Multi-Output Information Systems, Management Science Vol. 31 No.2, INFORMS, pp.150-162 (1985).

[ES07] Even, A.; Shankaranarayana, G.: Utility-Driven Assessment of Data Quality, The DATA BASE for Advances in Information Systems – ACM SIGMIS Journal, Vol. 38 No. 2, pp. 75-93 (2007).

[Fi09] Fischermanns, G.: Praxishandbuch Prozessmanagement, Verlag Dr. Götz Schmidt, 7th revised edition (in German) (2009).

[GD05] Gucegliogl u, A. S.; Demirors, O.: Using Software Quality Characteristics to Measure Business Process Quality, BPM 2005, Lecture Notes in Computer Science Vol. 3649, Springer, pp. 374-379 (2005).

[HMR09] Heravizadeh, M.; Mendling, J.; Rosemann, M.: Dimensions of Business Process Quality

(QoBP), BPM 2008 Workshops, Lecture Notes in Business Information Processing Vol. 17, Springer, pp. 80-91 (2009).

[ISO01] ISO/IEC 9126-1: Software engineering — Product quality — Part 1: Quality model, First edition (2001).

[ISO05] ISO/IEC 25000: Software engineering — Software product Quality Requirements and Evaluation (SQuaRE) — Guide to SQuaRE, First edition (2005).

[ISO07] ISO/IEC TR 25021: Software engineering — Software product Quality Requirements and Evaluation (SQuaRE) — Quality measure elements, First edition (2007).

[ISO08] ISO/IEC 25012: Software engineering — Software product Quality Requirements and Evaluation (SQuaRE) — Data quality model, First edition (2008).

[KSW02] Kahn, B. K.; Strong, D. M.; Wang, R. Y.: Information quality benchmarks: Product and service performance, Communications of the ACM Vol. 45 No. 4, pp. 184-192 (2002).

[LC05] Lange, C. F. J.; Chaudron, M. R. V.: Managing Model Quality in UML-Based Software Development, 13th IEEE International Workshop on Software Technology and Engineering Practice (STEP'05), IEEE Computer Society, pp. 7-16 (2005).

[Le02] Lee, Y. W.; Strong, D. M.; Kahn, B. K.; Wang, R. Y.: AIMQ: A Methodology for Information Quality Assessment, Information and Management Vol. 40 Issue 2, Elsevier, pp. 133-146 (2002).

[Me10] Melcher, J.; Mendling, J.; Reijers, H. A.; Seese, D.: On Measuring the Understandability of Process Models, 1st Workshop on Empirical Research in BPM (ER-BPM 2009), BPM 2009 Workshops, Lecture Notes in Business Information Processing Vol. 43, Springer, pp. 41-52 (2010).

[MD09] Mohagheghi, P.; Dehlen, V.: Existing Model Metrics and Relations to Model Quality, ICSE Workshop on Software Quality (WoSQ 2009), International Conference of Software Engineering, IEEE Computer Society, pp. 39-45 (2009).

[MDN09] Mohagheghi, P.; Dehlen, V.; Neple, T.: Towards a Tool-Supported Quality Model for Model-Driven Engineering, Proceedings of the 3rd Workshop on Quality in Modelling (QiM'08) at MODELS 2008, Lecture Notes in Computer Science Vol. 5421, Springer, pp. 74-88 (2009).

[OMG09] OMG: Business Process Modeling Notation (BPMN) Specification, Version 1.2, Object Management Group (2009).

[Os87] Osterweil, L.: Software Processes are Software Too, Proceedings of the 9th International Conference on Software Engineering, IEEE Computer Society, pp. 2-13 (1987).

[PLW02] Pipino, L. L.; Lee, Y. W.; Wang, R.Y.: Data Quality Assessment, Communications of the ACM, Vol.45 No.4, pp. 211-218 (2002).

[Va07] Vanderfeesten, I.; Cardoso, J.; Mendling, J.; Reijers, H.A.; van der Aalst, W.M.P.: Quality Metrics for Business Process Models, In L. Fischer (editor), 2007 BPM & Workflow Handbook, Workflow Management Coalition, pp. 179-190 (2007).

[WS96] Wang, R. Y.; Strong, D. M.: Beyond accuracy: What data Quality means to data consumers, Journal of Management Information Systems Vol. 12 No. 4, M.E. Sharpe Inc, pp 5-34 (1996).

Wie die Objektorientierung relationaler werden sollte

Eine Analyse aus Sicht der Datenmodellierung

Dilek Stadtler[1] und Friedrich Steimann[2]

Abstract: Die Kluft zwischen der relationalen Sichtweise auf Daten, wie sie u. a. in UML-Klassendiagrammen ausgedrückt werden kann, und der objektorientierten Programmierung ist seit langem Gegenstand von Untersuchungen. Angesichts jüngster Bestrebungen, relationale Elemente in die objektorientierte Programmierung einzubringen (wie etwa mit Microsofts LINQ-Projekt), zeigen wir auf, wie das Modell der objektorientierten Programmierung erweitert werden müßte, so daß sich auch die relationalen Teile eines Klassendiagramms direkt, d. h. unter Verzicht auf Transformationen, die umfangreichen stereotypen Code einführen, in objektorientierte Programme umsetzen lassen, ohne daß das objektorientierte Programmiermodell dadurch seinen navigierenden Charakter verlieren würde. Dabei beschränken wir uns hier auf die Ableitung der Anforderungen an eine solche Erweiterung — eine konkrete Umsetzung ist in einer parallelen Arbeit beschrieben.

1 Einleitung

Vor dem Aufkommen der objektorientierten Modellierung und ihrer Notationen dominierte für geraume Zeit das Entity-Relationship-Modell (ERM) [Ch76] die Datenmodellierung. Seine Vorteile sind die direkte Übersetzbarkeit in das relationale Datenmodell (RDM, einschließlich seiner mathematische Fundierung, dem Relationenkalkül [Co70]) sowie die generelle Anwendungsunabhängigkeit seiner Modelle: Mit der Vorgabe der Entitäts- und Beziehungstypen[3] wird lediglich ein logischer Sachverhalt wiedergegeben, der von beliebigen Anwendungen beliebig genutzt werden kann. Insbesondere sind durch Relationen keine Navigationsrichtungen vorgesehen, so daß Daten mittels relationaler Abfragesprachen in jede Richtung ausgewertet werden können.

Mit dem Aufkommen der objektorientierten Programmierung begann sich die Sicht auf die Daten wieder zu ändern. Zwar basiert die Objektorientierung auf Klassen, deren Datenanteil im wesentlichen den Entitätstypen des ERM entspricht, aber die Beziehungen zwischen Klassen werden durch Zeiger dargestellt und sind damit grundsätzlich gerichtet. Das objektorientierte Datenmodell (OODM), welches als die Projektion des objektorientierten Programmiermodells auf seine Datenanteile verstanden werden kann

[1] Lehrgebiet Programmiersysteme, Fakultät für Mathematik und Informatik, Fernuniversität in Hagen, dilek.stadtler@fernuni-hagen.de

[2] Lehrgebiet Programmiersysteme, Fakultät für Mathematik und Informatik, Fernuniversität in Hagen, steimann@acm.org

[3] Wir übersetzen hier „Relationship“ mit „Beziehung“. Zu Codds Unterscheidung der Begriffe „Relation“ und „Relationship“ s. Abschnitt 3.3.1.

(also im wesentlichen als Klassen mit Attributen, ohne Methoden, jedoch mit Vererbung), ist damit aber näher am (überkommen geglaubten) Netzwerkmodell als am ERM, was auch daran erkennbar ist, daß die Auswertung der Daten typischerweise nicht mehr deklarativ wie im relationalen Fall erfolgt, sondern imperativ mittels Methoden.

Seither gibt es immer wieder Bestrebungen, den Rückschritt, den die Objektorientierung in Sachen Datenmodellierung bedeutet, zu korrigieren und das OODM um Relationen zu

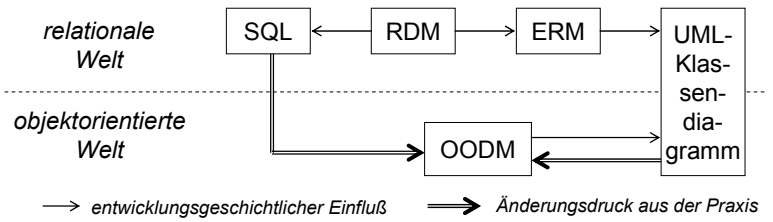


Abb. 1: Beziehung der in dieser Arbeit herangezogenen Modelle und deren Notationen. Wir leiten aus dem Änderungsdruck unter Berücksichtigung der Entwicklungsgeschichte eine sanfte Erweiterung des OODMs um relationale Aspekte ab.

erweitern (s. z.B. [BW05, NPN08, Øs07, Ru87]). Wir sehen hierfür vor allem zwei Beweggründe (zum ursächlichen, entwicklungsgeschichtlichen Zusammenhang s. Abbildung 1):

1. Wichtige objektorientierte Modellierungssprachen wie die UML setzen in ihren statischen Strukturbeschreibungen (das sind im wesentlichen Klassendiagramme) auf Relationen als neben Klassen gleichberechtigte Modellierungskonstrukte [OMG09a, OMG09b]. Daß sich diese in objektorientierten Programmen nicht unmittelbar wiederfinden, führt zu Problemen beim Übergang von Modellen zu Programmen und umgekehrt (s. z.B. [AHM07, Ge09, GRL03, No97]).

2. Die Datenhaltung erfolgt aus technischen Gründen derzeit noch vor allem mittels (objekt)relationaler Datenbanken (s. z. B. [MM95]). Selbst wenn die Probleme der transparenten Übertragung von Objekten aus objektorientierten Programmen in solche Datenbanken inzwischen weitgehend gelöst sind, so stören sich aus der relationalen Datenbankwelt kommende Programmierer doch an imperativ ausformulierten Datenauswertungen auf der Programmseite und vermissen die Prägnanz relationaler Abfragesprachen wie der SQL [ISO08].

In dieser Arbeit versuchen wir herauszuarbeiten, welche relationalen Ergänzungen des OODMs durch diese Argumente tatsächlich begründet werden können. Wie sich herausstellen wird, lassen sich die in der Praxis wichtigsten Anforderungen in eine vergleichsweise behutsame Erweiterung des OODMs transformieren, die insbesondere dessen zeigerbasierten Charakter voll erhält. Die Umsetzung dieser Erweiterung zeigen wir aufgrund ihrer Ferne zur Modellierung hier jedoch nicht; sie ist in einer parallelen Arbeit beschrieben [SS10].

Der Rest dieser Arbeit ist wie folgt gegliedert: In Abschnitt 2 diskutieren wir kurz die verwandten Arbeiten, bevor wir in Abschnitt 3 untersuchen, welche Elemente relationalen Ursprungs das UML-Klassendiagramm enthält, die nicht zugleich im OODM vorkommen, und welche Relevanz diese Elemente in der Praxis haben. In Abschnitt 4 untersuchen wir, wie viel „Relationalität" relationale Abfragesprachen vom OODM verlangen und stellen fest, daß dies lediglich eine Anforderung, die schon eine relationale Datenmodellierung an das OODM stellt, bestärkt. Wir schließen mit der Übertragung der Anforderungen in die Skizze einer sanften Erweiterung des OODMs, die einen guten Teil der Erwartungen an ein „OODM mit Relationen" bedienen können sollte.

2 Verwandte Arbeiten

Daß dem OODM Relationen fehlen, ist in der Literatur weitgehend unbestritten. Auch bei den Arbeiten dazu, wie sie eingebracht werden können, herrscht große Einigkeit: Sie lassen sich grob in solche einteilen, die Relationen mit stereotypen Codefragmenten emulieren (dazu gehören auch die Transformationsansätze aus dem Model-driven-Bereich), und in solche, die das OODM erweitern wollen (wozu auch die unsere zählt).

Noble beschreibt gleich eine ganze Reihe von Mustern zur Umsetzung von allgemeinen Relationen mit Hilfe der durch objektorientierte Programmiersprachen vorgegebenen Mittel [No97]. Gängiger Programmierpraxis folgend werden bei ihm unidirektionale und bidirektionale 1:1- und 1:*n*-Beziehungen mit Hilfe von Attributen und Collections umgesetzt. Für komplexere Beziehungen schlägt [No97] die Realisierung anhand von „relationship objects", also der Reifizierung der Beziehungen als Klasse/n, vor.

Einen Schritt weiter gehen Genova et al., indem sie einen Generator für die Umsetzung von binären UML-Assoziationen (inkl. Multiplizität, Sichtbarkeit und Navigierbarkeit) in Java-Codemuster vorstellen [GRL03]. Die Muster basieren auf einer Reifizierung von Beziehungen als Klassen („reified tuples"). Andere Konzepte wie z.B. höherstellige Beziehungen, Assoziationsklassen etc. werden jedoch nicht behandelt.

Auch Akehurst et al. betrachten die Problematik aus Sicht der MDA, fokussieren in ihrer Arbeit aber besonders auf in UML 2.0 neu definierte Konzepte und deren Realisierung in den neuen Sprachkonstrukten von Java 5 [AHM07]. Über andere Arbeiten hinausgehend schlagen sie auch eine Umsetzung von qualifizierten Assoziationen und eine Möglichkeit zur Einhaltung von unteren Schranken (Multiplizitäten) vor. Gessenharter geht noch einen Schritt weiter und löst das Problem der Kombination von Navigierbarkeit und Sichtbarkeit [Ge09]. Er widerlegt damit die Behauptung von [GRL03], daß dies mit reifizierten Beziehungen nicht möglich sei. Auf einer höheren Ebene befaßt sich die Arbeit von Amelunxen et al. mit der Umsetzung von Assoziationen in Java, indem sie die Sicht des Metamodells MOF 2.0 ein- und sich da insbesondere die assoziationsabhängigen Konzepte Union, Redefinition und Subset vornimmt [ABS04].

Während all diese Arbeiten im wesentlichen Relationen aus (mentalen oder reellen) Modellen in objektorientierte Codefragmente transformieren, die vom Programmierer später weiterzupflegen sind, erlauben es Bibliotheken wie die von Østerbye [Øs07], Relationen — analog zu Collections — als Klassen in ein Programm zu importieren. Allerdings unterliegt die Verwendung solcher Bibliotheksrelationen gewissen Konventionen, die wiederum zu stereotypem Code führen. Insbesondere lassen sich damit programmiersprachliche Primitive wie beispielsweise die Zuweisung nicht modifizieren.

Diese Beschränkung vermeiden Spracherweiterungen wie beispielsweise die von Schmidt et al., Rumbaugh oder die von Bierman und Wren. Schmidt et al. haben in verschiedenen Arbeiten den Einsatz von relationalen Datenbankprogrammiersprachen (Sprachen, in denen programmiersprachliche und datenbanksprachliche Konzepte miteinander verschmolzen werden), die das Relationenmodell in ihr Typsystem integrieren, untersucht. Bedeutende Arbeiten in diesem Bereich sind beispielsweise Pascal/R [Sc77] und DBPL [SM92]. Beide Sprachen haben gemeinsam, das Konzept der Relation im Sinne einer Datenbankrelation (mitsamt der Definition von Schlüsseln) als eigenen Datentyp in eine höhere Programmiersprache zu integrieren. Die jeweilige Programmiersprache wird dabei in vergleichsweise hohem Maße erweitert. Einen ebenfalls von Schmidt et al. entwickelten, bibliotheksbasierten Ansatz, stellt das Tycoon-System [MS93] dar. Das Tycoon-System ist eine Programmierumgebung zur Konstruktion von persistenten Objektsystemen, welche, je nach Version, auf der strikt typisierten, polymorphen Programmiersprachen Tycoon Language (TL) [MS92] oder der objektorientierte Variante, Tycoon Object-Oriented Language (TooL) [GM96], basiert. Mit diesem Ansatz wurde ein System zur Verfügung gestellt, dessen Fokus auf der bestmöglichen Erweiterbarkeit eines minimalen vordefinierten Sprachkerns liegt, und somit die Möglichkeit bietet, Datenbank-Funktionalitäten auf unterschiedlichste Weise zu integrieren.

Rumbaugh [Ru87] und Bierman und Wren [BW05] führen dagegen jeweils Relationen als native programmiersprachliche Konstrukte auf gleicher Ebene mit Klassen ein. Die beiden Arbeiten repräsentieren dabei zwei grundsätzlich verschiedene Auffassungen von Relationen als Typen: Für Rumbaugh sind Relationen Tupelmengen, die in Instanzen eines Typs `Relation` gehalten werden, während für Bierman und Wren Relationen die Typdeklarationen von Tupeln sind (und die Instanzen von Relationen entsprechend einzelne Tupel). Beiden Arbeiten gemein ist wiederum, daß Tupel an zentraler Stelle, also insbesondere außerhalb der Objekte, die ihre Komponenten ausmachen, gehalten werden. Im Gegensatz dazu favorisieren wir einen Ansatz, bei dem Tupel an die Komponenten gebunden sind. Die globale Extension einer Relation ergibt sich damit als die Vereinigung der Tupel aller beteiligten Objekte.

3 Notwendige relationale Ergänzungen des objektorientierten Datenmodells aus Modellierungssicht

In diesem Abschnitt versuchen wir, mögliche Defizite des OODMs aus Modellierungssicht systematisch herzuleiten. Wir gehen zu diesem Zweck davon aus, daß das UML-Klassendiagramm (begründet durch den entwicklungsgeschichtlichen Einfluß; s. Abbildung 1) ein weitgehend konsensuelles Datenmodell repräsentiert, daß es also insbesondere was die Modellierung mit Relationen angeht, nichts Wesentliches vermissen läßt, und setzen die Modellierungssicht auf objektorientierte Daten mit dem UML-Klassendiagramm gleich. Diese Annahme erlaubt uns, aus dem Aufbau des Diagramms (richtiger: des Diagrammtyps) sowie der Herkunft seiner Elemente entweder aus dem ERM (als Stellvertreter einer relationalen Sichtweise) oder aus dem OODM[4] systematisch abzuleiten, was dem OODM in seiner Datensicht an Relationalem möglicherweise fehlen könnte. Ob es ihm tatsächlich fehlt, versuchen wir anschließend durch die Einbeziehung der Modellierungspraxis zu klären und durch Betrachtungen, inwieweit das OODM ohne Erweiterungen in der Lage ist, für brauchbaren Ersatz zu sorgen. Doch bevor wir uns diesen Fragestellungen zuwenden, rekapitulieren wir kurz das ERM und das OODM.

3.1 Das Entity-Relationship- und das objektorientierte Datenmodell

Das ERM besteht in seiner ursprünglichen Form aus Entitätstypen, Beziehungstypen, Attributdeklarationen für beide sowie der Angabe von Kardinalitäten für Attribute die Stellen eines Beziehungstyps. Entitätstypen werden unterschieden in starke und schwache: Erstere sind durch eine Menge von Attributen als Primärschlüssel eindeutig identifizierbar, letztere nicht. Ergänzen läßt sich das ERM durch eine Reihe von generischen Operationen zum Einfügen, Ändern und Löschen von Daten.

Auch wenn das ERM vielleicht nicht als speziell auf das RDM Codds [Co70] zugeschnitten erscheinen sollte, so weist es doch starke Ähnlichkeiten mit ihm auf. Gleichwohl ist das ERM u. a. aufgrund seiner Unterscheidung von Entitäts- und Beziehungstypen differenzierter als das RDM — tatsächlich kann das RDM aus Modellierungssicht sogar als unzureichend angesehen werden, weil es nicht zwischen Entitäts- und Beziehungstypen unterscheiden kann und damit eine fundamentale semantische Differenzierung nicht auszudrücken erlaubt. Wenn wir also im folgenden davon sprechen, daß die Objektorientierung relationaler werden sollte, meinen wir damit immer relationaler im Sinne des ERMs und nicht im Sinne des puristischen RDMs.

[4] Genaugenommen gibt es *das* OODM gar nicht — zumindest ist eine den anderen Datenmodellen vergleichbar allgemein akzeptierte Definition in der Literatur nicht zu finden. Wie bereits weiter oben beschrieben, kann es jedoch im wesentlichen als Projektion des objektorientierten Programmiermodells auf seine Datenanteile verstanden werden.

Dem ERM gegenüber steht das OODM. Seine Klassen entsprechen zwar für sich genommen im wesentlichen den Entitätstypen des ERMs (zumal sie wie letztere über Attribute zur Beschreibung ihrer Instanzen, Objekte genannt, verfügen), jedoch kennt das OODM anders als das ERM keine Relationen: Beziehungen zwischen Objekten werden ausschließlich über die Attribute hergestellt, wobei die Attribute zu diesem Zweck Zeiger auf Objekte enthalten. In gewisser Weise leidet das OODM damit am selben Defizit wie das RDM (nur daß hier Relationen gegen Klassen und Primär- bzw. Fremdschlüssel gegen Objektidentitäten bzw. Zeiger getauscht sind): Ihm fehlt eine fundamentale semantische Abstraktion. Generische Operationen des OODMs sind die Instanziierung von Klassen in Objekte, die Wertzuweisungen an Attribute und das Löschen von Objekten.

3.2 Notationselemente des UML-Klassendiagramms

Ein UML-Klassendiagramm kann die folgenden primären (unabhängigen) Notationselemente enthalten:

1. *Klasse* [OMG09a, §7.3.7],
2. *Assoziation* (wobei zweistellige besonders häufig vorkommen und aus diesem Grund über eine vereinfachte Notation verfügen) [§7.3.3],
3. *Assoziationsklasse* (als Spezialisierung von *Klasse* und *Assoziation*) [§7.3.4],
4. *Generalisierung* [§7.3.20],
5. *Abstraktion* [§7.3.1] (mit Spezialfällen *Realisierung*, *Substitution* [§7.3.45, §7.3.50]),
6. *Abhängigkeit* [§7.3.12] (mit dem Spezialfall *Verwendung* [§7.3.53]) sowie
7. *Schnittstelle* [§7.3.24] (tritt typischerweise in Zusammenhang mit *Interface Realization* [§7.3.25], einem Spezialfall von *Realisierung* und *Verwendung*, auf).

Klasse ist nominal ein Element des OODMs, faktisch aber genausogut eines des ERMs, wo es allerdings *Entitätstyp* heißt. *Assoziation* findet man als Modellierungselement im ERM (wo es *Beziehungstyp* oder engl. *relationship type* heißt), jedoch nicht im OODM (s. o.): hier müssen Beziehungen durch Attribute ausgedrückt werden, was aber zunächst, da ein Attribut immer nur auf ein Objekt verweisen kann, auf zweistellige, gerichtete Zu-1-Beziehungen beschränkt bleibt (mehr dazu unten). *Assoziationsklasse* ist ein Konzept des ERM, dort aber nicht von Beziehungstypen zu unterscheiden (Beziehungstypen können im ERM genau wie Entitätstypen Attribute deklarieren) und kommt im OODM nur dann vor, wenn man als Klassen reifizierte Relationen als eigenständiges Modellierungskonstrukt betrachtet (mehr zur Reifizierung von Relationen in Abschnitt 3.3.4). Instanzen von *Generalisierung*, *Abstraktion* und *Abhängigkeit* definieren im Gegensatz zu *Assoziation* (deren Instanzen Beziehungen zwischen den Elementen der an einer Assoziation beteiligten Klassen deklarieren) Instanzen von mit der Definition der UML vorgegebenen Relationen auf der Menge der Klassen; sie stehen damit aus model-

lierungstechnischer Sicht eine Ebene über den Assoziationen und haben somit keine Beziehung zu relationalen Modellen.[5] Das gleiche gilt im Ergebnis für *Schnittstelle*. Es bleibt also, daß von den Diagrammelementen auf oberster Ebene lediglich *Assoziation* und *Assoziationsklasse* Modellierungskonstrukte repräsentieren, denen es im OODM (im Gegensatz zum ERM als Stellvertreter einer relationalen Sichtweise) an Entsprechung mangelt.

Von den beiden für unsere Belange relevanten primären Diagrammelementen *Assoziation* und *Klasse* (letztere, weil sie Teile der Definition von *Assoziationsklasse* enthält) abhängig sind eine Reihe weiterer definiert, die die primären näher bestimmen oder einschränken. Es sind dies[6]

1. für *Klasse*:
   a) ein *Name* zur eindeutigen Identifizierung,
   b) optional *Attribute*, deren Werte die Zustände der Instanzen der Klassen codieren,
   c) optional *Operationen*, die das Verhalten der Instanzen einer Klasse definieren,
   d) optional die Auszeichnung als *abstrakte Klasse*,
   e) optional die Auszeichnung als *aktive Klasse*;
2. für *Assoziation*:
   a) optional eine *Name*,
   b) optional eine *Leserichtung*,
   c) mindestens zwei *Assoziationsenden*,
   d) optional die Auszeichnung als *abgeleitete Assoziation* sowie
   e) optional eine Menge von vordefinierten *Eigenschaften* (Property Strings) die Eigenschaften der Assoziation ausdrücken (wie beispielsweise *ordered*, *subsets*,…).

Von diesen sind alle Elemente, die sich mit Verhalten befassen (*Operation*, *aktive Klasse* etc.), definitionsgemäß weder Bestandteil des ERMs noch des OODMs und können daher für weitere Betrachtungen entfallen. Von den verbleibenden im Fall *Klasse* entspricht *Name* dem Vorkommen im OODM und im ERM gleichermaßen. *Attribute* kommen zwar ebenfalls in beiden vor, unterscheiden sich aber, was deren Semantik angeht: Während im ERM Attribute stets Wertsemantik haben, können sie im OODM Wert- oder Referenzsemantik haben, also insbesondere auch Verweise auf andere Objekte enthalten. Das UML-Klassendiagramm folgt hier dem OODM. *Abstrakte Klasse* ist Zusammenhang mit *Generalisierung* zu sehen (s.o.) und deswegen hier ohne Bedeutung.

[5] Man beachte, daß in sog. semantischen Datenmodellen die Generalisierung häufig als besondere Relation auf gleicher Ebene mit der Aggregation eingeführt wird; dies ist ein Fehler.

[6] Elemente, die überall vorkommen können und für das Klassendiagramm keine spezifische Bedeutung haben, sind hier weggelassen. Dazu zählen Stereotype, Kommentare und Constraints.

Im Fall der *Assoziation* wird die Sache interessanter: Da es im OODM keine Assoziationen gibt, handelt es sich bei *Name* um die Namen der Relationen aus dem ERM. Das gleiche gilt im Prinzip für *Leserichtung* und *Assoziationsenden*, auch wenn es diese nominal im ERM nicht gibt: *Leserichtung*, wo notwendig, ergibt sich implizit aus Rollennamen (s.u.), *Assoziationsenden* entsprechen den Stellen der Beziehungstypen. *Abgeleitete Assoziation* ist vor allem im Zusammenhang mit Vererbung zu sehen und bringt keinen neuen relationalen Aspekt ein.

Von den obengenannten, von *Klasse* und *Assoziation* abhängigen Notationselementen können wiederum weitere abhängig sein. Es sind dies für

1. *Klasse*

   b) *Attribute*:

      i. ein *Name* zur Identifikation,

      ii. optional eine *Sichtbarkeit*,

      iii. optional ein *Typ*,

      iv. optional eine *Multiplizität*,

      v. optional eine Menge von vordefinierten *Eigenschaften* (Property Strings), die Eigenschaften des Attributs ausdrücken (bspw. *ordered*, *read only*, ...),

      vi. optional die Auszeichnung als *abgeleitetes Attribut*,

      vii. optional die Auszeichnung als *Klassenattribut* (durch Unterstreichung) und

      viii. optional ein *Ausdruck*, der den Default-Wert des Attributs darstellt; und für

2. *Assoziation*

   c) *Assoziationsende*:

      i. optional ein *Rollenname*,

      ii. optional eine *Navigierbarkeit* und, davon abhängig, *Navigationsrichtung*,

      iii. eine optionale *Qualifizierung*,

      iv. optional eine *Multiplizität*,

      v. optional eine Menge von *vordefinierten Eigenschaften* (Property Strings), die Eigenschaften der Assoziation bezogen auf das Assoziationsende ausdrücken (wie beispielsweise *ordered*, *read only*, …),

      vi. bei binären Assoziationen optional eine *Aggregation* oder *Komposition*,

      vii. optional eine *Sichtbarkeit*.

Die auf Attribute von Klassen bezogenen Diagrammelemente 1 b) i–viii können allesamt dem OODM zugerechnet werden (wobei *Name*, *Typ* und *Multiplizität* auch im ERM

vorkommen). Man würde vielleicht erwarten, daß im Gegenzug die auf Assoziationsenden bezogenen Diagrammelemente 2 c) i–vii relationalen Ursprungs sind — tatsächlich sind sie das aber bis auf *Rollenname*, *Multiplizität*, *Qualifizierung* und (wenn man Erweiterungen des ERM hinzunimmt) *Aggregation*/*Komposition* nicht. Es fällt vielmehr auf, daß sie mit *Sichtbarkeit*, *Multiplizität* und den vordefinierten *Eigenschaften* Elemente aufweisen, die denen der Attribute entsprechen. Mit *Navigierbarkeit* und *Navigationsrichtung* werden zudem zwei Elemente eingebracht, die dem ERM und seinen Beziehungstypen fremd sind und die vielmehr einen Zeigercharakter von Beziehungen nahelegen.

3.3 Mögliche Kompensation der relationalen Defizite im OODM

Bei der Analyse des UML-Klassendiagramms haben wir einige Notationselemente identifiziert, deren Ursprung relational ist und die keine direkte Entsprechung im OODM haben. Dabei hatte sich jedoch schon angedeutet, daß die Tatsache, daß sie darin nicht vorkommen nicht heißt, daß sie darin nicht ausgedrückt werden können — genauso, wie das RDM Klassen mit Relationen emuliert, kann vielleicht auch das OODM Relationen mit Bordmitteln umsetzen. Inwieweit dies möglich ist, soll im folgenden geklärt werden.

3.3.1 Mangelnde bidirektionale Navigierbarkeit

Im Gegensatz zu Relationen, die grundsätzlich in alle Richtungen navigierbar sind, sind Zeiger grundsätzlich nur in eine Richtung navigierbar. Insofern ist auch die häufig angetroffene mathematisch Interpretation von Zeigern als Funktionen, die ein Objekt, auf ein anderes abbilden (s. z. B. [BO97]), nicht ganz korrekt: Während eine Funktion zumindest im Prinzip immer umkehrbar ist (auch wenn die Umkehrung nur noch eine Relation und keine Funktion mehr ist), so daß man vom Bild zu seinen Urbildern gelangen kann, gilt das für einen Zeiger nicht — während der Zeiger als Attribut der Quelle dieser fest zugeordnet ist, weiß das Ziel noch nicht einmal, daß auf es gezeigt wird, geschweige denn, welche Quellen dies tun. Allerdings gilt für Zeiger wie für Funktionen, daß sie rechtseindeutig sind, daß also zu jeder Quelle höchstens ein Ziel gehört.

3.3.2 Darstellung von Beziehungen zu Mehreren und ihre Grenzen im OODM

Vernachlässigen wir zunächst die mangelnde bidirektionale Navigierbarkeit von Zeigern bzw. Attributen, die Beziehungen darstellen sollen, dann bleibt immer noch das Problem, daß sich Beziehungen von einer Quelle zu mehreren Zielen grundsätzlich nur über mehrere Zeiger realisieren lassen. Da Zeiger aber (als Attribute) benannt sind, ergibt sich daraus ein praktisches Problem, nämlich daß man schlecht eine variable Anzahl von Namen für eine variable Anzahl von mit einem Objekt in Beziehung stehenden anderen Objekten vorsehen kann (vgl. dazu auch [No97]). Dieses Problem kann man (wie beispielsweise in Smalltalk geschehen [GR83]) mittels sog. indizierter Instanzvariablen

lösen, also mit Attributen, auf deren Inhalt nicht über einen Namen (den Namen des Attributs), sondern über einen Index zugegriffen wird (Abbildung 2). Dieser Index muß keine natürliche Zahl, sondern kann jeder beliebige Wert und sogar ein Objekt sein, so daß sich damit sogar (der Zugriff über) ein qualifiziertes Assoziationsende umsetzen läßt. Da jedes Objekt jedoch nur ein namenloses, indiziertes Attribut haben kann, kann das Objekt auch nur in höchstens einer Zu-*n*-Beziehung (also derselben Beziehung zu mehreren anderen Objekten) stehen. Das jedoch ist unrealistisch, so daß Zu-*n*-Beziehungen in der Regel über eine Art Zwischenobjekte, also Objekte, deren einziger Zweck es ist, mehrere andere Objekte aufzunehmen (z. B. in indizierten Instanzvariablen), umgesetzt werden, wobei dann ein (benanntes) Attribut im Quellobjekt auf ein Zwischenobjekt und das Zwischenobjekt auf die *n* Objekte des Ziels verweist (Abbildung 2). Das aber bedeutet nicht nur eine Indirektion beim Zugriff, sondern führt auch dazu, daß man als Navigator immer wissen muß, ob es sich bei einer durch ein Attribut repräsentierten Beziehung um eine Zu-1- oder eine Zu-*n*-Beziehung handelt, da man im ersten Fall direkt zum in Beziehung stehende andere Objekt gelangt, im zweiten Fall hingegen direkt nur zum (nur zu Hilfszwecken eingeführten) Zwischenobjekt.[7]

Damit aber nicht genug, denn auch die Möglichkeiten zur Umsetzung von Zu-1-Beziehungen sind nicht vollständig befriedigend: Während Beliebig-zu-1-Beziehungen kein Problem sind, ja wegen der Zeigersemantik von Attributen sogar immer angenommen werden müssen (es können beliebig viele Zeiger auf dasselbe Objekt verweisen), lassen sich *n*:1-Beziehungen mit fester oberer Schranke *n* praktisch kaum durchsetzen. Allenfalls 1:1-Beziehungen ließen sich noch erzwingen, indem man Attribute mit Wertsemantik (die also das Objekt selbst und nicht einen Zeiger enthalten) verwendet und man die mangelnde Flexibilität (Objekte eines Subtyps sind in der Regel nicht mehr zuweisbar) sowie eine weitere Anomalie (Zielobjekte von 1:1-Beziehungen haben Wertsemantik) in Kauf nimmt; alles darüber hinaus fällt jedoch in das Gebiet der Alias-Kontrolle und liegt damit außerhalb der Ausdrucksmöglichkeiten heute gängiger objektorientierter Programmiermodelle (und damit auch des OODMs; vgl. Fußnote 2).

3.3.3 Darstellung von Bidirektionalität durch paarige Attribute

Die oben erwähnte mangelnde Umkehrbarkeit von Zeigern läßt sich natürlich durch paarige Zeiger, also Zeiger, die paarweise auftreten und so miteinander gekoppelt sind, daß der eine immer in die Gegenrichtung des anderen zeigt, kompensieren. Für 1:1-Beziehungen ist das direkt möglich, für alle anderen über indizierte Attribute oder per Umweg über Zwischenobjekte. Was dabei jedoch berücksichtigt werden muß, ist die Bedingung, daß zu jedem Zeiger von einem Objekt zu einem anderen auch ein Zeiger vom anderen genau zum Ausgangsobjekt existieren muß — es reicht dazu nicht, im Mo-

[7] In typisierten objektorientierten Programmiersprachen äußert sich diese Anomalie übrigens auch darin, daß die Instanzvariable im ersten Fall mit dem Typ des Ziels der Beziehung deklariert ist, im zweiten aber mit dem Typ des Zwischenobjekts (meistens ein Subtyp von `Collection` o. ä.), dem bestenfalls der Typ des Ziels als Typparameter beigeordnet ist.

dell einen Zeiger von Quelle zu Ziel und einen von Ziel zu Quelle verweisen zu lassen, denn die Ausprägungen könnten dann ja immer noch auf verschiedene Objekte derselben Klassen zeigen. Diese Bedingung muß insbesondere bei Zuweisungen an eine Instanzvariable beachtet werden, da sie hierbei verletzt werden kann (ja bei sequentieller Ausführung temporär immer verletzt wird), aber auch beim Löschen einzelner Objekte.

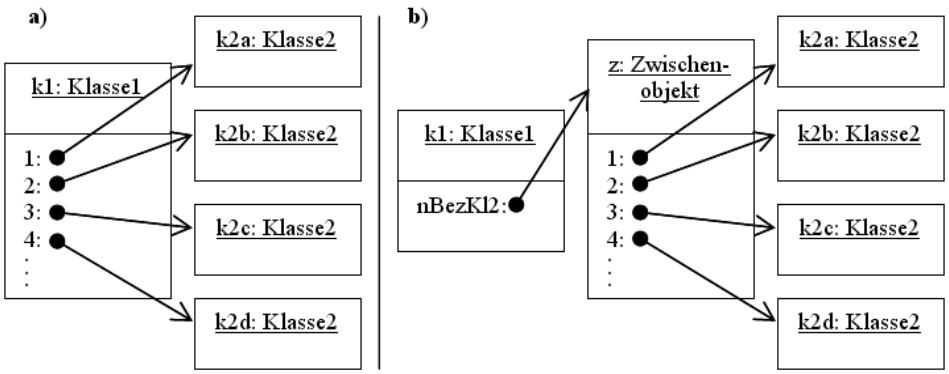


Abb. 2: Unterschiedliche Umsetzungen von Zu-n-Beziehungen im OODM: a) Indizierte Instanzvariablen mit natürlicher Zahl als Index. b) Umsetzung mit Hilfe eines Zwischenobjekts: Das Attribut mit dem Bezeichner nBezKl2 verweist auf das Zwischenobjekt z und ermöglicht dadurch eine 1-n-Beziehung zw. Objekten von Klasse1 und Klasse2.

Mit Hilfe paariger Attribute läßt sich auch auf einfache Weise die Einhaltung der Multiplizitäten beliebiger Beziehungen sicherstellen, da es nun keine Zeiger mehr gibt, von denen ein Objekt nichts weiß (vgl. oben) — es besitzt schließlich selbst Zeiger auf alle Objekte, die auf es verweisen. Diese Zeiger können leicht gezählt und ihre Anzahl damit kontrolliert werden.

3.3.4 Darstellung von höher als zweistelligen Relationen und von Assoziationsklassen

Da Zeiger nur eine Quelle und ein Ziel haben können, sind sie prinzipiell schlecht geeignet, höher als zweistellige Relationen umzusetzen. Man kann hier höchstens mehrere Stellen einer Relation zusammenfassen und gemeinsam (als Tupel) zu Quelle oder Ziel eines Zeigers machen. Die so entstehende geschachtelte Relation ist jedoch semantisch nicht dasselbe wie die Ausgangsrelation, so daß dieser Ansatz hier nicht weiter verfolgt werden soll. Es bleibt dann nur, die Ausgangsrelation zu reifizieren, also in eine Klasse umzuwandeln, die über zweistellige Beziehungen mit den Klassen der Ausgangsrelation verbunden ist (s. z. B. [No97]). Man verliert dadurch die direkte Navigierbarkeit zwischen diesen Klassen, eine Situation, die man allerdings auch schon von der Umsetzung von Zu-*n*-Beziehungen her kennt. Anders als Zwischenobjekte haben reifizierte Relationen aber eine semantische Entsprechung im Modell — sie stehen für ein bestimmtes

Konzept des modellierten Gegenstandsbereichs und es ist in der Tat die Frage, ob die Repräsentation dieses Konzeptes als Klasse weniger adäquat ist als die als Relation. Ein anderes als ein ästhetisches Defizit läßt sich aus dem Fehlen einer direkten Umsetzbarkeit mehrstelliger Relationen daher aus unserer Sicht nicht ableiten.

Ähnliches gilt im wesentlichen für die Umsetzung von Assoziationsklassen, also von Assoziationen, die selbst über Attribute verfügen, die nicht sinnvoll einer der beteiligten Klassen zugewiesen werden können — auch hier bleibt, da Zeiger keine Attribute haben können, nur die Reifizierung als Klassen. Diese ist insbesondere dann auch semantisch sinnvoll, wenn die Assoziationsklassen neben Attributen auch noch Verhalten aufweisen. Beziehungen mit eigenem Verhalten kann man aber konzeptuell auch als Kollaborationen auffassen, die jedoch im relationalen Kontext nicht auftauchen und deswegen hier nicht Gegenstand weiterer Betrachtungen sein sollen.

3.4 Bedeutung der Defizite für die Praxis

Nachdem die relationsbezogenen Defizite des OODM klar sind, stellt sich die Frage, welche Bedeutung diese für die Praxis haben. Rumbaugh berichtet, daß drei- und höherstellige Relationen in der Praxis nur selten vorkommen [Ru87]. Dies deckt sich mit den Erfahrungen der Autoren dieser Arbeit, die das OODM deswegen davon unberührt lassen, zumal die Reifizierung von solchen Relationen als Klassen durchaus eine gangbare Lösung ist. Auch aus dem Vorkommen von Assoziationsklassen in Modellen leiten sie keinen Zwang ab, Relationen als eigenständige Konstrukte in das OODM aufzunehmen, da auch hier Reifizierungen eine sinnvolle Lösung zu sein scheinen.

Was die zweistelligen Beziehungen angeht, ergibt sich in der Praxis ein differenziertes Bild. Die fehlende Kontrolle der Multiplizität *m* von unidirektionalen *m*:*n*-Beziehungen scheint durchaus verschmerzbar, zumal andere feste Obergrenzen als 1 häufig willkürlich erscheinen (selbst das oft bemühte Auto mit seinen vier Rädern ist als Beispiel angreifbar) — nötigenfalls muß hier eine bidirektionale Beziehung eingeführt werden, um die Multiplizität zu kontrollieren. Die Umsetzung von Zu-*n*-Beziehungen über Zwischenobjekte ist sicher lästig, aber im wesentlichen ein Implementierungsdetail, das hinter einer entsprechenden Abstraktion verborgen werden könnte (genauso, wie die Implementierung eines relationalen Datenbanksystems für effizienten Zugriff Indizes verwendet, die aber, abgesehen vielleicht von der Schemadefinition, nirgends explizit auftauchen). Vielmehr muß man anerkennen, daß die Dereferenzierung eines Zeigers, die die Navigation einer Zu-1-Beziehung im OODM bedeutet, sowohl aus programmiersprachlicher als auch aus implementierungstechnischer Sicht so effektiv ist, daß man nicht ohne Not darauf verzichten sollte. Was man sich vielmehr wünschen sollte, ist, daß Zu-*n*-Beziehungen genauso (sprach- und implementations-)effizient wie Zu-1-Beziehungen navigierbar wären. An dieser Stelle erscheint also eine Änderung am OODM durchaus wünschenswert — eine Notwendigkeit der Erweiterung um Relationen läßt sich daraus aber ebenfalls nicht ableiten.

Entsprechendes gilt für die Umsetzung bidirektionaler Beziehungen: Hier die Verantwortung für die Koordinierung paariger Attribute (für die Repräsentation beider Richtungen) in die Hand derjenigen zu geben, die für eine korrekte Umsetzung eines Modells zu sorgen haben, ist der Häufigkeit dieser Aufgabe und ihrer immer wieder gleichen Form nicht angemessen. Statt dessen Relationen einzuführen, denen die Bidirektionalität immanent ist, hätte aber den Preis, auf die einfache und effiziente Dereferenzierung von Zeigern (s. o.) zu verzichten. Alternativ wäre vielmehr zu bedenken, ob nicht die koordinierte Pflege paariger Attribute Bestandteil des OODM werden sollte.

3.5 Notwendige Erweiterung aus Modellierungssicht

Im wesentlichen können wir aus der Betrachtung der relationalen Elemente im UML-Klassendiagramm zwei wünschenswerte Erweiterungen des OODMs ableiten:

1. eine Integration von Zu-*n*-Beziehungen, die den Umweg über explizite Zwischenobjekte (Collections) überflüssig macht und gleichzeitig einen qualifizierten (mit beliebigen Werten indizierten) Zugriff auf die Elemente der Gegenseite erlaubt, und
2. die Einführung von paarigen Attributen.

Zur Umsetzung wären dann auch die Update-Operationen des OODMs zu erweitern bzw. anzupassen: Zur Zuweisung kommen Einfüge- und Entfernoperationen für die Attribute, die Zu-*n*-Beziehungen repräsentieren, hinzu und die Pflege (Zuweisung bzw. Einfügen/ Entfernen) von paarigen Attributen muß so angepaßt werden, daß zu jeder Richtung automatisch auch die Gegenrichtung gepflegt wird.

4 Notwendige Ergänzungen der Objektorientierung aus Datenabfragesicht

Ein weiterer Grund für eine relationale Datenmodellierung ist die Existenz von ausgereiften Datenabfragesprachen. So erlaubt es beispielsweise die weit verbreitete SQL [ISO08], Abfragen höchst unterschiedlicher Komplexitätsgrade gegen relationale Datenbestände deklarativ zu formulieren. Die Prägnanz solcher Abfragen steht im Gegensatz zur Ausführlichkeit imperativer Formulierungen, wie man sie als objektorientiert programmierte Abfragen gleichen Inhalts antrifft.[8] Die Ursache für diese (vermeintliche) Geschwätzigkeit liegt u. a. darin, daß objektorientierte Umsetzungen von Abfragen stets iterativ arbeiten, also im Gegensatz zur Mengenwertigkeit des Relationalen jedes Element (Objekt) einzeln behandeln müssen, und zwar selbst dann, wenn es nur darum geht, über ein Element zum nächsten zu navigieren (das Element selbst also gar nicht

[8] Daß dies längst nicht für alle möglichen Abfragen gilt und daß im übrigen die Formulierung korrekter SQL-Abfragen längst nicht immer einfach ist, soll hier nicht weiter diskutiert werden.

betrachtet wird). Dem kann man entgegenhalten, daß die Mengenbehandlung relationaler Abfragesprachen auch Nachteile hat, z. B. wenn das Ergebnis einer Anfrage genau ein Objekt sein soll oder wenn man mit jedem Objekt einer Ergebnismenge hinterher etwas anderes machen will. Nichtsdestotrotz gibt es Bestrebungen, relationale Anfragesprachen in die objektorientierte Programmierung zu integrieren, und nicht zuletzt wurde mit Microsofts LINQ-Projekt (genauer: mit LINQ to Objects) die Möglichkeit eröffnet, auch Collections (also programminterne, objektorientierte Datenstrukturen) mengenorientiert auszuwerten [To07]. Dies genauer zu betrachten hat sich als ausgesprochen instruktiv erwiesen.

4.1 Auswahlabfragen

Relationale Abfragesprachen wie SQL oder LINQ liefern grundsätzlich eine Ergebnismenge, das sog. Result set. Selbst wenn das Ergebnis der Abfrage genau ein Objekt ist, bleibt es eine Menge, die selbst wieder Gegenstand einer Abfrage sein kann. Ein Vorteil der Mengenbetrachtung ist, daß „kein Ergebnis“ durch eine leere Menge repräsentiert wird und somit in der Regel keiner Sonderbehandlung (wie einer Prüfung auf `not null`) bedarf. Ein weiterer Vorteil ist, daß das Ergebnis als Ganzes mit einem Schritt präsentiert werden kann, z. B. indem es eine neue Datenbanktabelle füllt oder in einer Tabelle auf dem Bildschirm angezeigt wird.

Letzteres ist aber längst nicht immer das Ziel und so entpuppt sich der vermeintliche Vorteil nicht selten als Nachteil: Auf eingebettete relationale Datenbankabfragen folgt häufig prompt eine Iterationen über die Ergebnismenge, bei der jedes Element des Ergebnisses einzeln betrachtet und behandelt wird. So haben praktisch alle Beispiele für LINQ-Abfragen, die keine Aggregatfunktionen oder Joins verwenden, die folgende stereotype Form:

```
var query = from <Laufvariable> in <Collection>
    where <Auswahlkriterium auf Laufvariable>
    select <Ausdruck auf Laufvariable>;
foreach (<Laufvariable> in query) {…}
```

Auf die Mengenwertigkeit des Ergebnisses kann also ohne Verlust verzichtet werden, wenn die Elemente des Ergebnisses mit gleichem Aufwand der Reihe nach innerhalb der For-each-Schleife erzeugt werden können. Genau dies ist aber häufig der Fall, wie die Gegenüberstellung obigen Beispiels mit der objektorientierten Standardformulierung

```
foreach (var <Laufvariable> in <Collection>)
  if (<Auswahlkriterium auf Laufvariable>) {…}
```

zeigt.

4.2 Aggregatfunktionen

Ein weiterer Vorteil mengenwertiger Abfragesprachen ergibt sich aus der Verfügbarkeit von Aggregatfunktionen wie der Bildung der Summe oder der Berechnung des Durchschnitts einer Menge von Werten. Entsprechende Abfragen in LINQ haben in etwa die folgende Form:

```
var query = (from <Laufvariable> in <Collection>
    where <Auswahlkriterium auf Laufvariable>
    select <Ausdruck auf Laufvariable>)
    .<Aggregatfunktion>([<Ausdruck auf Laufvariable>]);
```

Tatsächlich spart man sich hier gegenüber der iterativen Form

```
<Akkumulator initialisieren>
foreach (<Laufvariable> in <Collection>)
  if (<Auswahlkriterium auf Laufvariable>) {<akkumulieren>}
```

das Führen der Akkumulatoren, die zur Berechnung des Aggregatwerts notwendig sind. Allerdings ist die Zahl der möglichen Aggregatbildungen fest vorgegeben und man muß schon die Bibliothek erweitern, wenn man eine neue benötigt. Im Vergleich zu den sog. Folds, wie sie beispielsweise mit der Collection-Methode `inject:into:` von Smalltalk zur Verfügung stehen (und wie es sie auch in LINQ gibt), mit denen man beliebige Aggregatfunktionen realisieren kann, erscheinen die Aggregatfunktionen relationaler Abfragesprachen geradezu altbacken starr (und liefern sicher keinen Anhaltspunkt, das OODM über das, was sowieso schon sinnvoll erscheint, hinaus zu erweitern).

4.2.1 Joins

Vielleicht der größte Vorteil relationaler Abfragesprachen ist die Möglichkeit der Bildung von Verknüpfungen von Relationen durch sog. Joins. Anhand des OODMs

```
class A { Collection<B> b; … }
class B { Collection<C> c; … }
class C { D d; … }
```

läßt sich in LINQ beispielsweise die Abfrage

```
foreach (D dl in (
    from tmp1 in a1.b from tmp2 in tmp1.c select tmp2.d)
  ) { … dl … }
```

formulieren, die die Menge der Attributwerte `d` von Objekten `c` vom Typ `C` gewinnt, die zu einem Objekt `a1` vom Typ `A` indirekt über Objekte `b` vom Typ `B` (über die Verknüpfung der Beziehung von `A` zu `B` und von `B` zu `C`) in Beziehung stehen. Man beachte, daß sich diese Verknüpfung objektorientiert nur im Fall von Zu-1-Beziehungen zwischen `A` und `B` sowie `B` und `C` durch die Verkettung von Attribut-Dereferenzierungen wie in

```
a1.b.c.d
```

herstellen läßt. (wobei dies auch noch ignoriert, daß `a1.b` oder `b.c` auch `null` sein könnten). Andernfalls sind geschachtelte Iterationen wie in

```
foreach (B b in a1.b)
  foreach (C c in b.c)
    … c.d …
```

notwendig, die doch deutlich imperativen Charakter haben.

Wie man an diesem Beispiel allerdings deutlich sieht, sind für Joins auf objektorientierten Datenstrukturen gar keine Relationen als eigenständige Konstrukte notwendig — das Vorhandensein eines Attributs, das eine Collection enthält (die Standardrepräsentation von Zu-*n*-Beziehungen im OODM) reicht vollkommen aus. Jede als einfaches Attribut repräsentierte Zu-1-Beziehung in einer Kette von Joins wurde allerdings insofern zu einer Anomalie führen, als dann die dazugehörige From-Klausel durch eine Attributdereferenzierung ersetzt werden müßte. Dies ist insbesondere dann schlecht, wenn sich die Kardinalität im Laufe der Entwicklung ändert.

Die Schlußfolgerung aus diesem Umstand darf aber nicht sein, daß man Relationen (als eigenständiges Konstrukt) zur Vereinheitlichung von Zu-1- und Zu-n-Beziehungen einführen muß, sondern vielmehr, daß man Zu-1- und Zu-*n*-Beziehungen als Attribute grundsätzlich gleich behandeln sollte (was sich ja andeutungsweise oben schon als Forderung ergab, wenn auch durch die durch Zwischenobjekte verursachte Anomalie motiviert).

4.2.2 Navigationsrichtungen von Abfragen

Mit jeder Abfrage ist für jede Beziehung, die sie ausnutzt, eine Navigationsrichtung fest vorgegeben. Eine Forderung nach bidirektionalen Beziehungen ergibt sich von Abfrageseite nur dann, wenn verschiedene Abfragen verschiedene Richtungen derselben Beziehung benötigen.

4.3 Notwendige Erweiterung aus Datenabfragesicht

Während sich aus Modellierungssicht vielleicht noch umfangreichere Anforderungen an Erweiterungen des OODMs ableiten lassen, bleibt aus Abfragesicht im wesentlichen übrig, Zu-1- und Zu-*n*-Beziehungen gleichartig zu repräsentieren. Nimmt man die aus der Navigation motivierte Forderung ernst, bedeutet es in der Konsequenz, daß für beide Arten von Attributen, denen, die Zu-1-Beziehungen umsetzen und denen, die Zu-*n*-Beziehungen umsetzen, auch dieselben Update-Operationen (Zuweisung, Einfügen, Entfernen) zur Verfügung stehen sollten, so daß der Unterschied nur noch in ihrer Deklaration sichtbar wird. Bidirektionalität als Forderung ergibt sich nur dann, wenn sie auch aus Modellierungssicht notwendig ist (sofern man in Anfragen nur Zusammenhänge

ausnutzen können soll, die auch bei der Modellierung der auszuwertenden Daten vorgesehen waren).

5 Eine Sanfte „Relationalisierung“ der Objektorientierung

Unsere Betrachtungen münden im wesentlichen in der Forderung, die Attribute des OODMs in zweierlei Hinsicht zu erweitern (vgl. [SS10] für eine genauere Erläuterung und Beschreibung einer möglichen Umsetzung in Form einer Programmbibliothek):

1. Attribute, die Relationen repräsentieren, enthalten grundsätzlich *eine Menge* von Zeigern, die über einen Schlüssel indiziert (qualifiziert) sein kann (mit nur einem Zeiger als Spezialfall). Die Operation „Zuweisung“ wird so umdefiniert, daß stets eine Menge von Zeigern zugewiesen wird, und durch Operationen zum Hinzufügen und Entfernen von Zeigern ergänzt. Die Dereferenzierung von Attributen, die Relationen repräsentieren, erfolgt ausschließlich in Abfragen (analog zu denen LINQs).
2. Attribute, die bidirektionale Relationen repräsentieren, werden immer als Paare geführt. Die Paare werden dazu in einer separaten Deklaration, einer Relationsdeklaration, angegeben. Genau wie die Generalisierung und andere Beziehungen höherer Ebene (vgl. Abschnitt 3.2) wird die Relationsdeklaration nicht zur Laufzeit instanziiert; die Tupelmengen, die die Extension einer Relation darstellen, werden durch die Inhalte der entsprechenden Attribute der beteiligten Objekte repräsentiert.

Die auf relationale Aspekte zurückzuführenden Unterschiede zwischen dem UML-Klassendiagramm und dem OODM reduzieren sich damit im wesentlichen auf das Fehlen von höher als zweistelligen Relationen und von attributierten Relationen im OODM, die jedoch vergleichsweise selten sind und durch Klassen ersetzt werden können.

6 Zusammenfassung und Schluß

Das Fehlen von Relationen im OODM ist immer wieder bemängelt worden. Anstatt wie in der modellgetriebenen Softwareentwicklung die Relationen eines Modells in immer gleichen Code zu transformieren, der die Bedeutung der Relationen in den Primitiven objektorientierter Programme ausdrückt, schlagen wir vor, mit dem OODM der relationalen Modellierung entgegenzukommen, indem wir es an einigen Stellen behutsam erweitern. Die aus unserer Sicht sinnvollen Erweiterungen des OODMs haben wir aus der Betrachtung des UML-Klassendiagramms und seinem systematischen Vergleich mit dem OODM sowie einer Untersuchung der Ausdrücke einer relationalen Abfragesprache abgeleitet; die vorgeschlagene Erweiterung haben wir in einer parallelen Arbeit ([SS10]) konkretisiert.

Literaturverzeichnis

[ABS04] Amelunxen, C.; Bichler, L.; Schürr, A.: Codegenerierung für Assoziationen in MOF 2.0.Modellierung 2004. LNCS, vol. P-45. Springer-Verlag, 2004, S. 149-168.

[AHM07] Akehurst, D.; Howells,G.; McDonald-Maier, K.: Implementing associations: Uml 2.0 to java 5. Software and Systems Modeling, vol. 6(1), Springer, 2007, S. 3-35.

[BO97] Bock, C.; Odell, J.J.: A More Complete Model of Relations and Their Implementation – Part II: Mappings. In Journal Of Object-Oriented Programming, vol 10(6), 1997.

[BW05] Bierman, G.; Wren, A.: First-Class Relationships in an Object-Oriented Language. In Proceedings of ECOOP 2005. LNCS, vol. 3586. Springer, 2005, S. 25-29.

[Ch76] Chen, P.P.: The entity-relationship model – toward a unified view of data. In ACM Trans. Database Syst. 1, 1 (Mar. 1976), 1976, S. 9-36.

[Co70] Codd, E.F.: A relational model of data for large shared data banks. Commun. ACM 13, 6, 1970, S. 377-387.

[Ge09] Gessenharter, D: Implementing UML associations in Java: a slim code pattern for a complex modeling concept. RAOOL '09. ACM Press., 2009, S. 17-24.

[GM96] Gawecki, A; Matthes, F.: Integrating Subtyping, Matching and Type Quantification: A Practical Perspective. 10th ECOOP. LNCS, vol. 1098. Springer, 1996, 26-47.

[GR83] Golberg, A; Robson, D.: Smalltalk-80: The Language and its Implementation. Addison Wesley, Reading, Massachusetts, 1983.

[GRL03] Génova, G.; Llorens, J.; Ruiz del Castillo, C.: Mapping UML Associations into Java Code. In Journal of Object Technology, vol. 2(5), 2003, S. 135-162.

[ISO08] ISO/IEC 9075-2:2008. Information technology – Database languages – SQL – Part 2: Foundation (SQL/Foundation), 2008.

[MM95] Stonebraker, M.; Moore, D.: Object Relational Dbmss: the Next Great Wave. Morgan Kaufmann Publishers Inc., 1995.

[MS92] Matthes, F.; Schmidt, J.W.: Definition of the Tycoon Language TL - a preliminary report. Informatik Fachbericht FBI-HH-B-160/92, Universität Hamburg, 1992.

[MS93] Matthes, F.; Schmidt, J.W.: System Construction in the Tycoon Environment: Architectures, Interfaces and Gateways. Euro-Arch'93 Congress. Springer, 1993, S. 301-317.

[No97] Noble, J.: Basic relationship patterns. In Proceedings of EuroPLOP, 1997.

[NPN08] Nelson, S.; Noble, J.; Pearce, D.J.: Implementing first-class relationships in java. In Proceedings of RAOOL 2008. ACM Press, 2008.

[OMG09a] Object Management Group (OMG): OMG Unified Modeling LanguageTM (OMG UML), Superstructure, Version 2.2 (OMG Document formal/2009-02-02), 2009.

[OMG09b] Object Management Group (OMG): OMG Unified Modeling LanguageTM (OMG UML), Infrastructure, Version 2.2 (OMG Document formal/2009-02-04), 2009.

[Øs07] Østerbye, K.: Design of a class library for association relationships. In Proceedings of LCSD '07. ACM Press, 2007, S. 67-75.

[Ru87] Rumbaugh, J.: Relations as semantic constructs in an object-oriented language. In Proceedings of OOPSLA '87. ACM Press, 1987, S. 466-481.

[Sc77] Schmidt, J.W.: Some high level language constructs for data of type relation. In ACM Transactions on Database Systems, vol.2, No.3, 1977, S. 247-261.

[SM92] Schmidt, J.W.; Matthes, F.: The database programming language DBPL – Rationale and Report. Technical Report FIDE/92/46, FB Informatik, Universität Hamburg, 1992.

[SS10] Stadtler, D.; Steimann, F.: Objektrelationale Programmierung. In SE 2010 (im Druck).

[To07] Torgersen, M.: Querying in C#: how language integrated query (LINQ) works. In OOPSLA '07. ACM Press, 2007, S. 852-853.

Zur Validierung von Kompositionsstrukturen in UML mit USE

Lars Hamann[1], Martin Gogolla[2] und Mirco Kuhlmann[3]

Abstract: In der Softwareentwicklung rücken Modelle zunehmend in den Fokus des Entwicklungsprozesses. Dadurch steigen auch die Anforderungen an deren Qualität. Mit dem an der Universität Bremen entwickelten UML/OCL-Werkzeug USE können bereits bestimmte Qualitätsaspekte von Modellen statisch und dynamisch analysiert werden. Dieser Artikel beschreibt neue Modellierungselemente der UML 2 und zeigt, welchen Beitrag eine Weiterentwicklung von USE auf dem Weg zu einer integrierten Semantik der UML 2 Kompositionsstrukturen leisten kann.

1 Einleitung

In der Softwareentwicklung verlagert sich der Fokus zunehmend von der code-zentrierten Entwicklung, bei der Modelle (wenn überhaupt) nur für den Entwurf oder die Dokumentation benutzt werden, hin zur modell-zentrierten Entwicklung, bei der formale Modelle, aus denen auch Quellcode generiert werden kann, im Mittelpunkt stehen; aktuelle Schlagwörter dafür sind *Model Driven Development* bzw. *Model Driven Architecture* (MDD/MDA). Der Begriff „formales Modell“ bedeutet, dass das Modell einen Aspekt der zu erstellenden Software vollständig beschreibt [SVEH07, S. 11]. Ein Modell kann auf verschiedene Arten durch textuelle oder grafische Modellierungssprachen definiert werden, wobei im Kontext der Softwareentwicklung die *Unified Modeling Language* (UML) [OMG09b] weit verbreitet ist. Durch die zentrale Stellung von Modellen erhält auch deren Qualität eine größere Bedeutung, da diese essentiell für die weitere Verarbeitung (Codegenerierung oder Interpretation des Modells) ist. Die Qualität der (statischen) Softwarestruktur hat einen großen Einfluss auf die Gesamtqualität des resultierenden Systems, da diese (in der Regel) den Grundstein für dynamische Vorgänge innerhalb des Systems bildet.

Die Möglichkeiten der Strukturbeschreibung wurden in UML 2 auch im Hinblick auf die modellgetriebene Entwicklung gegenüber der UML 1 erweitert. In diesem Artikel liegt der Fokus auf den Erweiterungen im Bereich der Darstellung von Kompositions-

[1] Universität Bremen, Fachbereich Informatik, AG Datenbanksysteme, D-28334 Bremen, lhamann@informatik.uni-bremen.de
[2] Universität Bremen, Fachbereich Informatik, AG Datenbanksysteme, D-28334 Bremen, gogolla@informatik.uni-bremen.de
[3] Universität Bremen, Fachbereich Informatik, AG Datenbanksysteme, D-28334 Bremen, mk@informatik.uni-bremen.de

strukturen und deren Auswirkungen auf das Klassenmodell. Während UML 1 für deren Darstellung nur Komposition und Aggregation zwischen Klassen zur Verfügung stellte, können diese in UML 2 auch mit Hilfe von sogenannten *Composite Structures* [OL06, HMPW04] dargestellt werden. Die bisher vorhandenen Modellierungsmöglichkeiten zeigten vor allem bei der Verbindung von Elementen auf der gleichen Dekompositionsebene Schwächen [Boc04]. Der neue Ansatz erlaubt eine detaillierte Beschreibung von Kompositionen in einem UML Modell. Dabei können komplexe Kompositionsstrukturen innerhalb einer einzelnen Klasse dargestellt werden, was gegenüber der klassischen „Teil-Ganzes-Beziehung“ präzisere Spezifikationen von Modellen erlaubt. Damit wird es möglich, über die UML 2 Modelleinschränkungen auszudrücken, die vorher als zusätzliche Einschränkungen z. B. in OCL [WK03, OMG06] angegeben werden mussten. Im Folgenden wird ein kurzer Überblick über die neuen Modellierungselemente gegeben; eine detailliertere Einführung findet sich in [Boc04]. Das in Abbildung 1 dargestellte Beispiel (mit geringfügigen Änderungen übernommen aus [Boc04]) zeigt Kompositionen, wie sie in UML 1 möglich waren. Dadurch, dass die Assoziationen global für Klassen definiert sind, sich also auf alle Instanzen der Klassen beziehen, können sich in Abhängigkeit von Instanzeigenschaften verschiedene Probleme ergeben:

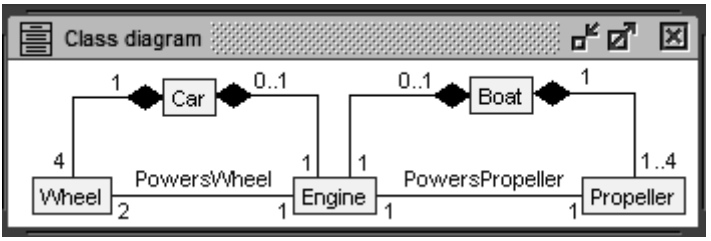


Abb. 1: UML 1 Kompositionen

- Ein Motor (Engine) innerhalb eines Auto-Objekts könnte die Räder eines anderen Auto-Objekts antreiben oder sogar die Schiffsschraube eines Bootes.
- Ein Motor muss Räder *und* Propeller antreiben.

Eine gültige Modellinstanz, die sicherlich so nicht gewünscht aber erlaubt ist, ist in Abbildung 2 gezeigt. Dabei treibt der Motor *e1* sowohl zwei Räder (*w1* und *w2*) des Autos *car* als auch die Schiffsschraube *p1* des Bootes *boat* an, obwohl der Motor *e1* zum Objekt *car* gehört. Entsprechende Unstimmigkeiten finden sich beim Motor *e2*. Weiterhin werden alle vier Räder von *car* angetrieben, obwohl nur zwei Räder mit dem zum Auto gehörenden Motor verbunden sind.

Durch Anpassung der Multiplizitäten (u. a. auf 0..1 statt 1 bei den Assoziationen `PowersWheel` und `PowersPropeller` an den Assoziationsenden des Motors) und zusätzliche OCL-Constraints kann dieses Modell entsprechend korrigiert werden. Ein mögliches OCL-Constraint, welches definiert, dass ein Motor innerhalb eines Autos zwei Räder antreibt und keine Schiffsschrauben, wäre:

```
context Engine inv inCarRequiresTwoPoweredWheels:
```

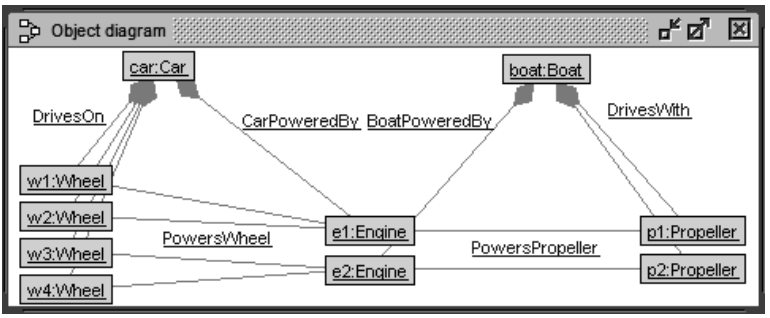


Abb. 2: Objektdiagramm

```
self.car.isDefined() implies
  self.wheel->size() = 2 and self.propeller->size() = 0
```

Dieses Problem kann auch durch weitere Spezialisierungen der Klasse `Engine` in `CarEngine` und `BoatEngine` und weiteren Assoziationen gelöst werden.

Das zweite angesprochene Problem, dass ein Motor Antriebsteile aus einem anderen Fahrzeug des selben Typs antreiben könnte, kann jedoch nur mit Constraints gelöst werden und nicht durch eine Veränderung der Vererbungsbeziehungen:

```
context Engine inv powersSameCar:
 self.car.isDefined() implies
  self.wheel->forAll(w:Wheel | w.car=self.car)

context Engine inv powersSameBoat:
 self.boat.isDefined() implies
  self.propeller->forAll(p:Propeller | p.boat=self.boat)
```

Ein Nachteil der angegebenen Constraints und der Spezialisierung in Auto- und Bootmotoren im Hinblick auf die Wiederverwendung und auf eine komponentenbasierte Entwicklung ist, dass der Entwickler eines Motors bereits zur Entwicklungszeit wissen muss, in welchen Kontexten ein Motor eingesetzt wird. Im Falle der Spezialisierung stellt sich zusätzlich die Frage, ob es tatsächlich eine Unterscheidung der Motorenarten geben soll oder ob dies nur eine technisch motivierte Vererbungsbeziehung wäre, was im Allgemeinen vermieden werden sollte.

Mit dem Kompositionsmodell der UML 2 kann die gewünschte Struktur ohne zusätzliche Constraints, also nur mit Hilfe von grafischen Mitteln, ausgedrückt werden. Die in Abbildung 3 dargestellten Structured Classifier stellen dies dar, indem sie den Blickwinkel auf individuelle Instanzen von Klassen (in diesem Fall Autos und Boote) legen. Für ein Auto bedeutet dies z. B., dass es genau einen Motor, zwei Vorder- und zwei Hinterräder besitzt und der Motor die beiden Vorderräder des Autos antreibt. Weiterhin wird laut [Boc04] sichergestellt, dass ein Motor in einem Auto nichts anderes in diesem oder einem ande-

ren Auto oder einem Boot antreibt. Die Unterteilung in Vorder- und Hinterräder, sowie der alleinige Antrieb der Vorderräder sind in dieser Modellierung hinzugekommen, da sie sich hier adäquat darstellen lassen. In der vorangegangenen Modellierung wäre dazu eine Unterteilung der Komposition zwischen `Car` und `Wheel` in zwei Kompositionen mit den Multiplizitäten 0..1 auf Seiten von `Car` nötig. Auch hier müsste über ein zusätzliches Constraint die Anforderung realisiert werden, dass ein Rad immer zu einem Auto gehört. Die

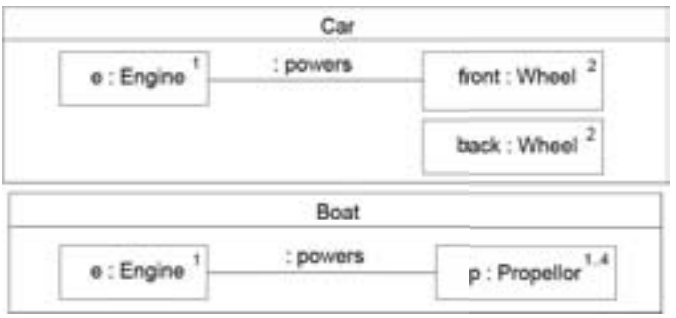


Abb. 3: UML 2 Kompositionen

eingebetteten Rechtecke innerhalb von `Car` und `Boat` stellen keine Klassen dar, sondern *Parts*. Die Verbindung zwischen Parts ist ein *Connector*. Im Gegensatz zu Assoziationen verbinden Konnektoren nicht zwei beliebige Instanzen der verbundenen Klassen, sondern Instanzen, die im jeweiligen Kontext als der angegebene Part auftreten (siehe [OMG09b, S. 174]).

Für die modellgetriebene Softwareentwicklung sind zusätzlich die neu hinzugekommenen *Ports* von Bedeutung, da sie eine komponentenbasierte Architektur unterstützen. Sie ermöglichen es, Komponenten zu orchestrieren, sodass eine lose Kopplung erreicht werden kann. Ports stellen Interaktionspunkte eines Classifiers dar, über die der jeweilige Classifier mit seiner Umgebung oder seinen internen Parts interagieren kann. Ein Port kann dafür bereitgestellte und benötigte Dienste spezifizieren (*Required/Provided Interfaces*). Eine detaillierte Beschreibung von Ports sprengt den Rahmen dieses Artikels und daher verweisen wir auf [HMPW04].

2 Wechselwirkungen zwischen Kompositions- und Klassendiagramm

Dass das Kompositionsdiagramm und das Klassendiagramm nicht unabhängig voneinander sind, liegt unter anderem an der Verwendung von Assoziationen als Typen von Konnektoren. Die in Abbildung 3 gezeigten Konnektoren verwenden als Typ die Assoziation `powers`, was durch den vorangestellten Doppelpunkt ausgedrückt wird; vor dem Doppelpunkt kann ein Name für den Konnektor stehen. Dies impliziert die Anwesenheit einer Assoziation zwischen `Engine` und `Wheel` sowie zwischen `Engine` und `Propeller` mit dem Namen `powers`. Damit unabhängig vom Kontext eines Motors von diesem zu seinen angetriebenen Teilen navigiert werden kann, um z. B. Zusicherungen für die an einen Motor angeschlossenen Teile definieren zu können, bietet es sich an, für die Klassen `Wheel` und `Propeller` eine gemeinsame Oberklasse zu definieren, z. B. `PowerTransmitter`.

Die Assoziation `powers` kann dann zwischen `PowerTransmitter` und `Engine` definiert werden, wie es in Abbildung 4 anhand eines Screenshots aus der nächsten USE Version dargestellt ist. Dabei dürfen sich die Multiplizitäten der Assoziation und die Multiplizitäten der Konnektoren nicht widersprechen. Die Konnektormultiplizitäten müssen entweder den gleichen oder einen kleineren Zustandsraum definieren [OL06]. Für Motoren können dann Einschränkungen definiert werden, sodass z. B. nur für die Leistung des Motors geeignete Teile angeschlossen werden dürfen:

```
context Engine inv noOverkill:
  self.transmitter->forAll(pt:PowerTransmitter |
    pt.allowedHorsePower >= self.horsePower)
```

Für komplexere Modelle müssen im Klassendiagramm zusätzliche Constraints an Assoziationsenden angegeben werden. Dies sind vor allem die in UML 2 hinzugekommenen Constraints der Form {*subsets* associationEnd}, {*redefines* associationEnd} und {*union*} (siehe [AS07], [Ame09]). Alle drei Constraints schränken die Verwendung der Assoziationsenden ein. Während *subsets* und *redefines* die Semantik eines spezialisierten Assoziationsendes verändern, verändert *union* die Semantik an einem allgemeineren Assoziationsende.

Das in Abbildung 4 dargestellte Klassendiagramm verwendet diese Möglichkeiten, um die in Abbildung 3 dargestellte Kompositionsstruktur auf Klassenebene abzubilden. Die

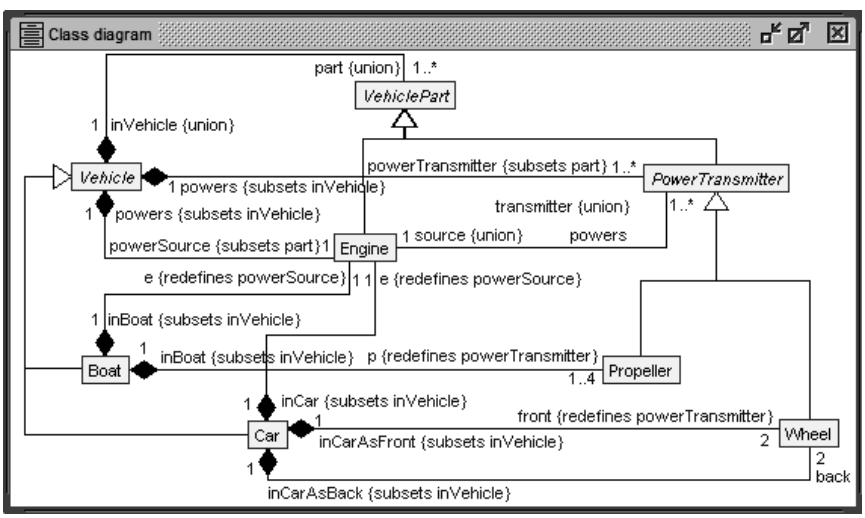


Abb. 4: Erweitertes Klassendiagramm

Komposition zwischen `Vehicle` und `VehiclePart` beinhaltet alle Teile eines Gefährts. Da durch spätere Spezialisierungen verschiedene Arten von Teilen hinzukommen können, sind die Assoziationsenden als {*union*} markiert. Das bedeutet, dass das Navigationsergebnis von `part` aus dem Kontext einer `Vehicle`-Instanz die Vereinigung aller Teilmengen ist, die sich durch die Navigation über Enden mit der Markierung {*subsets part*}

ergeben. Die andere Navigationsrichtung von `VehiclePart` zu `Vehicle` kann nur ein einzelnes Objekt ergeben, da das Assoziationsende eine Multiplizität von 1 hat. Für die Dokumentation macht es aber durchaus Sinn {*union*} anzugeben. Die UML Spezifikation [OMG09b] ist an dieser Stelle sehr frei interpretierbar. Während in der Spezifikation explizit erwähnt wird, dass sich die beschriebenen Constraints ausschließlich auf Assoziationsenden anwenden lassen und nicht auf Assoziationen (siehe [OMG09b, S. 40]), ergeben sich trotzdem Anforderungen an die zu den Enden gehörende Assoziation und an die gegenüberliegenden Assoziationsenden. So müssen alle Typen an den Enden einer Assoziation zueinander passend sein [OMG09a, S. 113]. Auffällig ist hierbei auch, dass die Typsicherheit nur in der UML Infrastructure gefordert wird, obwohl die jeweiligen Abschnitte über subsetting etc. in der Super- und Infrastructure nahezu identisch sind. Gar nicht in der UML Spezifikation angegeben sind Implikationen für die anderen Assoziationsenden. In [AP08] setzen sich die Autoren intensiv mit diesem Thema auseinander.

Der Unterschied von {*redefines*} und {*subsets*} kann mit der Komposition zwischen `Car` und `Wheel` für die Vorderräder veranschaulicht werden. Das Assoziationsende `front` ist mit {*redefines* powerTransmitter} und nicht mit {*subsets* powerTransmitter} markiert. Damit wird gefordert, dass alle Links zwischen einem Auto und einem kraftübertragenden Teil Räder sein müssen. Bei einer Markierung mit {*subsets*} würde stattdessen gefordert werden, dass die Menge der über die Assoziation verbundenen Räder eine Teilmenge der allgemeineren Assoziation sein müssen. Es wären also weiterhin Verbindungen zwischen Autos und anderen kraftübertragenden Teilen möglich (Schiffsschrauben fallen hier heraus, da diese an einer anderen Komposition teilnehmen müssen und eine Instanz nur an einer Komposition teilnehmen darf).

Der Vorteil der mit {*union*} gekennzeichneten Assoziationsenden gegenüber Enden ohne Markierung liegt in der möglichen automatischen Verarbeitung. Ein Modellinterpreter oder ein Codegenerator kann diese Markierung berücksichtigen und automatisch die Vereinigungsmenge berechnen bzw. Quellcode für die Berechnung erzeugen. Ohne diese Markierung muss die Berechnung manuell erfolgen und kann somit leichter zu Fehlern führen.

3 Werkzeugunterstützung

Schon das einfache Beispiel aus den vorangegangenen Abschnitten verdeutlicht, dass die Auswirkungen von Kompositionsstrukturen auf das Klassendiagramm nicht trivial sind und gut durchdacht werden müssen. Um bestimmte Sachverhalte zu validieren ist die Erzeugung von Szenarien der Modellnutzung hilfreich. Leider fehlt es hier, nicht nur im Zusammenhang mit den neuen Kompositionsstrukturen, an einer geeigneten Werkzeugunterstützung [OL06, BO06].

Das an der Universität Bremen entwickelte UML/OCL-Werkzeug USE [USE, GBR07] erlaubt es, auf Basis der UML und der OCL, Modelle zu spezifizieren und Instanziierungen (Objektdiagramme) dieser zu erzeugen. Dadurch unterstützt USE Entwickler bei der Validierung ihrer Modelle, indem die erzeugten Zustände auf ihre Gültigkeit hin überprüft und

Anfragen an das Modell gestellt werden können. So erhält der Entwickler frühzeitig eine Rückmeldung über die Qualität seines Modells. Weiterhin können endliche Zustandsräume über einen Generator geprüft werden [GBR03]. Dies ermöglicht es z. B. zu überprüfen, ob zu einem spezifizierten Modell ein gültiger Zustand existiert, also die Einschränkungen auf dem Modell nicht zu stark und nicht inkonsistent sind.

Bisher werden in USE die herkömmlichen Kompositionen und Aggregationen (*black* and *white diamonds*) unterstützt. Im Rahmen der Weiterentwicklung von USE ist die Integration der neuen Kompositionsstrukturen (vor allem Structured Classifier) geplant. Die Integration ist jedoch nicht das einzig verfolgte Ziel, vielmehr sollen durch die schrittweise Implementierung evtl. vorhandene Widersprüche in der UML Spezifikation aufgedeckt und behoben, sowie fehlende Angaben definiert werden. Dass die UML Spezifikation allein für die Umsetzung nicht ausreicht, zeigte sich bereits bei der Umsetzung der vorgestellten Constraints für Assoziationsenden. So ist z. B. unklar wie sich ein Validationswerkzeug verhalten muss, falls ein mit *subsets* markiertes Assoziationsende mit einem Assoziationsende ohne eine solche Markierung verbunden ist. Abbildung 5 zeigt eine solche Situation, bei der ein Werkzeug drei Möglichkeiten hat zu reagieren:

1. Einen Fehler melden und das Modell abweisen.
2. Das Assoziationsende `c` implizit mit *subsets* markieren und damit für die Berechnung der Vereinigung am Assoziationsende `a` verwenden.
3. Das Modell ohne Änderungen akzeptieren und das Assoziationsende nicht in die Berechnung der Vereinigung einbeziehen.

Das in Abbildung 5 auf der rechten Seite gezeigte Objektdiagramm ist nur im zweiten Fall gültig. Im dritten Fall entspricht die Menge der Objekte am Assoziationsende `a` nicht der Menge aller unterteilenden (subsetting) Enden. Da es keine Assoziationsenden gibt, die `a` unterteilen, ist die Vereinigungsmenge leer. Die Navigation `d1.a` ergibt aber `Set{c1}`, was ein Widerspruch ist. Im USE System wurde die dritte Variante realisiert, um die Transparenz der Modelleigenschaften zu wahren.

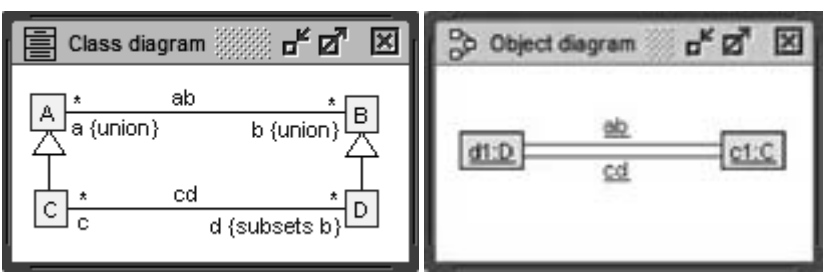


Abb. 5: Nicht angegebenes subsets Constraint

Das vorangegangene Beispiel verdeutlicht die Möglichkeiten der statischen Validierung innerhalb eines Werkzeugs. Eine der angesprochenen Stärken von USE ist zusätzlich die Validierung eines Modells zur Ausführungszeit. Dazu können Systemzustände erzeugt und deren Struktur gegen die Einschränkungen im entsprechenden Modell geprüft werden.

Eine Validierung zur Laufzeit kann Fehler aufdecken, die eine rein statische Analyse nicht findet. Ein Beispiel anhand des Klassendiagramms für Fahrzeuge verdeutlicht dies:

Fügt der Benutzer einen neuen Fahrzeugtyp hinzu, welcher eine andere Antriebsart verwendet, so muss auch eine neue redefinierende Assoziation für die Verbindung zwischen `Vehicle` und `PowerTransmitter` definiert werden, da eine einmal redefinierte Assoziation schreibgeschützt ist, um das Substitutionsprinzip zu gewährleisten [Boc04]. Wird keine redefinierende Assoziation modelliert und zur Ausführungszeit versucht ein Link zwischen dem neuen spezialisierten `Vehicle` und dessen speziellen `PowerTransmitter` zu erzeugen verhindert USE dieses. Dadurch erhält der Benutzer eine frühzeitige Rückmeldung, dass das von ihm erstellte Modell einen gewünschten Zustand nicht darstellen kann.

Durch die schrittweise Implementierung der Erweiterungen wird eine klar definierte Semantik der dazugehörigen UML Sprachelemente entwickelt, sodass mit USE ein Validierungswerkzeug mit einer integrierten Semantik von Structured Classifier und deren Elementen wie Ports, Parts, und Connectors verfügbar sein wird. Anhand von Szenarien können so z. B. Widersprüche in der Definition von Kompositionsstrukturen und Klassenmodellen aufgezeigt werden. Grundlegende Arbeiten dafür sind bereits umgesetzt, wie an den gezeigten Einschränkungen an Assoziationsenden zu sehen ist. In der nächsten USE Version wird es volle Unterstützung von *subsets*, *union* und *redefines* geben.

4 Fazit und Ausblick

Die in UML 2 hinzugekommenen Sprachelemente erweitern die Ausdrucksmöglichkeit der UML im Hinblick auf eine komponentenbasierte Architektur. Wie in diesem Artikel gezeigt, ist die UML Spezifikation allerdings nicht an allen Stellen eindeutig. Wir haben gezeigt, dass grundlegende Arbeiten zur Unterstützung von Structured Classifier im USE System umgesetzt worden sind und welche Interpretationsspielräume die UML Spezifikation einem Werkzeugentwickler lässt.

Mittelfristig soll in USE die Definition von Kompositionsstrukturen und deren Instanziierung möglich sein, so dass Inkonsistenzen zwischen diesen und dem Klassenmodell erkannt werden können. Interessant wird dabei auch die Frage sein, inwieweit bestimmte Angaben im Kompositionsdiagramm automatisch in das Klassenmodell übernommen werden können. Weiterhin ist die Integration von OCL in den Kontext der Kompositionsstrukturen geplant, bei der sich unter anderem die Frage stellt, ob z. B. Ansätze wie Komponenteninvarianten [HBKW01] übernommen werden können bzw. müssen. Die vollständige Integration des Konzepts der Kompositionsstrukturen mit der Unterstützung von Ports und Protokollautomaten (Protocol State Machines) ist das langfristige Ziel der USE-Entwicklung, sodass komponentenbasierte Modelle validiert werden können.

Literaturverzeichnis

[Ame09] C. Amelunxen. *Metamodel-based Design Rule Checking and Enforcement*. Dissertation, Technische Universität Darmstadt, 2009. Dissertation.

[AP08] Marcus Alanen und Ivan Porres. A metamodeling language supporting subset and union properties. *Software and Systems Modeling*, 7(1):103–124, feb 2008.

[AS07] C. Amelunxen und A. Schürr. Formalizing Model Transformation Rules for UML/MOF 2. *IET Software Journal*, 2(3):204–222, June 2007. Special Issue: Language Engineering.

[BO06] J. M. Bruel und I. Ober. Components Modeling in UML 2. *Studia Jurnal*, 1:79–90, 2006.

[Boc04] Conrad Bock. UML 2 Composition Model. *Journal of Object Technology*, 3(10):47–73, Between November and December 2004.

[GBR03] Martin Gogolla, Jörn Bohling und Mark Richters. Validation of UML and OCL Models by Automatic Snapshot Generation. In Grady Booch, Perdita Stevens und Jonathan Whittle, Hrsg., *Proc. 6th Int. Conf. Unified Modeling Language (UML'2003)*, Seiten 265–279. Springer, Berlin, LNCS 2863, 2003.

[GBR07] Martin Gogolla, Fabian Büttner und Mark Richters. USE: A UML-Based Specification Environment for Validating UML and OCL. *Science of Computer Programming*, 69:27–34, 2007.

[HBKW01] Rolf Hennicker, Hubert Baumeister, Alexander Knapp und Martin Wirsing. Specifying Component Invariants with OCL. In *GI Jahrestagung (1)*, Seiten 600–607, 2001.

[HMPW04] Øystein Haugen, Birger Møller-Pedersen und Thomas Weigert. *Structural Modeling with UML 2.0*, Kapitel 3, Seiten 53–76. Springer US, 2004.

[OL06] Ian Oliver und Vesa Luukala. On UML's Composite Structure Diagram. In *Fifth Workshop on System Analysis and Modelling*, Kaiserslautern, Germany, 2006.

[OMG06] *Object Constraint Language 2.0*. Object Management Group (OMG), Mai 2006. `http://www.omg.org/spec/OCL/2.0`.

[OMG09a] *UML Infrastructure 2.2*. Object Management Group (OMG), Februar 2009. `http://www.omg.org/spec/UML/2.2/Infrastructure/PDF/`.

[OMG09b] *UML Superstructure 2.2*. Object Management Group (OMG), Februar 2009. `http://www.omg.org/spec/UML/2.2/Superstructure/PDF`.

[SVEH07] Thomas Stahl, Markus Völter, Sven Efftinge und Arno Haase. *Modellgetriebene Softwareentwicklung: Techniken, Engineering, Management*. Dpunkt Verlag, 2007.

[USE] A UML-based Specification Environment. Internet. `http://sourceforge.net/projects/useocl/`.

[WK03] J. Warmer und A. Kleppe. *The Object Constraint Language: Precise Modeling with UML*. Addison-Wesley, 2003. 2nd Edition.

Modelling Interactions for Automatic Execution Using UML Activity Diagrams

Werner Putschögl[1] and Bernhard Dorninger[2]

Abstract: Software applications involving interactions of distributed systems are very common nowadays. Frequently, interactions are modelled during the analysis phase of a project and subsequently coded manually. Very often this results in a divergence of analysis model and the implemented code. Moreover, the border between interaction handling code and domain code may be blurred. In addition, hard-coding collaborations and interactions may impede maintainability of an application. In this paper, we propose a procedure for modelling interactions—and also collaborations—with the means of slightly extended UML activity diagrams. The resulting interaction model is then transformed to a machine interpretable format and may subsequently be processed and controlled by an interaction infrastructure, which we developed for this purpose. In addition, our procedure encourages a clear separation of interaction processing and domain code.

1 Introduction

Designing distributed, collaborative applications poses a demanding challenge. Major concerns include coordinating work among the participating nodes and the distribution of data needed and produced by the nodes. These concerns do apply for autonomous multi node systems as well as common client-server applications involving human-computer interaction. One means to cope with these challenges can be model-driven software development (MDSD), which nowadays is well established due to its various benefits [Se03]. Especially the Unified Modelling Language (UML) has emerged as the lingua franca for modelling the various aspects of software. Nearly every UML tool is capable of generating code at least from static models (class diagrams). There are also numerous tools that support the generation of code from behavioural models. However, UML [OMG09a] is often deemed insufficient for use in the context of MDSD [SV06]. Also, modelling interactions between autonomous systems and/or UI based applications with UML is not always considered adequate.

Frequently, interactions—especially in the field of business processes and workflows—are initially modelled via use cases and later detailed by activity diagrams (e.g. during requirements engineering). Use cases and activity diagrams have been criticized to lack formal semantics to generate code—although there are several proposals for enriching activities to solve this [SH05] [BS09]. A modelling method intended solely for business process modelling is Business Process Modelling Notation (BPMN) [GDW09]. Interactions can also be modelled with UML collaboration/communication diagrams [CBJ02], but are used to visualize object level collaborations rather than high level

[1] Software Competence Center Hagenberg, Softwarepark 21, 4232 Hagenberg, Werner.Putschoegl@scch.at
[2] Software Competence Center Hagenberg, Softwarepark 21, 4232 Hagenberg, Bernhard.Dorninger@scch.at

interactions. The Human Computer Interaction (HCI) community also has offered several suggestions to extend UML with mechanisms supporting the modelling of interactions [SP03] [PBL03]. These methods focus on the modelling of user interface related aspects of interactions and/or were developed for communicative purposes rather than for generating code or executable information.

Often, the analysed model of interactions is lost as the design and implementation process progresses, especially when interactions are coded manually. Modelled interactions are split and refined to various domain or infrastructure objects and functions in the codebase [CBJ02]. This results in a lack of traceability from the modelled interaction to the code. This problem is reinforced by the fact that implementations often mix up interaction related code with functional code and infrastructure related code. It may lead to highly complex method/function implementations in the collaborative peers, which in turn decreases maintainability as well as comprehensibility. In addition, it makes interactions and domain functions difficult to reuse in other scenarios. The principles of Domain Driven Design (DDD) [Ev04] may help here, which suggest a clear separation of domain code from infrastructure code.

In this paper we outline an approach of how to avoid the aforementioned downsides by proposing a procedure of interaction modelling with regard to a strict decoupling of interaction and domain functions and preserving the modelled information in the code.

2 Goals and Challenges

The primary goal of our work described in this paper is to provide a procedure for modelling and implementing interactions between software systems as well as a reusable software infrastructure for processing the modelled interactions.

It is desired that efforts for implementing and maintaining collaborations/interactions is kept low. Thus, implementation of interactions shall be automated as far as possible. Manual coding shall be reduced to the need of implementing the content of domain operations, whereas interaction related code or interaction descriptions shall be generated from predefined models. The procedure shall prescribe a clear separation of interaction and domain/business functions and preserve this separation in the resulting implementation. On the other hand, the key here is to provide traceable and comprehensible information concerning the flow of interaction and the relationship of actions to domain operations. Finally, modelling shall be based on a standardized technique. Possible enhancements shall be as simple as possible and shall not depend on a specialized modelling tool.

The interaction infrastructure shall not only suggest the compliance to an architectural pattern [CBJ02] or concentrate on the protocol or application level communication layer [ASQP05] but rather provide a reusable mechanism for processing any interaction or collaboration scenarios.

To satisfy these requirements, we have to cope with two key challenges.

- Distribution of control: Processing interactions in collaborative systems certainly involves the changeover of control. Each participant has its tasks to solve and actions to perform, which may depend on the actions and decisions of the fellow participants. We have to find a way to control the flow of interactions independent of a concrete interaction, i.e. the infrastructure shall be able to handle arbitrary interaction scenarios.
- Distribution of interaction data: Each action or decision within an interaction needs and/or produces data. This data may be needed by subsequent actions and/or decisions in the interaction flow. Since the interaction is distributed over potentially more than two nodes, we have to secure the availability of the needed data in the respective participant. Of course data distribution and availability shall be addressed already in the model and shall not lie in the responsibility of the programmer.

The remainder of this paper outlines the developed procedure and explains the necessary steps from analysis of the interaction to deployment of the solution.

3 Creating an Executable Interaction Model

In the software development process, the analysis phase usually results in a more or less detailed model of the respective domain, consisting at least of a data model and the usage scenarios of the planned software. Scenarios are documented in form of use cases, which may be represented by textual descriptions, use case diagrams and even behavioural diagrams like activity diagrams (AD). These use cases implicate necessary domain operations (aka business logic) as well as interactions between systems or users and a system.

In this section, we will outline a procedure, which builds on the results from the analysis phase by augmenting AD with additional information and subsequently generating an interpretable representation of interactions. For executing the interactions, an infrastructure was developed managing the distributed execution of an interaction at runtime. Figure 1 illustrates the fundamental steps of our procedure, structuring the procedure into three basic steps:

1. Modelling: The use cases from analysis phase have to be modelled as AD according to specifications we introduce in our procedure. At the same time, domain operations have to be modelled matching the actions from the AD. Next, data used by these operations during the interaction has to be specified. The procedure of modelling is described in detail in Section 3.1.
2. Transformation and Implementation: The goal of this step is the generation of an interpretable representation of the interaction model. Another task is the generation of code stubs for the modelled domain operations from step 1.

Subsequently, the functionality of the generated stubs has to be implemented. Section 3.2 will elaborate on this step.

3. Application: To verify the results of our efforts, we developed an interaction infrastructure, which is responsible for distributed invocation and execution of modelled interactions. This infrastructure ensures that each participating system processes the assigned actions, executes the respective domain operation and is supplied with data required from other participants. The concept of the infrastructure and its processing of interactions is explained in Section 3.3.

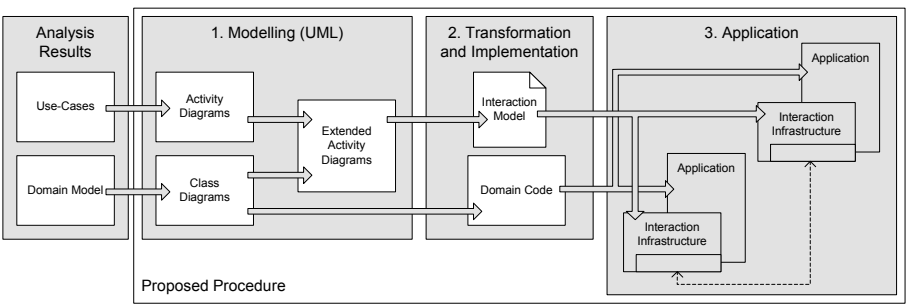


Fig. 1: Procedure Overview

The following sections describe each step in detail by using a simple example. We will demonstrate the way from use case descriptions to executable representations of the implicated interactions, and how interactions are executed and controlled by the infrastructure.

3.1 Modelling

The modelling step builds on the artefacts of the analysis phase and can be divided into three sub-steps, illustrated in Fig. 2. The figure depicts the artefacts required and generated in each sub-step.

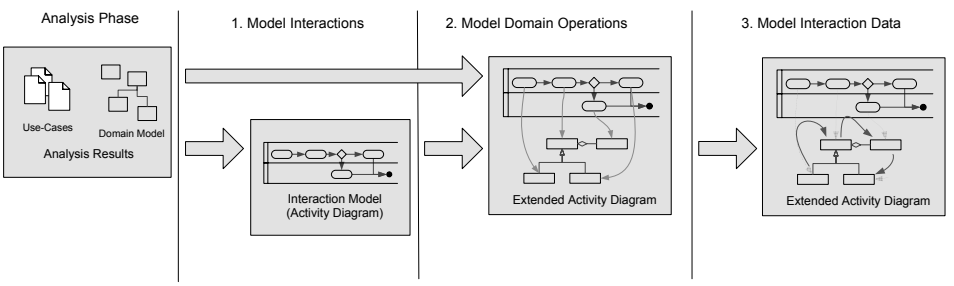


Fig. 2: Modelling Overview

The analysis phase provides us with use cases, which implicate domain operations and optional domain rules their execution is subjected to. Domain rules are conditions which have to be fulfilled before an operation can be executed. In addition we get a data model of the domain. These artefacts form the basis of our four modelling sub-steps.

1. Model Interactions: If not already created during analysis, the interactions described in use cases have to be transformed into activity diagrams following certain rules (see Section 3.1.1).
2. Model Domain Operations: Since we want to automatically execute and control the modelled interactions, our procedure prescribes the presence of domain operations for each modelled activity or domain rule from an activity diagram. Therefore, in this step, we explicitly model the domain operations in a static view, in our case an UML Class diagram. In doing so, we consider the recommendations of the Domain Driven Design approach [Evans]. Finally each activity and decision node is assigned to the appropriate domain operation. This connects the dynamic view of the activity diagram with the static view of the class diagram. The step "Model Domain Operations is described in detail in Section 3.1.2.
3. Model Interaction Data: Finally, we have to model the data, which is distributed between the individual operations. Furthermore, we have to concretize our domain rules in a machine-recognizable way. Section 3.1.3 describes this step in detail.

After these steps, the model is ready for transformation.

3.1.1 Model Interactions

We use a simple example of a flight booking interaction to demonstrate our proposed procedure. Listing 1 shows the use case gained from the analysis phase. Fig. 3 shows the activity diagram of the interaction based on the use case shown in Listing 1.

Flight booking:

Main Course:

1. The user chooses a flight to book.
2. The system asks the user for the number of passengers.
3. The user enters the number of passengers.
4. The system ensures that there are enough places vacant, reserves the places and books the flight.
5. The use-case ends successfully.

Alternative Courses
4.a.: The check fails as there are not enough seats available.
4.a.1. The system informs the user.
4.a.2. The user chooses to pick another flight

List. 1: Flight Booking Use Case

The example includes actions that are performed by a user on a client system, actions performed by the systems as well as a simple domain rule. This domain rule indicates that there must be sufficient seats available to continue with the reservation. In addition, it serves as precondition for the reserving and booking actions. The "Check Availability" decision (Fig. 3) evaluates this domain rule.

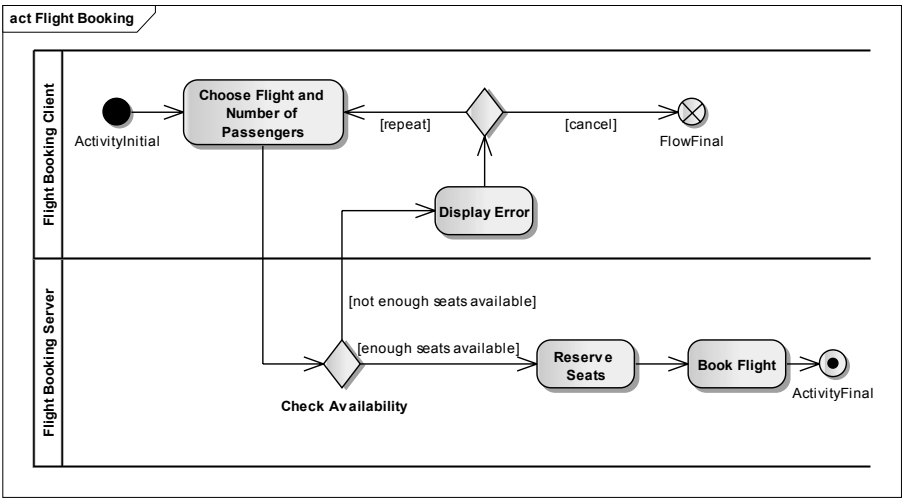


Fig. 3: A Basic Flight Booking Example

Distributed execution requires a mechanism which specifies where an operation is going to be processed. Our procedure introduces system roles to identify different systems in an interaction. A system role is a classifier such as "Flight Booking Client" or "Flight Booking Server", not a specific physical system, as for instance there can be multiple client systems connected to one server in the above example. In the AD we use partitions to model these system roles, each partition representing one role. Each diagram element has to be put in one partition. In the example shown in Fig. 3 each element is put in one of the two partitions specifying that the "Flight Booking" action is to be executed on a "Flight Booking Server"-type system.

Domain rules are represented by a decision node. To enable an automated evaluation of the decision each outgoing control flow has to be modelled using mutually exclusive conditions. At this stage of modelling, guards can only be expressed by abstract, textual conditions. These guard expressions will be refined into machine-recognizable expressions after specifying the interaction data (see Section 3.1.3).

3.1.2 Model Domain Operations

Having modelled the dynamic view with activity diagrams is not sufficient for our purpose. We also need to model the operations, which will be executed when processing an interaction. When modelling applications, it is common to have static views depicting

the domain's data model as well as operations on these data. Concerning operations, our procedure suggests applying the principles of Domain-Driven Design (DDD) [Ev04]. Evans introduces the so called "Service" pattern, which proposes the modelling of domain operations, which manipulate other domain objects (such as data or resources), as stateless services. We follow this view and expect the domain operations to be modelled according to this pattern. In addition, we model each operation as an interface, since our procedure requires the presence of an appropriate domain operation for each action in the activity diagram. To be able to process domain rules within an interaction we apply the "Specification" pattern proposed by Fowler and Evans [EF97]. It describes the modelling of domain rules as interfaces just like operations. The pattern allows integrating domain rules into our interaction model with their execution being similar to the execution of domain operations.

After having modelled the domain operations and rules we create an explicit relation between the dynamic view from the activity diagrams and the static view of the operations in the class diagrams. This is done by mapping each activity element and its respective domain operation (or the shortcut thereof).

Fig. 4 shows how the "Reserve Seats" action from the activity diagram is mapped to the "`ReserveSeatsOperation`" interface via an explicit link. The result of the link is that upon processing the interaction, the operation implemented in the interface is executed when the action is processed.

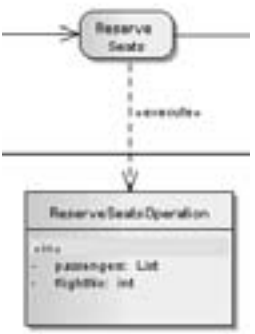


Fig. 4: Example Link

3.1.3 Model Interaction Data

The next task is to specify the data used in the context of the interaction. As already mentioned the distribution of data to systems involved in an interaction is a key challenge. We have to model data that will be exchanged by the operations and rules during the execution of interaction. The entirety of data exchanged between operations or between operations and rules within one interaction—which may span several system roles—is called *interaction context*. Data is specified as properties of the modelled

domain operations or rules. Data properties of operations must be stereotyped as input or output (or as both if this applies). The result of an evaluation of a domain rule is regarded an output data property. Once defined, each data property in the interaction context is available for all other operations and rules within the regarded interaction. The mapping of data properties between the operations and rules is achieved via lexical identity of the property names.

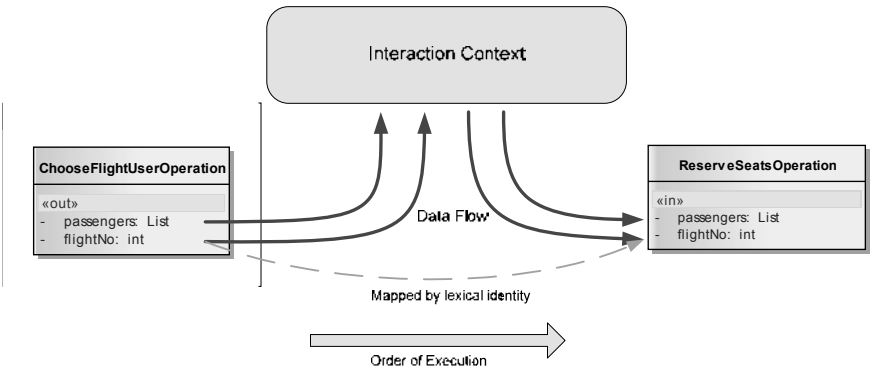


Fig. 5: Data Supply Example

In Fig. 5 the solid line arrows depict the data flow between two operations (via the interaction context) during the execution of an interaction. The action `ChooseFlightUserOperation` produces the data *flightNo*, which is required by the operation `ReserveSeatsOperation`. The mapping of domain operation data is achieved by lexical identity of the property names. This is indicated by the dashed line arrow. Any domain operation in the same interaction seeking to access *flightNo,* would have to define it as a `<<in>>` or `<<in,out>>` property.

Only data exchanged between operations and rules has to be specified here. Additional data required privately by a domain operation should not be part of the model.

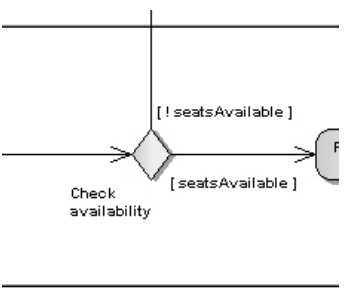


Fig. 6: Modelling Guard Conditions

After the interaction context has been defined, we now have to adapt the guard

conditions of our domain rules to match the appropriate data. In addition, we mandate the guards being expressed in a machine-interpretable way for allowing their automatic evaluation. Fig. 6 displays an example.

The Boolean data property "*seatsAvailable*" is the result generated by evaluating the preceding domain rule. Knowing the data property name, the guard condition must be transformed manually from a textual definition to a machine processable condition using the correct property name and values.

3.2 Transformation and Implementation

The interaction model and the domain operations created during modelling form the base for the step *Transformation and Implementation*. This step includes two sub-steps:

1. Model Transformation: The UML model is transformed into a XML representation of the interaction model used as input for the infrastructure.
2. Generation and Implementation: Based on the interfaces modelled to represent operations and rules, interfaces and class stubs are generated. The generated stubs have to be completed by manually coding the required functionality of the domain operations and domain rules.

This section describes the sub-steps to generate the artefacts required to be used by interaction infrastructure.

3.2.1 Model Transformation

For the model transformation, we define our own metamodel representing interactions. The next step is to export the AD modelling the interaction from the modelling tool and transform it into our metamodel. The XML Metadata Interchange (XMI) standard defined by the Object Management Group is widely supported by UML Modelling tools for exchanging and exporting model data. Since XMI is very verbose, we extract the relevant information concerning the interaction and transform it to an XML representation of our own metamodel (Fig. 7).

Our metamodel is loosely based on the UML metamodel [OMG09b], but of course takes a simplified view suitable for our needs. All elements of the metamodel representing actions or decisions implement the *InteractionElement* interface. This implies that all elements have an identifier unique for the interaction as well as a target role, defined by the partitions in the UML interaction model. The current metamodel supports two types of interaction elements: *ActionElements* and *DecisionElements*. *SpecificationElements* differ from *ActionElements* only in that *ActionElements* execute domain operations whereas *SpecificationElements* evaluate domain rules. The information required for both comprises the domain operation or rule to process including the data properties modelled and the succeeding element. For decisions associated with a domain rule a *SpecificationElement* is generated followed by a *DecisionElement.* The *DecisionElement*

contains the guard conditions determining the possible successors.

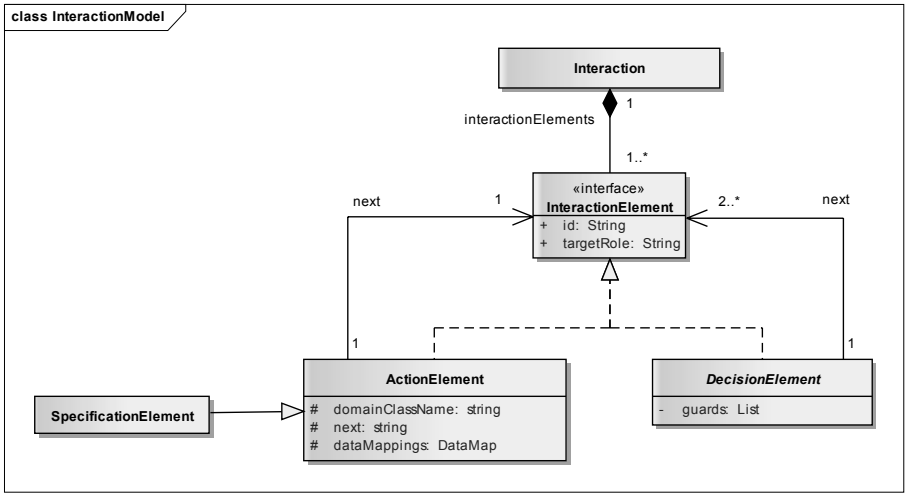


Fig. 7: The Interaction Metamodel

List. 1 is a short excerpt of the transformed XML file showing the "Reserve Seats" operation from the example.

```
…
<actionelement id='Reserve Seats' targetrole='Flight Booking Server'
operation='ReserveSeatsOperation' next='Book Flight'>
    <property name='flightNo' kind='in'>
    <property name='passengers' kind='in'>
</actionelement>
…
```

List. 1: Excerpt of generated XML.

3.2.2 Generation and Implementation

Based on the modelled domain operations and rules, interfaces and class stubs are generated. The domain operations and rules represented by the class stubs have to be implemented for the system they are intended for. This is done by manually adding the code of the functionality to the generated stubs. For data properties modelled in the interaction, get and set methods are generated automatically. The interaction infrastructure uses these to supply or retrieve data required by other operations or rules within the same interaction. Access to additional data necessary for an operation has to be coded as needed.

3.3 Application and Execution of Interactions

After having generated interpretable interaction definitions and the implementation of the relevant domain code (operations and rules), it is desired to execute these interactions. We developed an infrastructure to invoke and control the modelled interactions using Java. The infrastructure may be embedded into any Java based application, where it handles control flow, data supply and synchronization for the participating systems. Control flow management involves a mechanism keeping track of the progress of an invoked interaction and controlling which element has to be executed by which system. Execution encompasses calling the associated domain operation or evaluating a rule as well as supplying the required data.

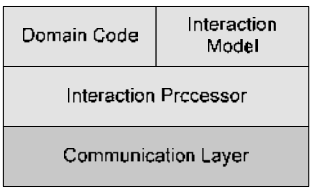


Fig. 8: Infrastructure Layers

The interaction infrastructure can be separated into two layers: the Interaction Processor and the Communication Layer (Fig. 8). The Domain Code and Interaction Model have to be created individually for each application.

The Interaction Processor is responsible for building an object representation from the generated interaction definition.

It furthermore has to handle the processing of the interactions, which includes data supply as well as the execution of domain operations or rules. The Interaction Processor also manages the execution focus, i.e. it has to take care that each domain operation or rule is executed by the system defined in the model. To achieve this, the processor uses the Communication Layer. The Communication Layer defines an interface that has to be implemented to connect the systems participating in an interaction. All technology dependent functionality is encapsulated in the Communication Layer. Our prototypical implementation relies on Java RMI, but other implementations using different technologies (such as CORBA) are feasible without changing the Interaction Processor.

Deployment and Configuration

To make use of the interaction infrastructure each of the systems requires deployment information in addition to the interaction model and the domain code. Deployment information has to be present for each participating system in an interaction. It includes the respective participant's system ID, its system role and information concerning the other potentially participating systems. The system roles are determined by the different partitions used in the interaction model. The system ID is an abstract, technology independent name for uniquely identifying a specific system within the set of

participants. Deployment information also has to contain a mapping from logical system IDs to physical system addresses. This information depends on the communication protocol used by the Communication Layer. In our prototypical implementation we map the logical system IDs to IP addresses.

Another issue is the distribution of deployment information. For our prototype the deployment information has been provided using one configuration file for each system participating in an interaction. This is error-prone and may prove inappropriate for more complex systems. Registry servers may be an alternative to local configuration files with the deployment information being automatically downloaded.

Interaction Invocation and Execution

Upon start-up of an application, the interaction infrastructure parses the deployment information and initialises the application's system role and ID. The infrastructure loads the interaction definitions from the XML file and constructs the corresponding interaction element graphs conforming to the metamodel from Fig. 7. The Communication Layer has to be configured with the deployment information about possible interaction participants. For all systems the system ID, system role and physical address has to be known to enable a connection.

An application invokes an interaction by passing the interaction's unique name to its embedded interaction processor. Subsequently, an interaction context for that specific interaction is created, which is valid for one invocation only.

The interaction context contains all information required for the execution of the interaction. The most important part of this information is the data exchanged by the domain operations and rules. When invoking an interaction, this data is extended by additional runtime information, which encompasses system role to ID mappings as well as the progress of the interaction.

Each role of an interaction has to be assigned to one specific system. Upon invocation of the interaction, the infrastructure assigns the role in the interaction to its system ID. This assignment does not change for the duration of the interaction invocation. After assigning the system role, the infrastructure starts to traverse the interaction object graph. If the current interaction element is to be executed locally, the interaction processor instantiates the according domain operation class and sets the data properties defined upon modelling. The data is read from the interaction context and passed to the domain operation or rule by invoking its setter methods. Then, the domain operation or rule is executed. After successful completion, the values of the operation or rule' output data properties are retrieved and written to the interaction context. Finally, the progress indicator in the interaction context is advanced to the successor of the just processed interaction element

If the current interaction element has to be executed by a system with another system role, the execution focus must be transferred to an appropriate participant. If the needed role has not been assigned to a system yet, the infrastructure tries to assign the role

automatically. Ambiguities (i.e. there are more participants potentially fulfilling a role) can be resolved by predefining the participating systems upon invocation or by routing algorithms making the choice. The interaction infrastructure provides extension mechanisms to support routing.

The execution focus is transferred by transmitting the interaction context containing interaction data, system role to ID mappings and the current progress indicator to the system that has been assigned the role the current interaction element was modelled in. The Communication Layer translates the system ID to an actual physical system and performs the context transfer. To reduce the network load, only new or changed data in the context is transferred. The interaction infrastructure of the target system is responsible for merging the changes into its interaction context. Once the other participant has the execution focus, it may continue with processing the interaction until it runs into an interaction element, which again needs to be processed by another participant. While the interaction is continued by other systems the local application waits for the interaction context and execution focus to return or the interaction to end.

This process is repeated until the interaction ends successfully or an error occurs. Upon termination, all participants are informed about the end of the interaction and if it was successful or not. If possible, information about the error is added to the context to provide feedback.

4 Discussion

The goal of our work described in this paper was to provide a modelling procedure combined with a reusable infrastructure to process and control interactions and/or collaborations between distributed systems. While there are a number of possible approaches to model interactions such as UMLi [SP03] or MoLIC [PBL03], we use activity diagrams, since UMLi and MoLIC both focus on human computer interaction (HCI). UMLi provides its own diagram type for modelling interactions between users and the user interface. MoLIC also introduces diagram types for modelling interactions. It focuses on the user's actions and goals and deals with system actions (domain operations in our sense) from a very abstract point of view. Moreover, MoLIC does not intend to generate machine-executable or interpretable interaction definitions, but classifies itself as a means of communications between software development and potential users. We take a different approach to interaction modelling, as we do not aim at UI design, but primarily focus on the aspect of distributed execution and control of interactions. . Interactions in our sense may be also addressed as collaborations. They take place between arbitrary systems, which may incorporate UI clients as well as autonomous systems. We concentrate on modelling the distribution of activities and actions and the corresponding data. Moreover, we did not want to introduce new types of diagrams

Strengths

We argue that using a de-facto standard modelling language is an advantage of our procedure. We refrain from introducing special diagram types, which allows utilizing readily available tools without further adaptations. Another benefit we see in our procedure is that we require only few rules to cover all aspects for a distributed execution of interactions, keeping our procedure simple.

Using a separate, reusable infrastructure has two major advantages: it reduces the effort for implementation as the functionality for executing and controlling interactions as well as distributing corresponding data is already provided. Furthermore, it supports architectural layering suggested by Evans [Ev04]. We extend Evans' suggestion with the inclusion of interactions as a separate layer. Having a layered architecture increases flexibility by limiting the dependencies to a few well known points, thus improving maintainability and facilitating changes.

As propagated in MDSD [SV06], models are considered equal to code. We follow a similar point of view, generating a machine-readable representation of our models and subsequently use an infrastructure to interpret them. As with MDSD, our procedure mandates the (transformed) model as an integrated part of an interactive or collaborative application avoiding extra effort to keep the documentation up-to-date.

Limitations

Besides its strengths, the procedure has some weaknesses which need further consideration.

One major issue concerns the current form of modelling interaction data. Lexical mapping for connecting data properties is error prone as even simple typos may cause fatal runtime errors. Furthermore, when modelling complex interactions, the sheer number of properties might become increasingly confusing. The improvement to modelling interaction data should provide a clear and straightforward method to model data properties while not cluttering the model.

Another aspect has to do with distribution of data. At the moment, interaction data is distributed over system boundaries, regardless if any interaction element executed in the context of a system needs to access a specific data property at all. In other words, interaction data is available globally. An improvement would be to distribute data only to systems, which in fact need to access it. The information of data property usage by systems participating in an interaction could already be drawn from the current model, yet this information is not considered upon transforming the model or at execution time.

A third potential improvement would be supporting parallel execution of interaction elements within an interaction. This applies for parallel execution of interaction elements on one system as well as on different systems. At present, only one system can hold the execution focus at any given time during the processing of an interaction. While this suffices for most cases of client-server applications, collaborations of autonomous

systems may require such a mechanism. A key challenge regarding this issue will be the merging of data written by different execution paths to guarantee consistent data for the interaction.

5 Summary

In this paper we have presented a procedure based on descriptions of interactions in form of activity diagrams to create a machine interpretable model of interaction. We described how UML activity diagrams could be extended to attain such a model.

Despite a number of issues, we think this procedure allows us to achieve our goals of reducing the effort to create and maintain interaction implementations and having a clear separation between domain-code and interaction handling. The procedure proposed does not require specialised modelling tools or new diagram types but only extensions to activity diagrams. We introduced a simple metamodel for representing interactions and explained the transformation of the created UML model into a machine-readable instance of this metamodel.

We also presented a prototypical implementation of a reusable infrastructure which takes the transformed model as input, allowing the distributed execution of the modelled interactions. We explained how this prototype handles the challenges of coordinating distributed interaction execution and distributing data required during the process.

Acknowledgement

This project was sponsored by the initiative “Regionale Wettbewerbsfähigkeit OÖ 2007-2013” funded by the European Regional Development Fund and the state of Upper Austria.

References

[ASQP05] Joao Paulo Almeida, Marten van Sinderen, Dick A.C. Quartel, and Luis Ferreira Pires. Designing Interaction Systems for Distributed Applications. IEEE Distributed Systems Online, 6(3), 2005.

[BS09] Anup Kumar Bhattacharjee and R.K. Shyamasundar. Activity Diagrams: A Formal Framework to Model Business Processes and Code Generation. Journal of Object Technology, 1:189–220, January-February 2009.

[CBJ02] E. Cariou, A. Beugnard, and J.M. Jezequel. An Architecture and a Process for

Implementing Distributed Collaborations. In Enterprise Distributed Object Computing Conference, 2002. EDOC '02. Proceedings. Sixth International, pages 132–143, 2002.

[PBL03] María Greco de Paula, Simone Diniz Junqueira Barbosa, and Carlos José Pereira de Lucena. Relating Human-Computer Interaction and Software Engineering Concerns: Towards Extending UML through an Interaction Modelling Language. In Proceedings of the IFIP INTERACT 2003 Workshop, 2003.

[SP03] Paulo Pinheiro da Silva and Norman W. Paton. User Interface Modeling in UMLi. IEEE Software, 20(4):62–69, 2003.

[EF97] E. Evans and M. Fowler. Specifications. In Proceedings of PLoP 97 Conference, 1997.

[Ev04] Eric Evans. Domain Driven Design: Tackling Complexity in the Heart of Business Software. Addison-Wesley, 2004.

[GDW09] A. Grosskopf, G. Decker, and M. Weske. The Process: Business Process Modeling using BPMN. Meghan Kiffer Press, 2009.

[OMG09a] OMG Unified Modeling Language (OMG UML), Infrastructure Version 2.2, 2009.

[OMG09b] OMG Unified Modeling Language (OMG UML), Superstructure Version 2.2, 2009.

[Se03] Bran Selic. The pragmatics of model-driven development. IEEE Software, 20(5):19–25, 2003.

[SH05] Harald Störrle and Jan H. Hausmann. Towards a formal semantics of uml 2.0 activities. In Software Engineering, pages 117–128. Gesellschaft fuer Informatik, 2005.

[SV06] Thomas Stahl and Markus Voelter. Model-Driven Software Development. John Wiley & Sons, 2006.

Security Testing by Telling TestStories

Michael Felderer[1], Berthold Agreiter[2], Ruth Breu[3] and Alvaro Armenteros[4]

Abstract: Security testing is very important to assure a certain level of reliability in a system. On the system level, security testing has to guarantee that security requirements such as confidentiality, integrity, authentication, authorization, availability and non-repudiation hold. In this paper, we present an approach to system level security testing of service oriented systems that evaluates security requirements. Our approach is based on the Telling TestStories methodology for model–driven system testing. After the elicitation of security requirements, we define a system and a test model. The test model is then transformed to executable test code. We show how traceability between all artifacts can be established and how the tests can be executed focusing on security relevant aspects. All steps are explained based on an industrial case study.

1 Introduction

While testing functional system requirements is one of the core software engineering disciplines, testing security requirements is a new emerging field. We contribute by defining and executing security tests based on the Telling TestStories (TTS) approach [FBCO+09], a methodology for model–driven system testing of *service oriented systems*. Service oriented systems consist of a set of independent components interacting via services to execute collaborative or managed processes. Based on a taxonomy of requirements, TTS defines a system model for a service oriented system and a related test model that invokes service operations of the system model. All model artifacts and the executable services are traceable. More details on TTS are presented in [FBCO+09, FFZ+09].

The work at hand shows how security requirements can be specified as functional requirements according to the TTS methodology such that tests of security requirements can be treated like functional system tests. Functional security tests, as defined in our approach, are more powerful than tests with other security testing approaches because our methodology extends the set of testable security requirements.

Based on the definition of security requirements we design a system model and test model containing test stories that are traceable to security requirements. Test stories are then transformed to executable test code which makes security requirements executable. We also show how the tests can be executed by integrating the test component as passive

[1] Institute for Computer Science, University of Innsbruck, Technikerstr. 21a, 6020 Innsbruck, Austria, michael.felderer@uibk.ac.at
[2] Institute for Computer Science, University of Innsbruck, Technikerstr. 21a, 6020 Innsbruck, Austria berthold.agreiter@uibk.ac.at
[3] Institute for Computer Science, University of Innsbruck, Technikerstr. 21a, 6020 Innsbruck, Austria, ruth.breu@uibk.ac.at
[4] Telefónica I+D, C/ Emilio Vargas 6, 28043 Madrid, Spain, aap@tid.es

participant into the process under test. The methodology is demonstrated by an industrial case study.

The paper is structured as follows. In the next section, we present our case study and show how security requirements can be modelled and tested with TTS. In Section 3 we provide related work and finally in Section 4, we sum up and draw conclusions.

2 Security Modeling and Testing

This section explains our security testing approach by TTS based on an industrial case study[5] to control the network access of clients in a home network. Depending on client properties such as the age of a user or the status of the installed anti–malware application, the network access control applies policies, e.g. that an underage user may only be allowed to access a restricted set of resources on the network.

The scenario consists of different peers distributed among the home network and the operator network. These peers are the *Access Requestor* (AR), *Home Gateway* (HG), *PolicyEnforcement Point* (PEP) and the *Policy Decision Point* (PDP). Following service oriented principles [Erl05], each peer shares interfaces defining the terms of information exchange. The AR is the client application to establish and use the connection to the home network. An AR always connects to a HG. The HG is a device installed at the home of customers controlling access to different networks and services (e.g. domotics, multimedia, data services). The enforcement of who is allowed to access which resources on the network is made by an internal component of the HG called PEP. The PEP gets the policy it has to enforce for a specific AR by the PDP which is the only component run by the operator and not the end user herself. Because we have four independent components only interacting via well-defined interfaces to execute a process, the example adheres to our definition of a service oriented system. Furthermore, we are focusing on testing dedicated example sequences (i.e. the test stories) of the system and verify whether certain security requirements hold under such conditions. The TTS framework adheres to a test-driven development approach, thus it allows the execution of test stories in early stages of system development and supports the evolution of the underlying system. Thus, TTS is an ideal choice for a testing framework of the presented system.

2.1 Requirements Model

The requirements are modelled by a requirements hierarchy. It defines a refinement from abstract requirements resp. goals to more detailed requirements. Security requirements and any other type of non–functional requirements can be integrated into this hierarchy in a natural way. For this purpose, we use SysML requirements diagrams [OMG07] and mark security requirements with the stereotype `securityRequirement`. Several classifications of security requirements can be found in the literature, e.g. [Fir03]. In this work we consider confidentiality, integrity, authentication, authorization, availability and non-repudiation. In Figure 1 the security requirements relevant to our case study are represented in a requirements hierarchy.

[5] The case study was kindly provided by Telefónica.

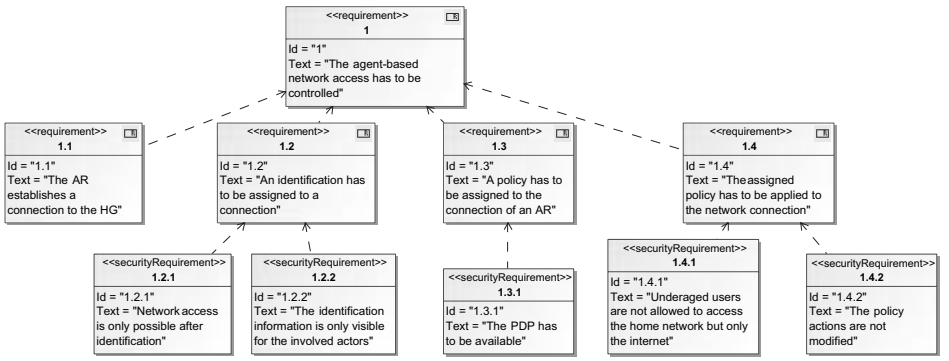


Fig. 1: Requirements

In this representation, to each security requirement as to all other requirements, model elements of the test model (test stories, assertions, test sequence elements) verifying the requirement can be assigned. Normally, compliance to a security requirement will be checked by one or more assertions. In our basic requirements hierarchy, security requirements are formulated as positive statements, defining how a vulnerability can be avoided. Attached test stories may then define possible vulnerabilities that make the requirement fail. In Figure 1 we have defined an example for different types of security requirements. Requirement 1.2.1 is an example for authentication, 1.2.2 for confidentiality, 1.3.1 for availability, 1.4.1 for authorization, 1.4.2 for integrity and 2.1 for non–repudiation.

Requirements traceability refers to the ability to describe and follow the life of a requirement, in both a forwards and backwards direction [GF94]. Traceability has to be guaranteed by a system testing approach to report the status of the system requirements. Our representation of requirements allows for a traceability definition by links between model elements, i.e. by assigning test stories to requirements. Because service operation calls in test stories are linked to service operation calls in the system model which are linked to executable service operations in the system implementation, we have traceability between the requirements model, system model, test model and the executable system.

2.2 System Model

As already mentioned, the AR is the client application to establish and use the connection to the home network. We model this with an `AccessRequest` interface required by the AR. This interface is provided by the HG because an AR always connects to a HG. The data used to decide to which networks a client is granted access is retrieved via the `Identification` interface which is provided by the AR. The HG uses an internal component to enforce these restrictions, the PEP. The PEP receives the policy it has to enforce for a specific AR by the PDP via the `PolicyDecision` interface.

All components in this scenario are connected with each other and the interfaces between them are well-defined (see Figure 2). A request by the AR will trigger the input of user credentials via the identification interface. The data returned by the AR is of type

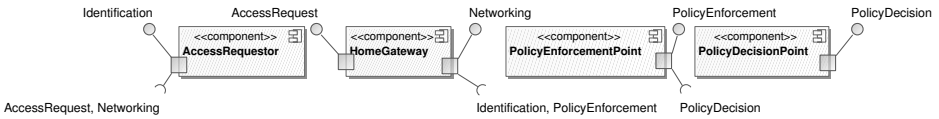


Fig. 2: Actors modelled as components with provided and required interfaces.

`IdentificationData` (see Figure 3). With this information the PDP is able to look up the appropriate policy for the request and send the corresponding list of `PolicyActions` to the the PEP which enforces them. `PolicyAction` and `IdentificationData` are data types defined internally in the system model. The type `IdentificationData` describes a username, a password and optionally attestation data of an AR; the type `PolicyAction` is currently only used to describe to which VLAN the PEP should allow access by a specific AR.

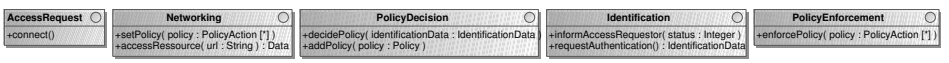


Fig. 3: Interface definitions of services.

The communication among the peers is based on different protocols and standards. Authentication follows IEEE 802.1X[6] which defines a supplicant, an authenticator and an authentication server. In our case the supplicant is the AR, the role of the authenticator is taken over by the HG and the authentication server (a RADIUS[7] database) is represented by the PDP. Note that the system model describes all components, their interfaces and optionally also behavioural parts of the system. For the present contribution we only show the components and interface definitions as it suffices to describe the present scenario.

2.3 Test Model

The TTS test model contains a set of test stories whose execution order is defined in a test sequence. To make all requirements executable, we assign an assertion, a test story or a whole sequence element to it. Due to space limitations we present just one complex and representative test story in Figure 4 and show how the requirements can be mapped to it.

The test story in Figure 4 defines a basic network access scenario containing two assertions. First the `AccessRequestor` connects to the `HomeGateway` (step 1 in Figure 4), which then requests the authentication data containing a username, a password and assessment data from the `AccessRequestor` (steps 2 and 3). This information is forwarded to the `PolicyDecisionPoint` (step 4), which sends a sequence of policy actions to the `HomeGateway` (step 5) based on the identity information of the `AccessRequestor`. We then assert that there has to be a policy action that contains the expected VLAN (`$vlan`) to check the policy actions for integrity. The `HomeGateway` sends the policy actions to the `PolicyEnforcementPoint` (step 6), and then informs the `AccessRequestor` (step 7), which then accesses a specific URL (steps 8 and 9). Finally, we check whether the

[6] Available at `http://www.ieee802.org/1/pages/802.1x-rev.html`.
[7] Remote Authentication Dial In User Service, as specified in RFC 2865.

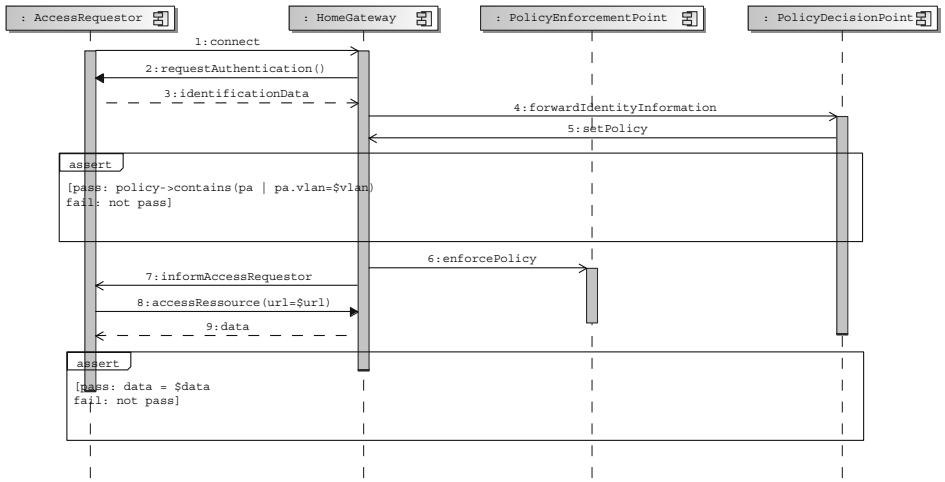


Fig. 4: Test story `TestPolicy`

`accessRessource()` call returns the expected data. Test cases for this test scenario are defined in Table 1.

#TC	$username	$password	$vlan	$url	$data
1	'michael'	'0815'	'HomeNetwork'	'http://74.125.43.99'	webpage_1
2	'michael'	'0815'	'HomeNetwork'	'http://192.168.1.1'	webpage_2
3	'philipp'	'0000'	'Internet'	'http://74.125.43.99'	webpage_1
4	'philipp'	'0000'	'Internet'	'http://192.168.1.1'	null
4	'guest'	'0000'	'Internet'	'http://192.168.1.1'	null

Tab. 1: Test Data Table

The test story is completed by adding some policies to the `PolicyDecisionPoint` in an initial setup. In our case, three policies are added to the `PolicyDecisionPoint`. Each policy assigns a sequence of policy actions, in our basic example just a list set of accessible VLANs, to a username/password combination. The identification data objects are stored in a data pool and are as follows in our example:

```
policy1:('michael','0815',(['Internet','HomeNetwork'])
policy2:('philipp','0000',(['Internet'])
policy3:('*','*',(['GuestNetwork'])
```

This initialization has to be executed before the test story `TestPolicy` can be executed for every test case of Table 1. Test sequence elements can contain additional arbitrations that aggregate the verdicts of the stories' test cases, e.g. such an arbitration could be `pass%=100%`, i.e. all test cases of a test story have to pass. The two assertions in our test story are traceable to requirements. In the requirements model of Figure 1, the first assertion can be assigned to Requirement 1.4.2 testing integrity, and Requirement 1.4.1 testing authorization. Additionally the overall test story can be mapped to Requirement 1.4 which is done implicitly in this case because the test story covers all sub requirements. Test sto-

ries, their states, test sequence elements and traceability for testing other requirements are similar to the one presented in Figure 4 but differ at least in the assertions.

2.4 Test Execution

The case study consists of four different actors communicating with each other. A crucial point of our testing strategy is that the different services are not tested individually and in an isolated way. Instead we define test stories which describe possible sequences of service invocations on the SUT. Testing each service separately is out of scope of our testing strategy. What we are interested in, is the value of certain parameters at specific points in a test story to evaluate assertions.

Another important point of our test execution technique is that the test engine is primarily a passive participant in this process. However, this is not a limitation of the Telling TestStories framework itself, see [FBCO+09]. The reason for a passive execution engine lies in the scenario itself: all actors except the AR are hard-wired to each other. For instance, when the AR sends the `EAP-Response/Identity` message (i.e. the return value of the `requestAuthentication()`-call) to the HG this will trigger a message exchange between HG and PDP. Thus, a central execution engine acting as an orchestration unit is not reasonable in this scenario because it would simply "miss" certain messages. The test execution technique for this scenario starts a test story and only interacts with the `AccessRequestor`. The parameters for this interaction are given in the data table. The remaining communication is only observed by the execution engine. For monitoring this communication we use packet sniffers (TShark[8]) at various points in the environment so that we are able to track the full communication in a non-intrusive way.

Before the test story is started, the system is first set to a specific state. In our case the setup consists of a number of `addPolicy()`-calls to the PDP for installing the policies. After the system is initialized, the execution engine triggers the `connect()`-call by the AR. In the `requestAuthentication()`-call, the HG then requests the user credentials which are provided by the execution engine delivering values for the variables `$username` and `$password` from the data table. The next step involving the AR is the `informAccessRequestor()`-call where the AR is notified about the decision by the PDP. Immediately after this notification the AR can try to access a specific network resource via the `accessRessource()`-call. Again, the parameter for the requested URL is fetched from the data table, i.e. `$url`. The rest of the communication, where the AR is not involved, is only observed.

By monitoring all messages, the execution engine is able to keep track of the current value of variables defined in the interfaces among services, e.g. which `PolicyActions` are returned by the PDP. This information and the content of the data table are sufficient to compose assertions and to check the behaviour of the system. For example, the assertion `[pass: data = $data]` in the test story `TestPolicy` depicted in Figure 4 checks whether accessing a specific URL is allowed/denied as specified in the policy. This assertion can be evaluated by getting the value for `data` from the monitored return value of `accessRessource()` and the value for `$data` from the data table.

[8] Available at `http://www.wireshark.org`.

For each captured message of a running test story, the sniffer matches it to an interaction step of the test model and assigns the values according to the defined interface. After the test execution, the results can be evaluated.

3 Related Work

The topic of model-based testing is well-covered in the literature (see [BJK+05] for an overview) and many tools are already on the market supporting model-based approaches, e.g. [UL06]. However, to the best of our knowledge, the contribution at hand is the first to combine model-based tests on system level with *security functional testing* and *security requirements testing* (cf. [Bis03]).

The aim of functional security tests is mainly the quality assessment of specified (security) requirements. In [JMT08, MJP+07] the authors describe a model-based testing approach for checking whether access control policies are properly enforced by the SUT. The functional model is written in the B language and used for the security test generation from so called *test purposes*. Test purposes are defined as regular expressions and describe a general sequence of operation calls to induce a certain situation on the SUT. The approach aims at the automatic generation of test cases from the SUT. Our approach differs in several points: it supports test-driven development which also implies that test cases are not meant to be generated automatically but modelled by a domain expert (e.g. the customer); for the same reason, TTS allows for the execution of tests during system development. Our approach, furthermore, describes how the information passed among components is to be *interpreted* and how this information can be used to *check for compliance* with arbitrary security requirements and not only access control rules.

Vulnerability scanners, e.g. Nessus[9] constitute a tool-based approach to perform security tests on a very low level. Furthermore, only known vulnerabilities are detectable with such tools.

Other approaches, such as [WJ02], also use the system specification for generating security tests. However, our goal is not the automatic test case generation but a continuous connection among security requirements, tests and the system. Opposed to other security testing approaches, TTS supports a test-driven way of system development. This means that the system model does not have to be complete beforehand.

4 Conclusions

We have presented an approach to security requirements testing of service oriented systems by the methodology of Telling TestStories. Based on an industrial case study, we have defined a requirements model including security requirements, a static system model containing services, a test model and its execution. The test execution itself is accomplished by integrating the test engine as passive component into the process under test. Security requirements are specified as functional requirements. This enables the application of the TTS framework and extends the set of testable security requirements on the system level. We also guarantee traceability between security requirements and the executable system.

[9] Available at `http://www.nessus.org`.

Security testing which is generally considered as very difficult can be addressed by TTS in a clear, structured and intuitive way.

Our next step will be the implementation of our testing methodology on the real industrial system to conduct empirical data and to evaluate the system.

Acknowledgements. This work was partially supported by the SecureChange (ICT-FET-231101) EU project[10], the Telling TestStories project[11] funded by the trans-it and the MATE project funded by the FWF.

References

[Bis03] Matt Bishop. *Computer Security: Art and Science*. Addison Wesley Professional, 2003.

[BJK+05] M. Broy, B. Jonsson, J.P. Katoen, M. Leucker, and A. Pretschner. Model-based Testing of Reactive Systems, volume 3472 of Lecture Notes in Computer Science, 2005.

[Erl05] T. Erl. *Service-oriented Architecture: Concepts, Technology, and Design*. Prentice Hall PTR Upper Saddle River, NJ, USA, 2005.

[FBCO+09] M. Felderer, R. Breu, J. Chimiak-Opoka, M. Breu, and F. Schupp. Concepts for Model-based Requirements Testing of Service Oriented Systems. In *Proceedings of the IASTED International Conference*, volume 642, 2009.

[FFZ+09] M. Felderer, F. Fiedler, P. Zech, , and R. Breu. Flexible Test Code Generation for Service Oriented Systems. 2009. QSIC'2009.

[Fir03] D.G. Firesmith. Engineering Security Requirements. *Journal of Object Technology*, 2(1), 2003.

[GF94] O. C. Z. Gotel and C. W. Finkelstein. An analysis of the requirements traceability problem. 1994.

[JMT08] Jacques Julliand, Pierre-Alain Masson, and Regis Tissot. Generating security tests in addition to functional tests. In *AST '08: Proceedings of the 3rd international workshop on Automation of software test*, New York, NY, USA, 2008. ACM.

[MJP+07] P.A. Masson, J. Julliand, J.C. Plessis, E. Jaffuel, and G. Debois. Automatic generation of model based tests for a class of security properties. In *Proceedings of the 3rd international workshop on Advances in model-based testing*. ACM, 2007.

[OMG07] OMG. *OMG Systems Modeling Language*, 2007. http://www.omg.org/docs/formal/2008-11-01.pdf.

[UL06] M. Utting and B. Legeard. *Practical model-based testing: a tools approach*. Morgan Kaufmann Publishers Inc. San Francisco, CA, USA, 2006.

[WJ02] G. Wimmel and J. Jürjens. Specification-based test generation for security-critical systems using mutations. *Lecture notes in computer science*, pages 471–482, 2002.

[10] Information available at `http://www.securechange.eu/`

[11] Information available at `http://teststories.info/`

Durchgängige Modellierung von Geschäftsprozessen in einer Service-orientierten Architektur

Stephan Buchwald[1], Thomas Bauer[2] und Manfred Reichert[3]

Abstract: Häufig genannte Ziele einer Service-orientierten Architektur (SOA) sind die bessere Unterstützung und Anpassbarkeit von Geschäftsprozessen sowie das Business-IT-Alignment. Diese werden heute nicht erreicht, da die bei der Implementierung eines Fachprozesses notwendigen komplexen Transformationen in einen ausführbaren Workflow schwer nachvollziehbar sind. Dadurch gehen fachliche Anforderungen verloren und es entsteht ein hoher Aufwand bei späteren Prozessanpassungen. Im vorgestellten Ansatz wird ein sog. Abbildungsmodell eingeführt, das Zugehörigkeiten von Aktivitäten des Fachmodells zu denen des Systemmodells (d.h. technische Spezifikation des Informationssystems) explizit dokumentiert. Dadurch werden im Software-Entwicklungsprozess automatisierte Konsistenzprüfungen zwischen den Modellebenen möglich. Werden später Prozessanpassungen erforderlich, so lassen sich die zu einer fachlichen Aktivität gehörenden technischen Aktivitäten direkt erkennen, was die Durchführung der Anpassung erleichtert. Ein wesentlicher Vorteil unseres Ansatzes besteht darin, dass die Erstellung des Abbildungsmodells nur einen minimalen Aufwand verursacht, da keine komplexen Regeln, sondern nur einfache Beziehungen definiert werden müssen.

1 Motivation

Workflow-Technologie alleine reicht nicht aus, um die Anforderung einer *Service-orientierten Architektur (SOA)* an *Flexibilität* zu erfüllen. Gegenüber der heutigen Praxis ist insbesondere eine Verbesserung des Zusammenspiels zwischen Fachbereichen und IT-Abteilungen bei der Software-Entwicklung erforderlich. Dieser Aspekt wird auch als *Business-IT-Alignment* bezeichnet: Informationssysteme sollen die fachlichen Anforderungen exakter treffen als bisher. Hierzu müssen Informationsverlust und Verfälschungen bei der Entwicklung des (prozessorientierten) Informationssystems reduziert werden. Falls Änderungen an fachlichen Anforderungen erfolgen, sollen diese korrekt, unverfälscht und zeitnah in die Implementierung des Informationssystems einfließen. Außerdem kann es in einer SOA zu Änderungen in der Umgebung kommen, z.B. wenn Services wegfallen (d.h. abgeschaltet werden) oder zu einer neuen Version migrieren.

Um solchen Szenarien flexibel zu begegnen, ist eine zusätzliche Modellebene notwendig, welche die Beziehungen zwischen fachlichen Anforderungen und entsprechender IT-Implementierung nachvollziehbar dokumentiert. Wir verwenden hierfür ein Zwischenmodell, welches im Folgenden *Systemmodell* genannt wird. In dieses Systemmodell müssen fachliche Anforderungen an einen Prozess einfließen. Nur dann werden sie in der tatsächlichen Implementierung des Informationssystems umgesetzt. Hierzu ist ein

[1] Abteilung für Daten- und Prozessmanagement, Daimler AG, stephan.buchwald daimler.com
[2] Abteilung für Daten- und Prozessmanagement, Daimler AG, thomas.tb.bauer@daimler.com
[3] Institut für Datenbanken und Informationssysteme, Universität Ulm, manfred.reichert@uni-ulm.de

Konzept erforderlich, bei dem die Beziehung zwischen fachlichen Anforderungen und ihrer Implementierung im geplanten Informationssystem nachvollziehbar ist. So muss für jede Aktivität des fachlichen Modells (und damit für deren Eigenschaften und Anforderungen) ableitbar sein, in welche Aktivität(en) des Systemmodells sie eingeflossen ist (vgl. Abb. 1). Dies ist normalerweise nicht ohne weiteres erkennbar, da für IT-Implementierungen meist Vorgaben für Aktivitätenbezeichnungen bestehen, die nicht mit denen der Fachabteilungen übereinstimmen.

Ein Beispiel einer einfachen Transformation zeigt Abb. 1 mit der Umbenennung der fachlichen Aktivität `[Änderungsantrag stellen]` in `[HT_..._Request-Change]` im Systemmodell. Während solche Namensunterschiede noch einfach handhabbar sind, sind bei der Abbildung eines Fachmodells auf ein ausführbares Modell meist größere Umstrukturierungen notwendig. Grund dafür ist, dass ein Fachbereich seine Geschäftsprozesse im Regelfall auf einer anderen Abstraktionsebene beschreibt als sie für eine IT-Spezifikation benötigt wird. So werden in einem Fachprozessmodell zahlreiche manuelle Einzelaktivitäten beschrieben, die nicht automatisiert werden sollen, die aber für Verfahrenshandbücher oder Prozesskostenrechnungen relevant sind. Diese werden in einer IT-Spezifikation nicht 1:1 übernommen, sondern zusammengefasst oder ganz weggelassen. Ein Beispiel ist die Aktivität `[Änderung umsetzen]` in Abb. 1. IT-unterstützte Prozessschritte hingegen sind im Fachprozess häufig nur grob beschrieben, so dass sie verfeinert oder sogar zusätzlich hinzugefügt werden müssen. Andere fachliche Aktivitäten werden in mehrere IT-gestützte Aktivitäten (Benutzerinteraktionen, Service-Aufrufe und Datentransformationen) aufgespaltet. So wird in unserem Beispiel die Aktivität `[betroffene Bauteile angeben]` in eine Benutzerinteraktion (Human Task) und einen Service-Aufruf aufgespaltet. Schließlich ist zu beobachten, dass Fehler- und Ausnahmefälle (z.B. Rücksprünge bei unvollständigen Daten) in Fachprozessmodellen meist nicht abgebildet werden, um die Komplexität in dieser frühen Phase gering zu halten.

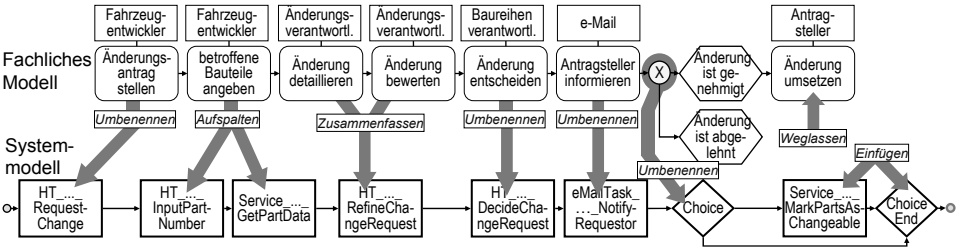


Abb. 1: Transformationen zwischen fachlichem Modell und Systemmodell

Unter Berücksichtigung solcher Transformationen ermöglicht unser Ansatz *Enhanced Process Management by Service Orientation (ENPROSO)* die Nachvollziehbarkeit der Beziehungen zwischen einer fachlichen Aktivität und ihrer IT-Implementierung. Ebenso wird Nachvollziehbarkeit in umgekehrter Richtung unterstützt: So ist es für die robuste Ausführung einer SOA-Anwendung wichtig, die von Umgebungsänderungen betroffenen Prozesse und Aktivitäten zu identifizieren. Nur so wird es möglich, entsprechend schnell auf anstehende Änderungen wie “Service-Abschaltungen“ reagieren zu können.

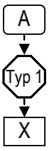
In Kapitel 2 stellen wir die Anforderungen an das zu entwickelnde Konzept vor, insbesondere hinsichtlich der zu unterstützenden Prozesstransformationen. Kapitel 3 skizziert das Abbildungsmodell unseres ENPROSO-Ansatz und beschreibt die dazu notwendigen Prozesstransformationen. Der Ansatz wird in Kapitel 4 detailliert. Kapitel 5 diskutiert verwandte Arbeiten, bevor mit einer Zusammenfassung geschlossen wird.

2 Rahmenbedingungen und Anforderungen

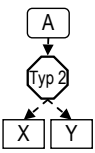
Die Modellierung von Geschäftsprozessen aus fachlicher Sicht dient, neben der Prozessanalyse und -optimierung, vor allem der Dokumentation fachlicher Anforderungen seitens der Fachabteilungen und Prozessverantwortlichen. Da Letztere meist wenig oder keinen IT-Hintergrund haben, werden die Inhalte eines *fachlichen Modells* nicht formal beschrieben. Stattdessen werden einfache graphische Notationen und textuelle Beschreibungen verwendet, wie sie von Geschäftsprozess-Modellierungswerkzeugen angeboten werden (z.B. erweiterte Ereignisgesteuerte Prozess-Ketten (eEPK) in ARIS). Wie erwähnt, sind in dieser frühen Phase noch nicht alle Aspekte im Detail bekannt bzw. sollen aus Komplexitätsgründen noch nicht modelliert werden, so dass das fachliche Prozessmodell (kurz: Fachprozess) an bestimmten Stellen bewusst vage und offen gehalten wird. Diese Unvollständigkeit betrifft die Prozessstruktur (d.h. den Kontrollfluss) selbst, aber auch andere Aspekte (z.B. erfolgt im Fachprozess noch keine detaillierte Festlegung von Datenstrukturen). Hingegen muss ein implementierter Workflow von einer Workflow-Engine ausgeführt werden, weshalb die entsprechende technische Prozessbeschreibung (*ausführbares Modell*) vollständig und formal sein muss. Außerdem muss sie den Vorgaben des von der Workflow-Engine verwendeten Metamodells genügen (z.B. [Rei00]).

Bei der *Transformation* eines Fachprozesses in einen Systemprozess ist es wichtig, dass das Systemmodell geeignet angepasst (d.h. umstrukturiert) wird. Da die durchgängige Realisierung solcher Umstrukturierungen den Schwerpunkt dieses Beitrags bildet, werden die verschiedenen Typen von Strukturänderungen entlang des in Abb. 1 dargestellten (stark vereinfachten) Antragsprozesses für Produktänderungen in der Automobilindustrie motiviert. Dieser stellt sicher, dass Änderungsvorhaben an Bauteilen vor ihrer eigentlichen Umsetzung bewertet, genehmigt und entsprechend dokumentiert werden. Ein Änderungsvorhaben wird angelegt, indem die Änderung in Form eines Antrages beschrieben wird. Da sich Änderungen meist auf mehrere Bauteile auswirken, müssen Informationen über die von ihnen betroffenen Bauteile eingeholt werden. Anschießend wird die Änderung detailliert und bewertet. Abhängig von dieser Bewertung wird entschieden, ob das Änderungsvorhaben umgesetzt wird oder nicht.

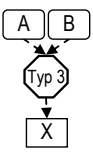
Im Folgenden werden häufig vorkommende Transformationstypen beschrieben.



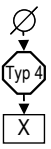
Typ 1 (Übernahme einer Aktivität inkl. Umbenennung): Im einfachsten Fall wird eine Aktivität des Fachprozesses auf genau eine Aktivität des Systemmodells abgebildet. So kann eine Benutzerinteraktion zum Ausfüllen eines Formulars z.B. als *Human Task* [Agr07] in einem BPEL-Prozess realisiert werden. Da für Aktivitäten auf IT-Ebene aber häufig Namenskonventionen existieren (z.B. um die zugehörige Applikation im IT-Betrieb erkennen zu können), ergeben sich meist unterschiedliche Namen für Aktivitäten und Daten im fachlichen Modell und im Systemmodell. Zwecks Nachvollziehbarkeit müssen wir solche Anpassungen explizit verwalten. Beispielsweise wird die fachliche Aktivität `[Änderungsantrag stellen]` durch die Human Task `[HT_…_ChangeAppl_ProductDevelopment_RequestChange]` im Systemmodell realisiert (vgl. Abb. 1).



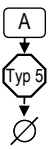
Typ 2 (Aufspalten einer Aktivität): Service-orientierte Workflow-Engines erfordern eine strikte Unterscheidung von Aktivitäten mit Benutzerinteraktion (*Human Tasks*) und Service-Aufrufen (Invoke-Aktivität in BPEL). Diese Aufspaltung ist neu: Klassische Workflow-Engines haben Aktivitäten als größere Einheiten betrachtet, die sowohl mit dem Benutzer interagieren als auch Daten mit Backend-Systemen austauschen. Da solche Einheiten aber kaum wiederverwendbar sind, entsprechen sie nicht der Grundphilosophie einer SOA. Im Beispiel aus Abb. 1 ist es deshalb notwendig, die fachlich zusammengehörige Aktivität `[betroffene Bauteile angeben]` in eine Benutzerinteraktion zur Eingabe der Bauteilnummern und einen Service-Aufruf zur Ermittlung der restlichen Bauteildaten aus einem Produktdaten-Management (PDM)-System aufzuspalten



Typ 3 (Zusammenfassen von Aktivitäten): Bei der fachlichen Analyse von Geschäftsprozessen werden logisch zusammengehörige Aufgaben ermittelt, die mittels separaten Aktivitäten modelliert werden. Wenn solche Aktivitäten z.B. unmittelbar aufeinander folgen und stets von derselben Person erledigt werden, kann es Sinn machen, diese im Systemmodell zu einer Aktivität zusammenzufassen, um sie dann durch eine Sequenz von Bildschirmmasken zu bearbeiten. Im dargestellten Beispiel wird es dem Änderungsverantwortlichen dadurch erspart, nach Beendigung der Detaillierung, dieselbe Workflow-Instanz in seiner Arbeitsliste erneut zu suchen, um anschließend die Bewertung der Änderung durchzuführen.



Typ 4 (Einfügen zusätzlicher Aktivitäten in das Systemmodell): Nachdem der Baureihenverantwortliche eine Entscheidung über die Änderung getroffen hat und der Antragsteller entsprechend informiert ist, kann die Änderung durchgeführt werden. Damit dies im PDM-System möglich ist, müssen dort die entsprechenden Bauteile in den Zustand `Veränderbar` versetzt werden. Dies erfolgt durch den Serviceaufruf `[Service_MarkPartsAsChangable]`, der zusätzlich in das Systemmodell eingefügt wird. Oftmals sind auch zusätzliche Aktivitäten zur Protokollierung relevanter Aktionen oder eingetretener Fehler auf Workflow-Ebene erforderlich.



Typ 5 (Weglassen von Aktivitäten aus dem fachlichen Modell): In Fachprozessen können sich häufig Aktivitäten befinden, deren Ausführung nicht vom Workflow-System gesteuert bzw. überwacht werden soll. So erfolgt in unserem Beispiel die Umsetzung einer Änderung selbstständig durch den Antragssteller, nachdem dieser informiert wurde, dass sein Antrag genehmigt ist. Die Modellierung dieser Aktivität im Fachprozess ist zwar sinnvoll, da sie nicht unerheblich zu den Prozesskosten und -durchlaufzeiten beiträgt (sie fließt z.B. bei einer Prozess-Simulation ein), ist aber für die Workflow-Ebene nicht relevant.

Ferner können weitere Transformationstypen definiert werden (z.B. *n* zu *m* Aktivitäten).

3 Konzept für die Prozesstransformation

Wir entwickeln nun ein Konzept, um ausgehend von einem fachlichen Prozessmodell die erwähnten Transformationen durchzuführen. Hierzu wird eine mehrstufige Vorgehensweise vorgestellt: Zuerst wird der Fachprozess in einen korrespondierenden Prozess auf Systemmodellebene (Systemprozess) transformiert. Aus diesem wird später das ausführbare Prozessmodell abgeleitet. Außerdem wird aufgezeigt, wie Beziehungen zwischen diesen Ebenen in Prozessmodellierungsumgebungen explizit dokumentiert werden können. Dieser Aspekt wird in Kapitel 4 detailliert.

Ebenen der Prozessmodellierung: Der Fachprozess dient der Dokumentation von fachlichen Anforderungen an das zu realisierende Informationssystem. Fachliche Anforderungen werden meist durch Befragung von Endanwendern und Prozessverantwortlichen bzw. -eignern ermittelt. Häufig wird diesen Personen der Fachprozess graphisch dargestellt, so dass sie ihre eigenen Prozessschritte in den Fachprozess einordnen oder Fehler erkennen können. Deshalb ist die wichtigste Anforderung an ein fachliches Modell, dass es leicht verständlich ist. Zudem liegt die Verantwortung für die Erstellung des *Fachmodells* meist bei den jeweiligen Fachbereichen, auch wenn die operative Umsetzung dieser Aufgabe oftmals durch (externe) Berater erfolgt. Für die Inhalte sind die Fachbereiche verantwortlich, da nur diese das entsprechende Fachwissen haben. Bei der Modellierung werden primär die Struktur des Prozessablaufs (Kontrollfluss) mit seinen Aktivitäten, deren Ein- und Ausgabedaten sowie zuständigen Bearbeitern festgelegt.

Die Verantwortung für die Erstellung des *Systemmodells* liegt beim IT-Bereich. Durchgeführte Änderungen am Systemmodell sollten vom zuständigen Fachbereich bestätigt werden. Deshalb muss die Darstellung des Systemmodells (z.B. graphische Notation, Struktur) so gewählt werden, dass sie für Fachanwender verständlich ist. Die zugehörigen Inhalte sind dieselben wie im fachlichen Modell. Allerdings müssen diese ausreichend detailliert, vollständig und formal sein, um die Basis einer plattformunabhängigen IT-Spezifikation bilden zu können. Das bedeutet, vage textuelle Beschreibungen und offene Aspekte aus dem Fachmodell müssen ersetzt werden.

Die Software-Realisierung (*ausführbares Modell*) hingegen wird von IT-Implementieren erstellt. Diese treffen keine (fachlich relevanten) Entscheidungen bei der Erstellung des

ausführbaren Modells, sondern übernehmen stattdessen Struktur und Inhalte des Systemmodells 1:1. Für das ausführbare Modell sind zahlreiche weitere zielplattformabhängige Informationen notwendig, wie Datenobjekte, implementierte Services, Benutzerschnittstellen (z.B. als Maskenentwurf), Geschäftsregeln und zugrundeliegende Organisationsmodelle. Aufgrund des hierfür notwendigen formalen Charakters und der technischen Detaillierung kann es ausschließlich von IT-Spezialisten erstellt werden. Da solche in Fachbereichen nicht verfügbar sind, liegt die Verantwortung für diese Modellebene beim IT-Bereich. In vielen Unternehmen, die sich als IT-Anwender verstehen, gibt es solche Verantwortlichen nicht. Deshalb wird die Erstellung des ausführbaren Prozessmodells häufig an externe Software-Hersteller vergeben (vgl. *Offshoring*).

Wie Abb. 2 zeigt, beinhalten alle drei Modellebenen relevante Aspekte wie Datenobjekte, Geschäftsregeln, Services oder Organisationsmodelle [Rei00]. Diese werden in den unterschiedlichen Ebenen (ausgehend vom Fachmodell) verfeinert. Änderungen an diesen Aspekten werden durch Fachbereiche bestätigt, und schließlich durch den IT-Bereich implementiert. Außerdem stehen die verschiedenen Objekttypen in Beziehung zueinander: So erzeugt ein Prozess verschiedene Datenobjekte, verwendet Geschäftsregeln und ruft Services auf. Da die Umstrukturierung des Kontrollflusses und einzelnen Aktivitäten im Fachmodell (vgl. Kapitel 2) die größte Herausforderung bzgl. Modelltransformationen darstellt, fokussieren wir im Folgenden darauf.

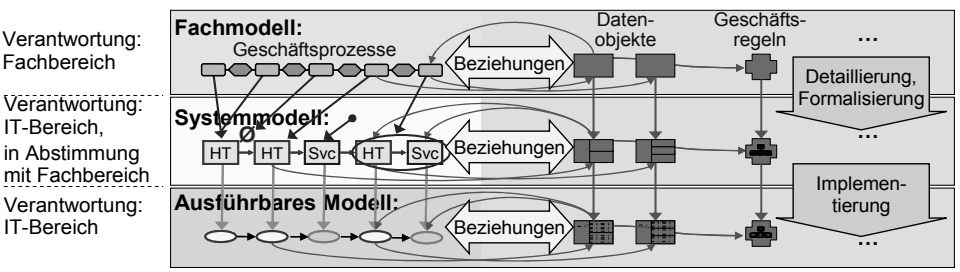


Abb. 2: Ebenen der Modellierung von Prozessen und auch anderer Aspekte

Beziehung zwischen Fach- und Systemprozess: Eine wesentliche Anforderung ist die Nachvollziehbarkeit von Transformationen. In [BBR09] untersuchen wir, welche Varianten bei der Transformation eines Fachprozesses in einen Systemprozess prinzipiell möglich sind. Um die Vielfältigkeit entsprechender Transformationstypen vernünftig zu handhaben, ist eine explizite Repräsentation der Transformationsschritte zur Abbildung von Fachprozessen in Systemprozesse (und umgekehrt) nötig. Dazu führen wir im Folgenden einen neuen Modelltyp ein, dessen Instanzen wir als *Abbildungsmodell* bezeichnen. Es zeigt für alle Aktivitäten des Fachprozesses auf, wie sie in den Systemprozess überführt werden. Ebenso ist für alle Aktivitäten des Systemprozesses erkennbar, aus welchen fachlichen Aktivitäten sie entstanden sind.

Zweck des Abbildungsmodells ist es, Beziehungen zwischen Aktivitäten des Fach- und Systemmodells zu dokumentieren. Deshalb wird keine Reihenfolge zwischen ihnen beschrieben. Ist zwischen den Ebenen eine Werkzeuggrenze zu überwinden, müssen nur fachliche Aktivitäten in das (Modellierungs-) Werkzeug des Systemmodells importiert

werden (und kein Kontrollfluss). Dies ist einfach möglich, da keine Überführung des Fachprozesses in ein anderes Metamodell notwendig ist.

Es lassen sich alle Transformationstypen dokumentieren. Der Ansatz ist erweiterbar, indem zusätzliche Transformationstypen für evtl. zukünftig benötigte Transformationen definiert werden. Alle Veränderungen sind direkt erkennbar und die Beziehungen von Aktivitäten des Fach- und Systemmodells sind bidirektional nachvollziehbar.

4 Gestaltung des Abbildungsmodells

Ein Abbildungsmodell stellt das Verbindungsglied zwischen Fachprozess und Systemprozess dar. Es wird von heutigen Prozessmodellierungswerkzeugen nicht direkt unterstützt. Wir zeigen im Folgenden, wie ein Abbildungsmodell prinzipiell gestaltet werden muss, damit die Beziehung zwischen Fach- und Systemprozess nachvollziehbar ist.

Erstellung und Speicherung: Das Abbildungsmodell wird während der Erstellung des Prozesses auf Systemmodell-Ebene (vgl. Abb. 2) erzeugt. Da beide Modelle in der Verantwortung des IT-Bereichs liegen, sollten dasselbe Modellierungswerkzeug und möglichst dieselbe Notation verwendet werden. Allgemein betrachtet definiert das Abbildungsmodell eine Menge von Relationen, die Aktivitäten des Fachprozesses auf Aktivitäten des Systemmodells abbilden. Jede dieser Relationen entspricht genau einem Transformationstyp. In der Modellierungsumgebung sollte eine Relation durch jeweils ein Objekt realisiert werden, dem außer dem Transformationstyp auch Attribute wie Name, Beschreibung oder der Verantwortliche aus dem Fachmodell zugeordnet sind.

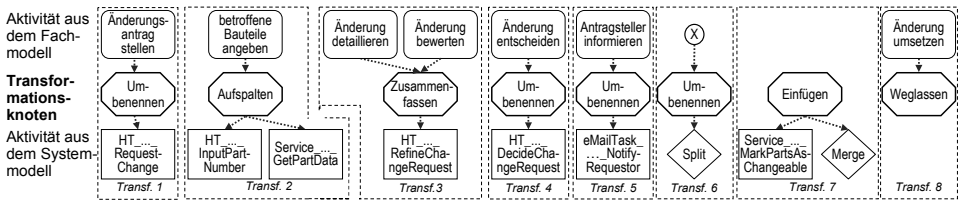


Abb. 3: Abbildungsmodell für das in Abb. 1 vorgestellte Beispielszenario

Da heutige Modellierungswerkzeuge das Konzept des Abbildungsmodells nicht unterstützen, muss bei ihrer Verwendung für diesen Zweck ein existierender Modelltyp verwendet werden (z.B. eEPK, BPMN oder UML Activity Diagram). Dieser muss Aktivitäten aus dem Fach- und Systemprozess sowie Relationen zwischen diesen darstellen können. Je nach Notation und Werkzeug kann hiervon ein Modelltyp für Abbildungsmodelle abgeleitet werden. In diesem abgeleiteten Modell können dann spezielle Knotentypen für Transformationen (Transformationsknoten) sowie Kantentypen für Transformationskanten verwendet werden. Abb. 3 zeigt exemplarisch das zu dem in Abb. 1 eingeführten Beispiel gehörende Abbildungsmodell in einer „neutralen“ Notation. Hier werden die Transformationsknoten durch Achtecke repräsentiert, um sie leicht von Aktivitäten unterscheiden zu können; Transformationskanten werden gestrichelt dargestellt.

Verwendung des Abbildungsmodells: Durch Konsistenzanalysen wird es nun möglich, eine schnelle Zuordnung (und damit Umsetzung) von geänderten fachlichen Anforderungen zu Aktivitäten des ausführbaren Modells zu erhalten:

1. Nach Änderung des Fachprozesses werden dessen Aktivitäten in das Abbildungsmodell importiert. Die Konsistenzanalyse vergleicht anschließend die im Abbildungsmodell aktualisierte Menge fachlicher Aktivitäten mit der bereits vorhandenen. Sind bestimmte Aktivitäten nicht mehr enthalten, wurden diese aus dem Fachprozess gelöscht (bzw. zusätzliche wurden erzeugt).
2. Verwendet das Werkzeug zur Fachprozessmodellierung zugreifbare Zeitstempel für Aktivitäten-Objekte, sind Veränderungen einzelner fachlicher Aktivitäten erkennbar. Die Konsistenzanalyse vergleicht nicht nur die aktualisierte mit der bereits vorhandenen Aktivitätsmenge, sondern zusätzlich die Zeitstempel einzelner Aktivitäts-Objekte.
3. Analog können bei Veränderungen in der Umgebung der Prozessapplikation, etwa die Abschaltung eines Services (vgl. Abb. 1, *`Service_GetPartData`*), mit diesem Ansatz leicht betroffene fachlichen Aktivitäten (`betroffene Bauteile angeben`) ermittelt werden.

5 Verwandte Arbeiten

MDA-basierte Ansätze wenden Patterns auf fachliche Modelle an [All07, Bau08, FKT08]. Allerdings sind solche Ansätze nicht immer realisierbar. Dies trifft auch auf unser Szenario zu, bei dem eine *freie* Modellierbarkeit des Systemmodells gefordert ist: So ist für Geschäftsprozesse, die fachlich meist grob, sehr abstrakt und vage beschrieben sind, eine strukturelle Adaption auf Systemmodell-Ebene notwendig. Eine solche Überarbeitung mittels automatisch anwendbaren Patterns zu realisieren, wäre zu aufwendig, da umfangreiche Transformationen fachlicher Objekte auf Objekte im Systemmodell formal spezifiziert werden müssten. Zudem sind Transformationen, wie das Einfügen von zusätzlichen Aktivitäten, schwer realisierbar, da für diese keine entsprechende Aktivität im fachlichen Modell existiert. In dem von uns betrachteten Anwendungsfall ist eine Wiederverwendung vordefinierter Patterns in unterschiedlichen Geschäftsprozessen nicht realistisch. Deshalb muss für jeden Geschäftsprozess individuell entschieden werden, wie eine entsprechende technische Repräsentation im Systemmodell aussieht.

6 Zusammenfassung

Von Fachbereichen erstellte Prozessmodelle müssen strukturell adaptiert werden, bevor sie mittels Prozess-Management-Systemen implementiert werden können. Hierbei ist es das Ziel, möglichst schnell und nachvollziehbar von fachlichen Anforderungen zur IT-Implementierung zu gelangen (Business-IT-Alignment). Um dies zu erreichen, haben wir zwischen dem fachlichen und dem ausführbaren Prozessmodell die Ebene des Sys-

temmodells eingeführt, das in Verantwortung des IT-Bereichs liegt und als Spezifikation für die IT-Implementierung dient. Um bei der Modellierung des Systemmodells die durchgeführten Prozesstransformationen nachvollziehbar zu gestalten, haben wir das Konzept des Abbildungsmodells entwickelt. Es ermöglicht weiterhin, die bei der Erstellung des Systemmodells benötige „freie Geschäftsprozess-Modellierung", aber erzielt dennoch einen hohen Grad an Nachvollziehbarkeit. Diese Nachvollziehbarkeit wird genutzt, um Prozessimplementierungen schneller erstellen bzw. anpassen zu können. Sie gewährleistet zudem die Konsistenz zwischen den verschiedenen Modellebenen. Entwurfsalternativen und eine prototypische Umsetzung sind in [BBR09] dargestellt.

Literaturverzeichnis

[Agr07] A. Agrawl et al.: WS-BPEL Extension for People Specification. Technical Report, Active Endpoints, Adobe, BEA, IBM, Oracle, SAP AG, 2007

[All07] T. Allweyer: Erzeugung detaillierter und ausführbarer Geschäftsprozessmodelle durch Modell-zu-Modell-Transformationen; Proc.: 6. Workshop Geschäftsprozessmanagement mit Ereignisgesteuerten Prozessketten, St. Augustin, Germany, S. 23-38, 2007

[Bau08] P. Bauler et.al.: Usage of Model Driven Engineering in the context of Business Process Management; In: Proc. 4th GI Workshop XML Integration and Transformation for Business Process Management, S. 1963 – 1974, 2008

[BBP09] S. Buchwald, T. Bauer, R. Pryss: IT-Infrastrukturen für flexible, service-orientierte Anwendungen; In: Proc. BTW 2009, Münster, 2009

[BBR09] S. Buchwald, T. Bauer, M. Reichert: Durchgängige Modellierung von Geschäftsprozessen durch Einführung eines Abbildungsmodells: Ansätze, Konzepte, Notationen. Technical Report UIB-2009-12, University of Ulm, 2009

[FKT08] K.P. Fähnrich; S. Kühne; M. Thränert: Model-Driven Integration Engineering; Eigenverlag Leipziger Informatik-Verbund (LIV), 2008

[Rei00] M. Reichert: Dynamische Ablaufänderungen in Workflow-Management-Systemen. Dissertation, Universität Ulm, Fakultät für Informatik; 2000

Model Transformation Chains in Model-Driven Performance Engineering: Experiences and Future Research Needs

Mathias Fritzsche[1], Wasif Gilani[2], Ralf Lämmel[3] and Frédéric Jouault[4]

Abstract: We gained experiences in implementing rule based model transformations within an industrial case study called Model-Driven Performance Engineering (MDPE). Similar to other MDE scenarios, these transformations have been implemented via multiple transformation steps interconnected in an automated model transformation chain. In this short paper, we use the MDPE case study to demonstrate reasons for decomposing model transformations and discuss disadvantages in terms of execution costs. Based on these experiences, we propose, as an input for future research, an architecture to optimize decomposed model transformation chains.

1 Introduction

Early usage of model transformation approaches has generally been limited to scenarios involving a single transformation from one source model to one target model. This is notably the case for QVT — as illustrated by the list of examples from its specification [Obj08]; it is also the case for ATL [JABK08]. As more complex problems have been tackled, the number of transformations involved in a given solution has increased. Hence, more complex transformations are organized in chains (or more general forms of composite transformations) with the output of some transformations being fed as input to others. In fact, most real-world MDE scenarios seem to involve chains of multiple transformations, e.g., twelve in the interoperability scenario for business rules presented in [FABJ09], five in the interoperability scenario for code clone tools presented in [SDJ+09], and five in the scenario for performance engineering presented in [FPG+09].

In this paper, we use the latter scenario to demonstrate reasons for model-transformation chains to appear in MDE scenarios.

It is clear that modular transformations (say, transformation chains, in particular) is an established technique in the broader field of software transformation. One example is an

[1] SAP Research CEC Belfast, mathias.fritzsche@sap.com
[2] SAP Research CEC Belfast, wasif.gilani@sap.com
[3] Universität Koblenz-Landau, Germany, rlaemmel@acm.org
[4] AtlanMod Team (INRIA-EMN) - Ecole des Mines de Nantes, France, frederic.jouault@inria.fr

aspect-oriented weaver, e.g. for .NET [LC03], which will reasonably operate at the level of byte code so that it can provide flexibility with regard to the specific .NET languages that are used for components and aspects.

We observed that decomposing a transformation into a chain may make *future extensions* less costly in terms of development costs in the case of extensions. Another issue is the decoupling of different concerns by addressing them in distinct transformations. However, we also observed the extra runtime costs that merely arise from managing transformation chains. Therefore, the paper proposes an architecture of a tool for optimizing model transformation chains. This architecture deals with the specific properties of most rule based model to model transformation languages, such as ATL or QVT, compared to traditional programming languages.

The paper is organized as follows. Section 2 presents the MDE case study including a discussion of the MDPE transformation steps. Based on this, we describe future research needs to merge decomposed model transformation steps 3. Section 4 concludes the paper.

2 Industrial Model-Transformation Case Study - Model-Driven Performance Engineering

Model-Driven Performance Engineering (MDPE) [FPG+09] is an architecture to extend existing *Process Modelling Tools* [FG09] with multi-paradigm performance decision support functionality. The BPMN [Obj06] based modelling tool of the SAP NetWeaver BPM Suite [SRMS08] and the JPASS based modelling tool of the jCOM! BPM Suite are examples of Process Modelling Tools which are currently supported via MDPE. The provided performance decision support functionality answers questions like (1) Can available staff handle future business growth? (2) How many employees are needed at which point in time? (3) Which are the most sensitive resources of the process?

All questions can be answered based on discrete event simulations, e.g. the tool AnyLogic [XJ 09] and/or analytical performance analysis approaches, e.g. the FMC-QE approach [PKFR10]. In some cases (questions 2 and 3), additional optimization or sensitivity algorithms are needed to be utilized in order to guide the performance analysis.

The MDPE architecture does not re-implement the required performance analysis engines, but makes reuse of existing ones, for instance, the AnyLogic simulation engine and the analytical FMC-QE tool. Integration of different engines permits to utilize strengths of different underlying performance prediction methodologies as discussed in [PKFR10].

Concluding, MDPE needs to interconnect *n* Process Modelling Tools with *m* Performance Analysis Tools. Figure1 shows an example transformation chain realizing this interconnection. A detailed description of the chain can be found in [FG09]. In the following subsections, a summary of reasons for the decomposed transformation is provided.

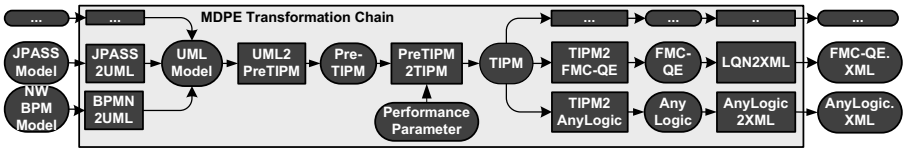


Fig. 1: MDPE Transformation chain for interconnecting two or more Process Modelling Tools with two or more Performance Analysis Tools as Block Diagram ([KGT06])

2.1 Integration of new Performance Analysis Tools

The central part of this chain is the so called Tool-Independent Performance Model (TIPM). This model particularly supports the combined use of m Performance Analysis Engines as it represents the common data base for a number of such engines, similar to the related "Core Scenario Model" in the PUMA architecture [WPP+05].

The TIPM permits the reduction of $n * m$ transformations for interconnecting chains of Process Modelling Tools with Performance Analysis Tools, into $n + m$. However, the TIPM structure is specialized for the use of Performance Analysis Tools and significantly differs from the structure of Process Modelling Languages. Therefore, UML Activity Diagrams are added to the transformation chain to simplify adaptation of MDPE for new Process Modelling Tools.

2.2 Integration of new Process Modelling Tools

The structures of most process modelling languages, such as BPMN, JPASS and UML Activity Diagrams are related, as they are close to Petri-nets. Therefore, it is sufficient to add an intermediate model into the transformation chain of Figure 1 in order to express process behaviour. We chose UML Activity Diagrams for this model due to the fact that this language is broadly used and supported by a numerous tools [FG09]. Additionally, one can apply formally defined Petri-net semantics to a subclass of UML Activity Diagrams, as discussed in more detail by Dehnert [Deh03].

We experienced that especially the transformation from BPMN to UML is close to a one to one mapping, whereas the UML to TIPM transformation is more complex as performance parameters, such as information about resource demands, sharing of resources between different process instances, etc. have to be taken into account. Thus, we would not be able to reuse the already existing complex UML to TIPM transformation every time when a new Process Modelling Tool is integrated into MDPE.

2.3 Separation of Concerns

Figure 1 shows that there is an additional transformation step between the TIPM and the generated input for the Performance Analysis Tools (see "FMC-QE.XML" and "AnyLogic.XML"). This step is caused by the fact that we have to deal with two concerns.

First, as the structural transformation concern, we were required to perform a structural transformation from the TIPM structure to the AnyLogic structure. Second, as the representation concern, the AnyLogic tool uses a specific XML format as input. Therefore, we were required to apply XML formatting so that our generated Performance Analysis Model can be read by the AnyLogic tool.

This is similar to the transformation between UML and TIPM. First, we had to translate the UML structure into the TIPM structure and, second, we had to consider Performance Parameters in order to generate a TIPM. Most of the Performance Parameters, such as number of process instances that are intended to be executed or resource demands of process steps, are simply transformed into attributes of the TIPM. However, some parameters, such as parameters about sharing of resources between different process instances, make it necessary to change the previously generated TIPM structure.

2.4 Penalties for the Transformation Chain in terms of Runtime Costs

The provided reasons for the MDPE model transformation chain can be summarized as means to reduce development effort for likely changes, such as integrating new Process Modelling Tools. A discussion of the runtime costs of this approach follows.

Due to the MDPE transformation chain we have to deal with a number of models which are unnecessary for the original task of transforming a Process Model to a Performance Analysis Model. For instance, one UML model and two different TIPMs need to be generated as intermediate models in case we execute a four-step transformation from BPMN to AnyLogic (see Figure 1). All additional models have to be stored, at least, while the transformation chain is executed. In case of a monolithic transformation approach, these intermediate models would not be required. Moreover, it is necessary to trace performance analysis results back through the model transformation chain. Hence, each transformation in the chain does not only have the direct transformation result as output but also a trace model [FJA+09], which stores information about which model element(s) are transformed to which model element(s).

For the transformation chain, we measured the memory footprint for a BPMN model with 15 process steps. The current implementation of MDPE uses file based model serialization. All transformations are implemented with ATL. For the measurements, we used a Laptop having 2GB of RAM and using a 2GHz Dual Core CPU. The BPMN model that we used as input for our measurements uses 202Kb of memory. However, only 20Kb of the data is behaviour specific and relevant for the simulation. The remaining information mainly concerns modelled rules. Additionally, we injected 47Kb of data representing Performance Parameters into the MDPE transformation chain. This data is transformed with the MDPE transformation chain to an AnyLogic input model which uses 528Kb of memory. Thus, a monolithic transformation would only require 777Kb (202+47+528) of memory to serialize the input- and output- models. However, the measured memory footprint for the chain with 2274Kb was significantly higher. Also, executing the described transformation currently consumes in average 13.6 seconds.

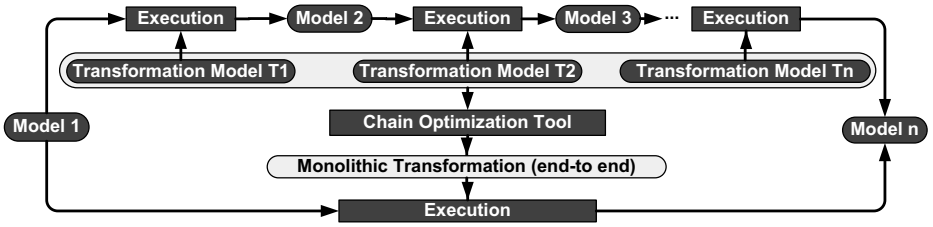


Fig. 2: Transformation chain optimization overview as Block Diagram ([KGT06])

Concluding, we paid the price of additional memory usage. We also believe that the chain significantly contributes to the high transformation execution time. Thus, we experienced the need to optimize model transformation chains to avoid high runtime costs, especially when it comes to larger MDE scenarios. In our case, all intermediate models are serialized text based. Hence, one measure to decrease the data footprint as well as performance would be to access all intermediate models in memory. Another possibility would be to develop an approach for merging decomposed model transformations before they are executed. This topic is explained as a direction for future research in the following section.

3 Proposed Future Research for Model Transformation Chains

Following the experiences that we explained in the previous section we identified the need for *Transformation Chain Optimization Tool* to merge adjacent model transformation steps before they are executed. The following Figure 2 depicts the problem space such tool is required to address. It has to take *n Transformation Models*, such as rule based ATL or QVT scripts, as input and translate them into one monolithic *End-to-End Transformation*. This end-to-end transformation needs to enable, identical to the original chain, the translation of a *Model 1*, which conforms-to a *Meta-Model 1*, into a *Model n*, which conforms-to a *Meta-Model n*.

Figure 3 shows the architecture which we propose to implement such Transformation Chain Optimization Tool. Within this architecture, Higher Order Transformations (HOTs) are employed as one of the underlying concepts. HOTs are transformations that are used to transform one Transformation Model A to a new Transformation Model $A*$. A broad set of applications for HOTs can be found in [TJF+09].

Figure 3 shows the main agents we propose to implement such a Transformation Chain Optimization Tool, namely the *Local Merge (HOT)*, *Analysis HOT* and *Global Merge Function*. The functionality of these agents is explained below.

The Local Merge HOT generates a Transformation Model called *Monolithic Transformation (T1,T2)* based on the *Transformation Model T1* and the *Transformation Model T2*. In our architecture, T1 and T2 represent adjacent transformation steps, which are defined via a rule based transformation language, such as ATL or QVT.

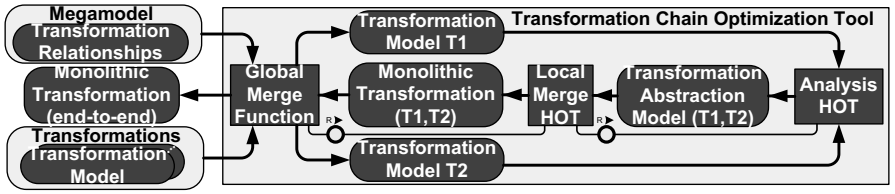


Fig. 3: Direction for Future Research as Block Diagram ([KGT06])

Rule based transformation languages, mix imperative and declarative statements [Jou05]. Therefore, monolithic transformation can not be generated by directly employing state of the art approaches for program optimization, such as deforestation [Wad88] or other kinds of optimization, such as program specialization (i.e. partial evaluation) [MWP+01].

Therefore, the Transformation Model called *Transformation Abstraction Model (T1,T2)* has been added to the Transformation Chain Optimization Tool. This model does not contain imperative statements any more and, thus, represents the pure mapping between the source and target elements of the Transformations Models T1 and T2 on the meta-level. A formal definition of this model is considered as a first topic for future research.

The Transformation Abstraction Model needs to be generated from the transformation scripts T1 and T2. So called *Analysis HOTs* [TJF+09] have already been implemented in order to analyse the input and output of model transformation on the meta-level. The similar implementation of analysis HOTs for the creation of Transformation Abstraction Models based on different rule based model transformation languages, such as ATL or QVT, is considered as a second topic for future research.

The Local Merge HOT takes the Transformation Abstraction Model as an input in order to generate the merged transformation based on T1 and T2. Due to the declarative nature of the Transformation Abstraction Model, we claim that the Local Merge HOT can be implemented more easily as existing kinds of optimization may become applicable more directly. However, as a third topic for future research, these established techniques would still need adaptation to the domain of model transformations because of the special expressivity used for models and transformations. For instance, models can be of graph shape, i.e., they use reference semantics (including aliasing) as opposed to value semantics and tree shape in classical functional programming.

The Local Merge HOT is controlled by the *Global Merge Function* agent (see "R" between the Global Merge Function and the Local Merge HOT). This agent needs to identify adjacent transformation steps. For this task, the application of a *Megamodel* is proposed.

A megamodel expresses relationships between different types of modelling artefacts, such as the definition of a transformation relationship between Transformation Models and intermediate models, which are employed as in- and output of model transformations. The Global Merge Function can iterates over the Megamodel in order to find adjacent transformations, which are then sent to the Local Merge HOT. This HOT sends the merged result back to the Global Merge Function, which then sends the next consecutive transformations

to the Local Merge HOT. Thus, the Global Merge Function works recursively as long as the *Monolithic Transformation(end-to-end)* is found.

4 Conclusions

In this paper we provided an industrial case study to demonstrate reasons for decomposed model transformations. Summarizing, such chains permit a high degree of modularization in order to reduce development effort in the case of likely changes. We also demonstrated increased execution costs in the MDPE transformation chains. Therefore, we identified the need to merge adjacent transformation steps of model transformation chains before they are executed. We proposed an architecture for rule based transformation languages, such as ATL and QVT, which permits merging model transformation chains. The architecture applies different HOTs and a so called Transformation Abstraction Model. The implementation of this architecture for different model transformation languages is proposed as an area for future research.

As another area for future research, we identified that MDE is currently lacking evaluated cost functions, which can be applied for model transformations. With such cost functions in place, we may be able to balance execution and development costs in cases where transformation chain optimizations cannot be applied, such as in the case of a distributed execution of a transformation chain.

References

[Deh03] Juliane Dehnert. *PhD Thesis: A Methodology for Workflow Modeling: From business process modeling towards sound workflow specification*. TU-Berlin, 2003.

[FABJ09] Marcos Didonet Del Fabro, Patrick Albert, Jean Bézivin, and Frédéric Jouault. In *Proceedings of the 5èmes Journées sur l'Ingénierie Dirigée par les Modèles (IDM)*, 2009.

[FG09] Mathias Fritzsche and Wasif Gilani. Model Transformation Chains to integrate Performance related Decision Support into BPM Tool Chains. In *Post-proceedings of the 3rd Summer School on Generative and Transformational Techniques in Software Engineering (GTTSE'09) (to appear)*, LNCS. Springer-Verlag, 2009.

[FJA+09] Mathias Fritzsche, Jendrik Johannes, Uwe Assmann, Simon Mitschke, Wasif Gilani, Ivor Spence, JohnBrown, and Peter Kilpatrick. Systematic Usage of Embedded Modelling Languages in Automated Model Transformation Chains. In *Proceedings of the 1st International Conference on Software Language Engineering (SLE'08), Revised Selected Papers*, volume 5452 of *LNCS*, pages 134–150. Springer-Verlag, 2009.

[FPG+09] Mathias Fritzsche, Michael Picht, Wasif Gilani, Ivor Spence, John Brown, and Peter Kilpatrick. Extending BPM Environments of your choice with Performance related Decision Support. In *Proceedings of the 7th Business Process Management Conference (BPM'09)*, volume 5701 of *LNCS*, pages 97–112. Springer-Verlag, 2009.

[JABK08] Frédéric Jouault, Freddy Allilaire, Jean Bézivin, and Ivan Kurtev. ATL: A model transformation tool. In *Science of Computer Programming*, volume 72, pages 31–39, 2008.

[Jou05] Frédéric Jouault. Loosely Coupled Traceability for ATL. In *Proceedings of the ECMDA workshop on traceability*, 2005.

[KGT06] Andreas Knöpfel, Bernhard Gröne, and Peter Tabeling. *Fundamental Modeling Concepts: Effective Communication of IT Systems.* John Wiley & Sons, 2006.

[LC03] Donal Lafferty and Vinny Cahill. Language-independent aspect-oriented programming. In *Proceedings of the 18th annual ACM SIGPLAN conference on Object-oriented programing, systems, languages, and applications (OOPSLA'03)*. ACM, 2003.

[MWP+01] Dylan McNamee, Jonathan Walpole, Calton Pu, Crispin Cowan, Charles Krasic, Ashvin Goel, Perry Wagle, Charles Consel, Gilles Muller, and Renauld Marlet. Specialization tools and techniques for systematic optimization of system software. *ACM Trans. Comput. Syst.*, 19(2):217–251, 2001.

[Obj06] Object Management Group. Business Process Modeling Notation Specification, Final Adopted Specification, Version 1.0., 2006.

[Obj08] Object Management Group. Meta Object Facility (MOF) 2.0 Query/View/Transformation Specification, Version 1.0, 2008.

[PKFR10] Tomasz Porzucek, Stephan Kluth, Mathias Fritzsche, and David Redlich. Combination of a Discrete Event Simulation and an Analytical Performance Analysis through Model-Transformations. In *Proceedings of the 17th International Conference on the Engineering of Computer-Based Systems (ECBS'10), to appear*. IEEE Computer Society, 2010.

[SDJ+09] Yu Sun, Zekai Demirezen, Frédéric Jouault, Robert Tairas, and Jeff Gray. A Model Engineering Approach to Tool Interoperability. 5452:178–187, 2009.

[SRMS08] Jim Hagemann Snabe, Ann Rosenberg, Charles Molle, and Mark Scavillo. *Business Process Management: The SAP Roadmap.* SAP Press, 2008.

[TJF+09] Massimo Tisi, Frédéric Jouault, Piero Fraternali, Stefano Ceri, and Jean Bézivin. On the Use of Higher-Order Model Transformations. In *Proceedings of the 5th European Conference on Model Driven Architecture - Foundations and Applications (ECMDA-FA'09)*, volume 5562 of *LNCS*, pages 18–33. Springer-Verlag, 2009.

[Wad88] Philip Wadler. Deforestation: transforming programs to eliminate trees. In *Proceedings of the Second European Symposium on Programming*, pages 231–248, Amsterdam, The Netherlands, The Netherlands, 1988. North-Holland Publishing Co.

[WPP+05] Murray Woodside, Dorina C. Petriu, Dorin B. Petriu, Hui Shen, Toqeer Israr, and Jose Merseguer. Performance by unified model analysis (PUMA). In *Proceedings of the 5th International Workshop on Software and Performance (WOSP'05)*, pages 1–12. ACM, 2005.

[XJ 09] XJ Technologies. AnyLogic — multi-paradigm simulation software. `http://www.xjtek.com/anylogic/`, 2009.

Adaptable Model Versioning in Action[1]

Petra Brosch[2], Gerti Kappel[3], Martina Seidl[4], Konrad Wieland[5], Manuel Wimmer[6], Horst Kargl[7] and Philip Langer[8]

Abstract: In optimistic versioning, multiple developers are allowed to modify an artifact at the same time. On the one hand this approach increases productivity as the development process is never stalled due to locks on an artifact. On the other hand conflicts may arise when it comes to merging the different modifications into one consolidated version. In general, the resolution of such conflicts is not only cumbersome, but also error-prone. Especially if the artifacts under version control are models, little support is provided by standard versioning systems.

In this paper we present the enhanced versioning process of the model versioning system AMOR. We show how AMOR is configured in order to obtain a precise conflict report which allows the recommendation of automatically executable resolution patterns. The user of AMOR chooses either one of the recommendations or performs manual resolution. The manual resolution may be in collaboration with other developers and allows to infer new resolution patterns which may be applied in similar situations.

1 Introduction

The development of software systems without version control systems (VCSs) is nowadays unimaginable. Especially optimistic VCSs are of particular importance because such systems effectively manage concurrent modifications on one artifact performed by multiple developers at the same time. Like other software artifacts, models are developed in teams and evolve over time; consequently they also have to be put under version control.

Standard VCSs for code usually perform the conflict detection by line-oriented text comparison of arbitrary artifacts. When applied on the textual serialization of models, the result is unsatisfactory because of the graph-based structure, single changes on the model may result in multiple changed lines in the textual serialization. Considering lines as unit of comparison, the information stemming from the graph-based structure is destroyed and

[1] This work has been partly funded by the Austrian Federal Ministry of Transport, Innovation, and Technology and the Austrian Research Promotion Agency under grant FIT-IT-819584 and by the fFORTE WIT Program of the Vienna University of Technology and the Austrian Federal Ministry of Science and Research.

[2] brosch@big.tuwien.ac.at
[3] kappel@big.tuwien.ac.at
[4] seidl@big.tuwien.ac.at
[5] wieland@big.tuwien.ac.at
[6] wimmer@big.tuwien.ac.at
[7] horst.kargl@sparxsystems.at
[8] philip.langer@jku.at

associated syntactic and semantic information is lost. Consequently, dedicated VCSs for models have been proposed (cf. [Alt08, KGE09, OWK03, MCPW08, CRP08, SZN04, OS06, Kög08, AP05]). However, they either support generic model versioning and therefrom do not consider language-specific aspects or they are built for a specific language and are not suitable for other languages. Furthermore, the focus in such systems is mainly set on the detection of conflicts. The actual resolution is usually left to the user without hardly any dedicated support. The provided resolution facilities are mostly limited to refuse modifications or to perform a manual remodeling. This task is often very repetitive as the same conflict types occur again and again. These systems are not capable to analyze the user's behavior and learn therefrom for similar situations. Finally, the merge is left to the person who does the later check-in. Therefore it is totally her responsibility to obtain a consolidated version containing all modifications in a high quality. Collaborative approaches as for code versioning [DH07], where all responsible developers are involved in the merge process, have not been realized in model versioning systems so far. Consequently, the different intentions of the modelers might not be captured in the merged version and might get lost. As models are often applied for early project phases, where a common understanding and a common terminology might not have been established yet, this loss could cause errors which become obvious only in later phases of the project and which then are very hard and expensive to correct.

The adaptable model versioning system AMOR [AKK+08] aims to combine the advantages of both generic and language-specific VCSs by providing a generic framework with extension points for including language-specific features. AMOR is built around subversion[8] and attaches an extended version of EMF Compare[9] for change detection. Hence, AMOR can deal with arbitrary Ecore[10] based modeling languages. The combination of generic and specific aspects improves conflict detection as well as conflict resolution. When manual resolution is necessary, a collaborative merge process might be initiated. If the resolution is performed manually, it is analyzed in order to derive resolution recommendations for similar situations which occur in future scenarios. In this paper we present the conflict detection and conflict resolution components of AMOR from the workflow perspective and show how their interplay eases the check-in process for models.

The remainder of this paper is structured as follows: In Section 2 we introduce a motivating conflict scenario in which a naive merge would change the originally intended semantics. In Section 3 we show how such conflicts are detected and resolved within the model versioning system AMOR and in Section 4 we present a novel approach for the automatic derivation of a general resolution strategy for similar conflicts. Section 5 gives an overview on related work. Finally, in Section 6, we draw conclusions and sketch some future work.

[8] http://subversion.tigris.org
[9] http://wiki.eclipse.org/index.php/EMF_Compare
[10] http://www.eclipse.org/modeling/emf

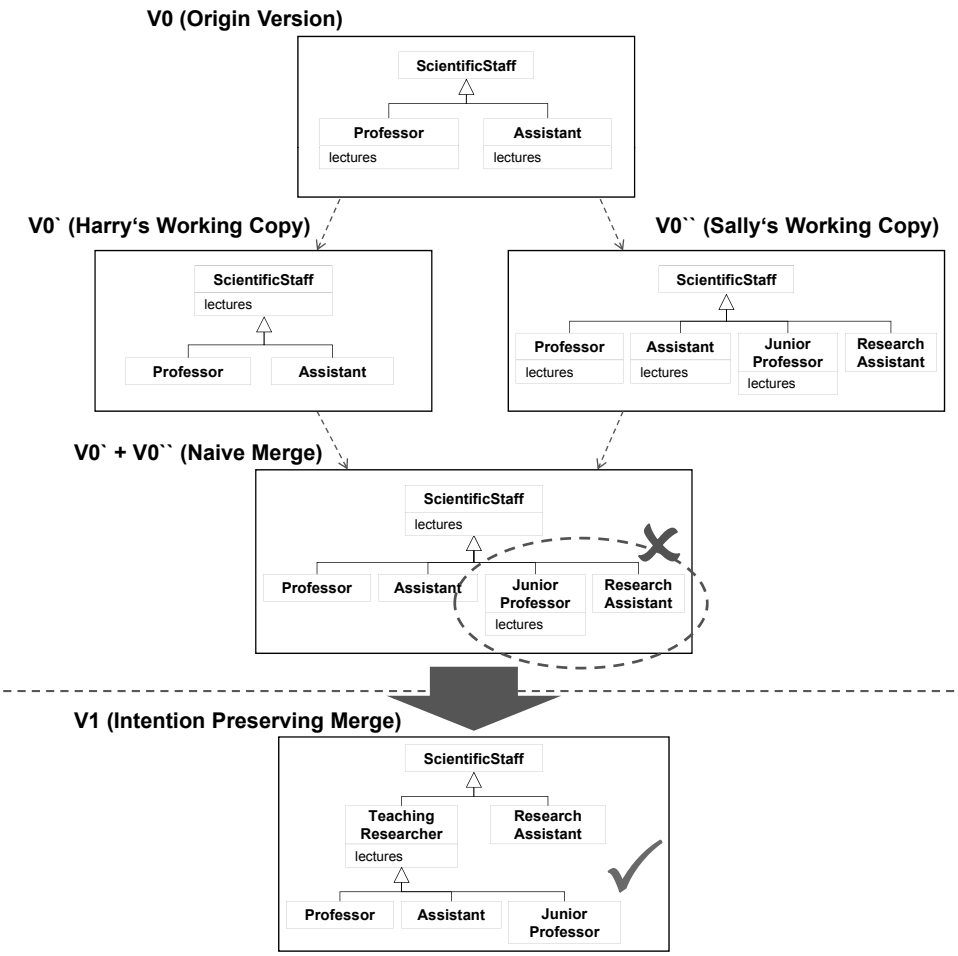


Fig. 1: Conflict Scenario.

2 Motivating Example

In the following, we introduce a small use case in which two modelers, Harry and Sally, concurrently modify the common origin model. In this scenario a naive merge of both working copies leads to an unintended effect regarding the semantics of the merged model.

Conflict Scenario. Both, Harry and Sally work on the common base model V0 depicted in Figure 1. The model contains the two classes Professor and Assistant which are subclasses of ScientificStaff. Each of them contains a property lectures. Harry decides to perform the refactoring “Pull Up Field” [FBB+99] by shifting the property lectures to the superclass ScientificStaff. His modifications are depicted in version V0’ of Figure 1. In the meanwhile, Sally introduces a new subclass JuniorProfessor which includes the

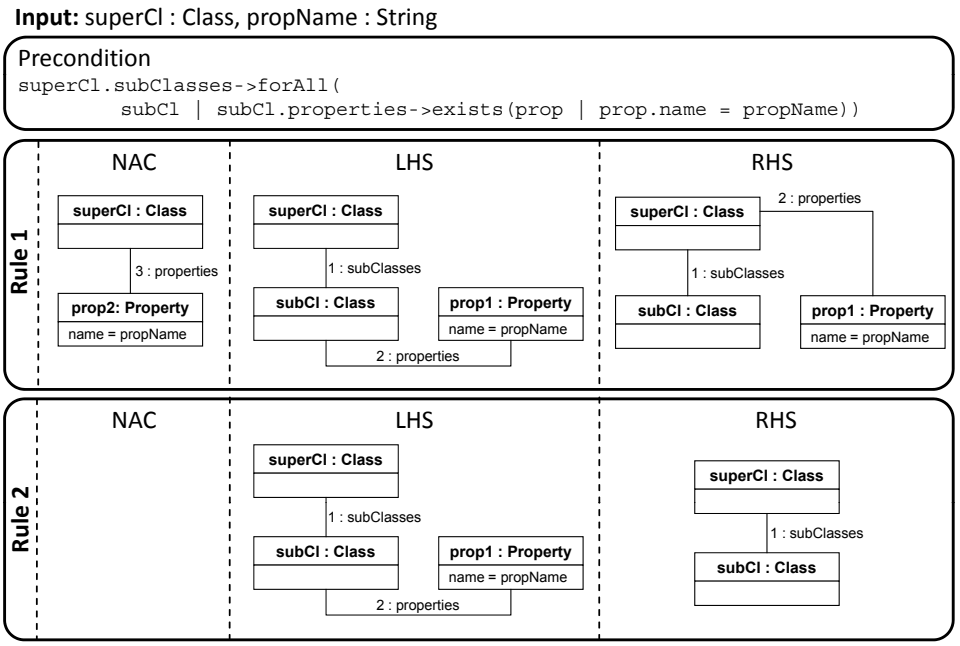


Fig. 2: Graph Transformation Rules for "Pull Up Field".

property lectures like the other existing subclasses. Additionally, she adds a new subclass ResearchAssistant which does not contain the field lectures (cf. V0" in Figure 1) since research assistants (at least in theory) are not holding lectures.

When merging naively there are no conflicts, because no overlapping changes are at hand. The modifications of both modelers are incorporated in the merged version V0' + V0" depicted in Figure 1. However, a closer look at the merged version reveals two problems. The first obvious one is due to the redundantly declared field lectures in class JuniorProfessor which already inherits the aforementioned field from ScientificStaff. Secondly, the class ResearchAssistant now also inherits the field lectures. This has not been intended by Sally who has introduced this class in V0". Thus, the merge has changed the originally intended semantics of the model, which should not happen when refactorings are applied.

Intention Preserving Merge. The aforementioned issues are only detectable if the applied changes are correctly identified. Especially, Harry's refactoring "Pull Up Field", a composite operation, has to be known to the conflict detection component in order to indicate the conflicts.

In AMOR language specific composite operations may be defined by the user to improve change detection. Figure 2 shows such a composite operation represented by a graph transformation rule. The refactoring "Pull Up Field" is defined by depicting required inputs, preconditions, and transformation rules. This refactoring needs two inputs—the superclass

to which the field is shifted and the name of the field to be pulled up. The depicted precondition ensures the preservation of the model's semantics by stipulating all subclasses to contain a property with the specified name. Consequently, the refactoring may only be applied to a superclass superCl if all of its subclasses provide a property with the name propName. Rule 1 is executed if superCl does not yet contain the field to be pulled up. In this rule the property, currently contained by a subclass, is shifted to the superclass. Due to the negative application condition this rule is executed only once. Henceforward, Rule 2 removes the shifted field from the subclass as long as there are subclasses containing the field to be pulled up.

With the knowledge about the applied refactoring, the introduction of the redundant field lectures can be avoided by first applying atomic operations like "Add Inheritance" and then replaying the composite operation, "Pull Up Field". The newly introduced elements, i.e., the JuniorProfessor, are then included in the refactoring. However, reconsidering the conflict scenario in Figure 1, Harry's refactoring cannot be directly applied to Sally's working copy. The subclass ResearchAssistant does not contain the field to be pulled up and therefore violates the refactoring's precondition.

The optimal merge, which preserves all intentions of both modelers, consists of the introduction of a new class TeachingStaff containing the property lectures as subclass of ScientificStaff (cf. V1 in Figure 1). The classes Professor, Assistant, and JuniorAssistant are subclasses of the newly created class TeachingStaff and consequently inherit the property lectures. In contrast, the class ResearchAssistant is a direct subclass of ScientificStaff without the property lectures. Hence, instances of ResearchAssistant cannot have lectures. So the intentions of Harry (optimization) and Sally (extension of the model) are adequately captured.

3 The AMOR Model Versioning System

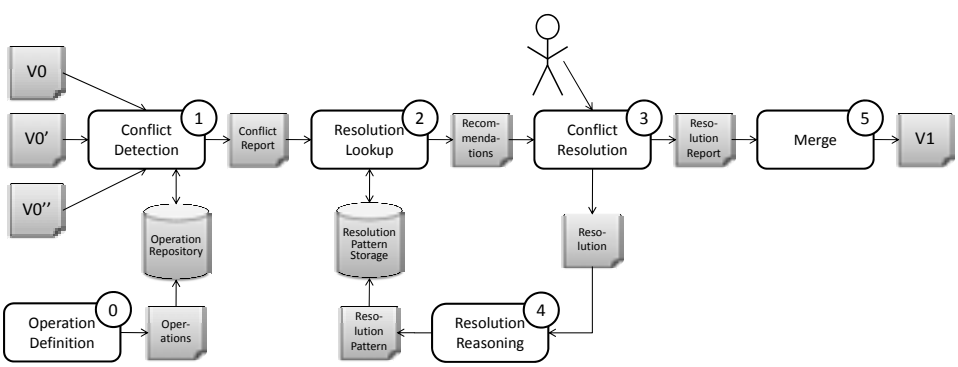


Fig. 3: AMOR Workflow.

In the following, we show how the model versioning system AMOR supports Sally to correctly merge the different versions of the origin model. Therefore, we first give an overview how AMOR extends the workflow found in common versioning systems and then we present how the different components of AMOR operate.

3.1 The AMOR Workflow

In the following, we employ the afore presented example and accompany Sally doing the check-in of her modifications and resolving the afore described conflict with AMOR. The model versioning system AMOR provides not only an enhanced conflict detection component, but also offers user support even to conflicts resulting from the application of composite operations like refactorings. The complete workflow is depicted in Figure 3.

0. *Operation Definition.* The model versioning system AMOR provides modeling language-independent versioning support. The quality of the detection and resolution of conflicts may be considerably improved when language-specific knowledge is incorporated in the merge process [DJ06]. Therefore, AMOR provides an extension point to integrate composite, user-defined operations like refactorings. Once this is done, applications of such refactorings are detectable. This additional information is the basis for a precise conflict detection.

1. *Conflict Detection.* For the comparison and conflict detection the inputs are the origin version V0 and Harry's and Sally's working copies V0' and V0". The conflict detection component detects not only the generic atomic changes like *insert*, *update*, *delete*, but also composite operations stored in the *Operation Repository*. In our example, a conflict between the operations "Pull Up Field" and "Add Inheritance" is reported. If an additional class is added as subclass of ScientificStaff and this new subclass does not contain the attribute lectures, the refactoring "Pull Up Field" is not applicable any more.

2. *Resolution Lookup.* In this step, the *Resolution Recommender* of AMOR checks whether there are solutions for the reported conflict in the *Resolution Pattern Storage*. For the moment, it is assumed that nothing suitable is found. Consequently, the only recommendations which are proposed to Sally are either to apply Harry's refactoring omitting her own changes or to apply her own changes and excluding Harry's refactoring.

3. *Conflict Resolution.* Sally has to decide how to resolve the conflict. She may either resolve the conflict completely manually or choose one of the recommendations made by the *Resolution Recommender*. As mentioned before, there are no useful recommendations and therefore Sally chooses to apply a manual resolution. Since Sally is not sure about Harry's intention, she decides to perform a collaborative resolution where she consolidates Harry. Their solution contains the modifications of both where the new class TeachingStaff is introduced as shown in Figure 1.

4. *Resolution Reasoning.* In the background, the *Resolution Reasoner* analyzes Harry's and Sally's decisions in order to derive a general resolution pattern for the conflict between the "Pull Up Field" and "Add Inheritance" operations. The derived resolution introduces a new intermediate class containing the pulled up property and positions this class at the appropriate place in the inheritance hierarchy. The pattern is stored in the *Resolution Pattern Storage* for the application in similar situations.

5. *Merge.* Finally, all previously chosen resolution recommendations are applied and the resulting model is saved into the repository as a new version.

3.2 The Components of AMOR

Operation Definition. In step 1 the merge process comprises the detection of conflicts concurrently performed by the modelers. Beside atomic changes like *insert*, *update*, and *delete*, modelers may also have performed composite operations like refactorings. Detecting and regarding these composite operations enhances the preservation of both modelers' intentions, as reported in [DJ06]. Of course, composite operations are always specific to a certain modeling language. The AMOR conflict detection component may be enhanced with user-defined composite operation specifications, which describe composite operations in terms of a set of atomic operations and necessary pre- and postconditions. AMOR provides a tool called *Operation Recorder* to easily specify composite operations for specific languages by example [BLS+09b]. These *Operation Specifications* are then either interpreted by the AMOR system or used to derive executable representations like graph transformations.

Conflict Detection. Once an operation specification is created and included in the *Operation Repository*, the *Change Detector* is able to identify applications of the respective composite operation. The detection mechanism is implemented by searching for the operation pattern contained in the *Operation Specification*. If the pattern is found and the model elements referenced by the matching operations fulfill the pre- and postconditions, an application of the composite operation is at hand. This detection allows a more compact representation of the difference and conflict reports by folding atomic operations which belong to a composite operation.

Based on the applied changes, conflicts are easily detected if the same element is modified in different versions of a model. Conflicts resulting from changes of different elements are much harder to detect. Especially, if composite operations are involved, standard versioning systems do not reveal conflicts and merge problems related to the application of the composite operation. To overcome this drawback, AMOR creates a tentative merge by first applying all non-interfering atomic changes and subsequently by replaying the executed composite operations to the common base version. Consequently, added or changed model elements are enclosed in the reapplied composite operation. On the one hand, this maximizes the combination of the original modelers' intentions and, on the other hand, reveals inconsistencies concerning the compatibility of operations. For instance, such inconsistencies occur if a composite operation cannot anymore be executed to the model after all atomic changes are applied. This is accomplished by testing the preconditions of the respective composite operation in the tentatively merged model.

Resolution Lookup. Recent VCSs indicate where conflicts interfere the merge process, but they hardly provide any resolution support to the user. AMOR provides the *Resolution Recommender* which offers resolution recommendations to the modeler who is responsible to perform the change correctly. These recommendations are stored in the *Resolution*

Pattern Storage. A resolution recommendation is a pair containing a conflict description and an executable operation pattern. The *Resolution Recommender* matches the conflicts obtained from the conflict report against the conflict descriptions in the *Resolution Pattern Storage*. The matching resolution recommendations are presented to the user ordered according to their relevance. The *Resolution Recommender* identifies not only perfectly matching resolution recommendations, but also proposes resolution recommendations for similar conflict situations by abstracting the resolution pattern (see Section 4).

Conflict Resolution. AMOR provides different mechanisms to support the conflict resolution which are described in the following.

- *Semi-automatic vs. Manual Conflict Resolution.* In semi-automatic conflict resolution the user selects one of the suggested recommendations offered by the *Resolution Recommender*. The resolution rule of the resolution recommendation is then applied on the maximally merged version. It is executed completely automatically as described in [BLS+09b]. In certain cases user input is required, e.g., if the name of an element has to be introduced. If none of the given suggestions fit the user's requirement, then it is still possible to resolve the conflict manually either alone or in collaboration with other modelers. This process is described in the next paragraph.
- *Collaborative vs. Single User Conflict Resolution.* When no adequate recommendations are found, then semi-automatic resolution is not possible. If one single modeler decides how to merge, the danger is inherently high that the merged version does not reflect the intentions of all involved modelers. Therefore, AMOR offers on the one hand a validation of the merged version and on the other hand an opportunity to resolve conflicts in a collaborative way. In this context, collaboration refers to communicating the intentions behind the changes and trying to combine these intentions in a new version. Thus, AMOR provides an extension called *Collaborative Conflict Resolver* [BLS+09a] to overcome the before mentioned issues by orchestrating the modelers when resolving conflicts. The *Collaborative Conflict Resolver* offers the modelers a communication platform to exchange the intentions behind their conflicting changes. The modelers may manually create a merged version together by partly remodeling the scenario to reach a model of high quality. Afterwards, both modelers have to accept the new version before a commit to the central model repository is possible. This ensures an approved and consolidated version to be checked in. To sum up, the *Collaborative Conflict Resolver* allows to distribute the responsibility in this critical and error-prone merge phase. Solving conflicts in collaboration ensures the preservation of all intentions of the modelers and, overall, increases the acceptance of the merged version and the whole development process.

Resolution Reasoning. The mission of the *Resolution Reasoner* is twofold. On the one hand, the *Resolution Reasoner* tries to derive resolution patterns if the solution is obtained manually, which can be later used for automatic resolution. This is done by reusing mechanisms of the *Operation Recorder*. On the other hand, the *Resolution Reasoner* tracks the application of already existing resolution patterns and persists metadata for ranking purposes.

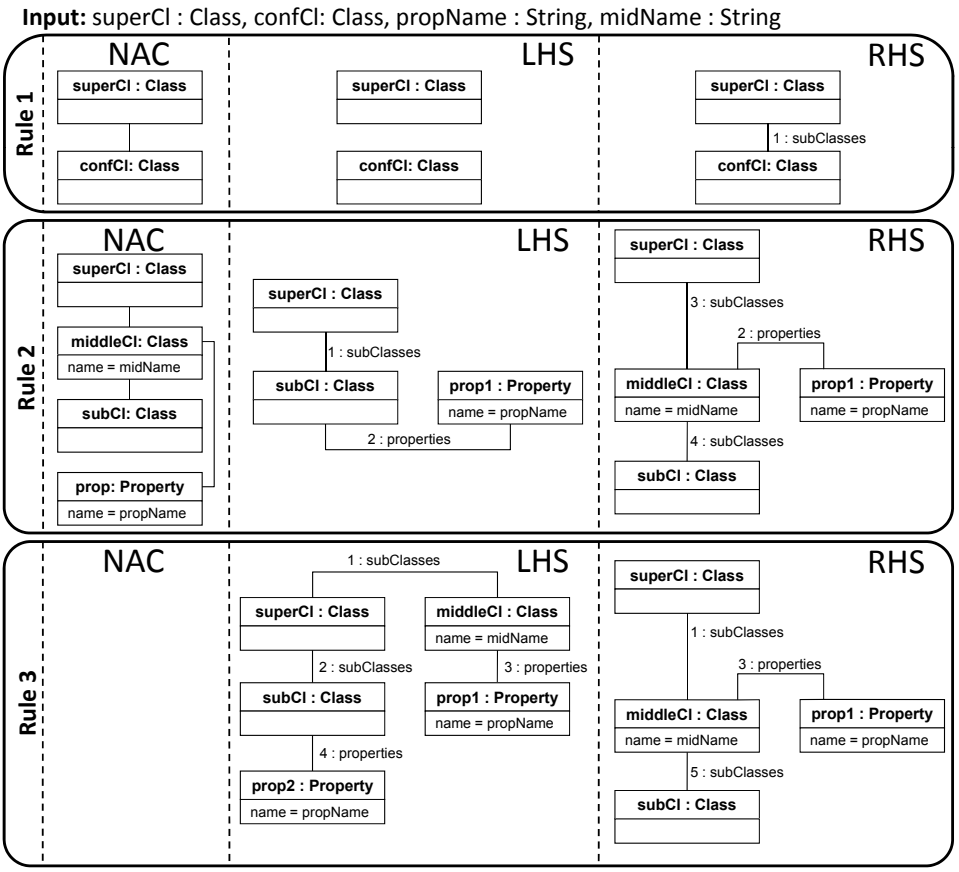


Fig. 4: Derived Resolution Pattern.

Concerning the example conflict, a graph transformation consisting of three rules is derived, which is illustrated in Figure 4. This pattern takes the superclass (cf. superCl), the class which violates the precondition of the refactoring (cf. confCl), and the name of the field to be shifted (cf. propName) as input. Moreover, the user has to provide the name of the newly introduced intermediate class (cf. midName). As conflicting operations are not automatically merged, the tentative merge neither includes the inheritance relationship between the conflicting class and the superclass nor the refactoring "Pull Up Field". Hence, both operations have to be captured by the resolution pattern. Rule 1 matches only once and adds the inheritance relationship, i.e. the pending operation of Sally. Rule 2 and Rule 3 are used to reflect the refactoring of Harry. Rule 2 introduces the intermediate class with the given name and links the property to be pulled up therewith. Due to the negative application condition matching the produced structure, this rule is either executed once. As long as subclasses comprising the given property exist, the property is removed from these subclasses by Rule 3. This pattern is stored in the *Resolution Pattern Storage* for reusing it in future scenarios.

Merge. The *Conflict Report* is used to construct the consolidated merged version. If the user has decided to apply suggested conflict resolution patterns, then they are automatically executed. Finally, the merged version is stored in the repository.

4 Derivation of New Resolution Patterns

A few days later, Sally gets in trouble again during the check-in. In order to put her new model version into the repository, she has to pass again the AMOR check-in process. As in the previous example Harry again has been faster with applying his changes and has already updated the head version of the repository. Hence, Sally's working copy has to be merged. Also this time she has a conflict with Harry's work. Fortunately, AMOR is now able to recommend a resolution strategy based on the previous example found in the *Resolution Pattern Storage*. In the following, we present a new versioning example and show how AMOR is able to automatically adapt and apply the afore derived resolution pattern for the new conflict.

4.1 Motivating Example 2.0

Again, Harry and Sally work on the common base model V0 depicted in Figure 5(a). The model contains an inheritance hierarchy of birds including the superclass Bird and the two subclasses Hawk and Duck. This time, both of the subclasses own an operation named getFlightSpeed(). In order to optimize the model's structure, Harry applies the refactoring "Pull Up Method" [FBB+99] to shift the common operation getFlightSpeed() to the superclass. In the meantime, Sally expands the model by inserting further subclasses Robin and Penguin. While the class Robin also has the operation getFlightSpeed(), the class Penguin has not. This is due to the fact that penguins cannot fly.

4.2 Metamodel-based Resolution Lookup

When checking-in her new version, AMOR supports Sally as follows.

1. *Conflict Detection.* When Sally checks in her working copy, the merge cannot be performed automatically. A conflict is reported because Harry has applied the refactoring "Pull Up Method" in parallel to Sally's modifications. In Sally's version, the preconditions for the application of the refactoring are not satisfied due to the absence of the operation which should be shifted to the superclass in the class Penguin. A *Conflict Report* is generated (cf. 5(b)).
2. *Resolution Lookup.* In this step, it is checked, whether suitable resolution patterns have already been defined. Therefore, the *Resolution Pattern Storage* is searched for candidate resolution patterns. If a perfect match is found in the *Resolution Pattern Repository*, only the associated resolution pattern has to be applied for resolving the conflict. However, if only a partial match is found, special reasoning

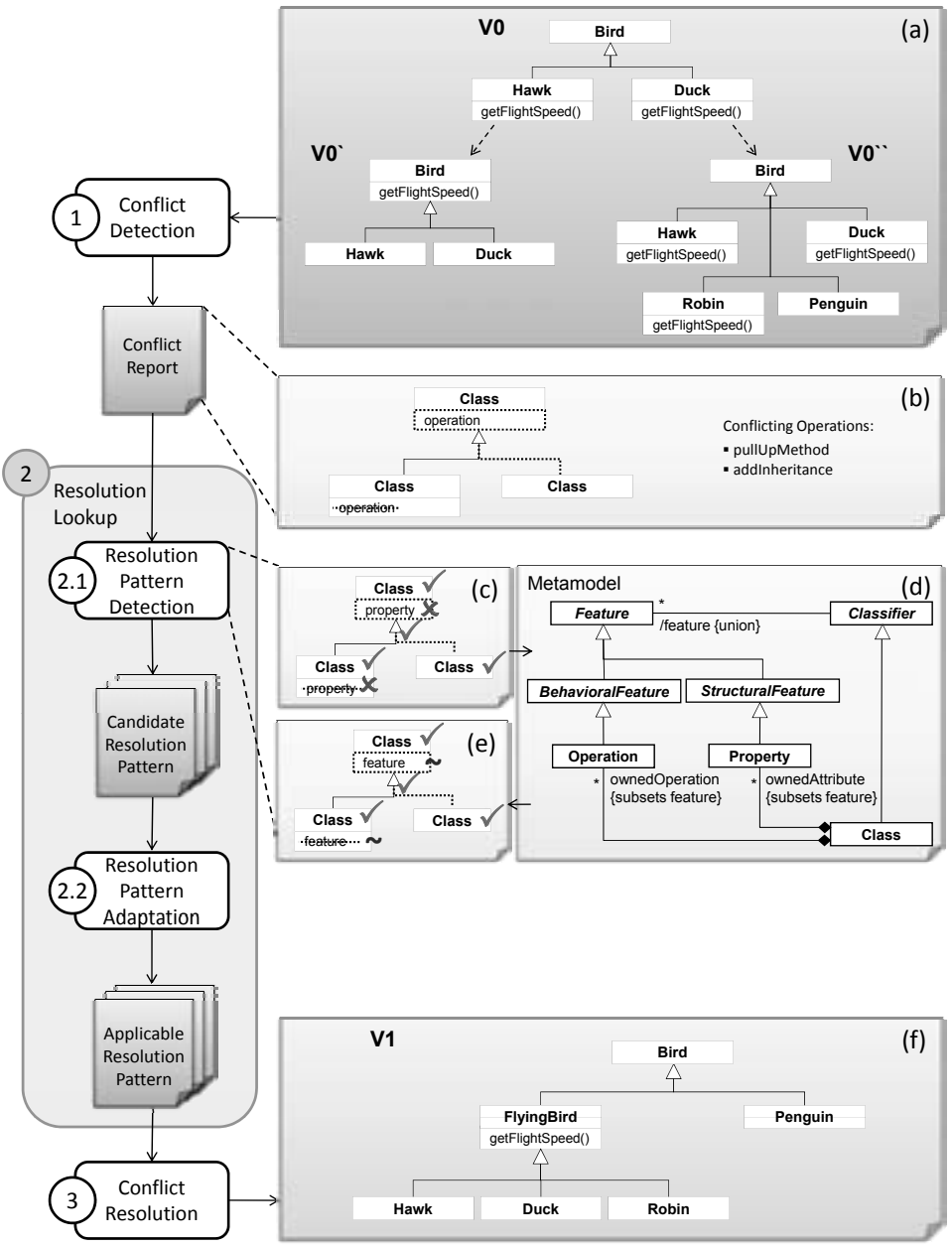


Fig. 5: Metamodel-based Resolution Lookup By-Example.

techniques have to be applied in order to decide if the partial matching conflict is generalizable to match the reported conflict. The generalization of partial matching conflicts requires an adaptation of the associated resolution pattern before it is ap-

plicable on the reported conflict which is done in the second phase called *Resolution Pattern Adaptation*.

2.1. *Resolution Pattern Detection.* With the *Conflict Report* at hand, the first phase in semi-automatic conflict resolution is the lookup of suitable conflict descriptions. A query, taking the *Conflict Report* as parameter, is sent to the *Resolution Pattern Storage* which not only returns resolution patterns for the actual exact conflict. It also returns resolution patterns which had been applied to similar scenarios and which might be adaptable to the actual one. This is done by first comparing the untyped graph structure of the model of the current conflict report with models already persisted in the *Resolution Pattern Storage*. Once matching graph structures are found, the type information is analyzed. For this, the metamodel is used as knowledge base for finding similarities of model elements.
For the new conflict situation, the conflict of the previous example (cf. Figure 1) is found as a similar conflict. The lookup algorithm detects the similar structure of the conflicting part of the model. By comparing the type information, a positive match is found for the classes and the inheritance relationship (cf. Figure 5(c)). However, instead of operations the classes contain properties. Instead of rejecting the found pattern immediately, a similarity analysis of the mismatching elements is performed. With the help of the metamodel (cf. Figure 5(d)) inheritance relationships between those elements are identified. By reflecting the metamodel, the algorithm finds out that Property and Operation both are subclasses of the Feature class and are both associated to Class which is relevant for both refactorings "Pull Up Field" and "Pull Up Method". Furthermore, the associations ownedAttribute and ownedOperation are subsets of feature. Hence, they may be considered as similar (cf. Figure 5(e)).
This analysis is accomplished for all matching graph structures found in the *Resolution Pattern Storage*. The most similar conflicts are returned ordered by similarity metrics and statistics of their previous usage.

2.2. *Resolution Pattern Adaptation.* For each candidate resolution pattern found in the previous step, a higher-order transformation [TJF+09] is applied in order to modify the corresponding resolution rule to fit the necessary model elements in the actual conflict situation. The higher-order transformation may replace the type information like in the example above. In our case, this means that the resolution pattern illustrated in Figure 4 is rewritten as followes. First, the types are substituted. Second, the links are adapted to match the new types. Third, links and attributes which are only existing for Property and not for Operation are eliminated. However, the last step is not necessary for our example.

3. *Conflict Resolution.* Regardless whether the user decides for a single or a collaborative conflict resolution, a tentative merge and a list of pending operations, which could not be applied so far, are presented to the user. In cases where similar conflicts were found in the *Resolution Pattern Storage*, the list of the adapted resolution patterns is also presented to the user. The user may now select one of the resolution patterns which is then automatically applied on the actual conflict situation. A preview of the resolution is displayed and manual adjustments are possible. In our example, the adapted resolution pattern is able to produce a new version which includes the

intentions of both modelers in a consistent way (cf. Figure 5(f)). Sally only has to provide a name for the newly introduced middle class.

According to step 4 of the AMOR workflow (cf. Figure 3), the new resolution pattern is derived by the *Resolution Reasoner* and persisted in the *Resolution Pattern Storage* in order to be recommended in future conflict situations. Finally in the Merge, the conflict is resolved and committed to the repository.

5 Related Work

In the following, we give a short overview of work related to the AMOR model versioning system. Therefore, we consider the three issues (1) model versioning in general, (2) refactoring-aware versioning, and (3) conflict resolution support.

Model Versioning Systems. In the last decades a lot of research approaches in the domain of software versioning have been published which are profoundly outlined in [CW98] and [Men02]. Most of them mainly focus on versioning of source code as they deal with software artifacts in a textual manner. Still, dedicated approaches aiming at the versioning of software models exist. For example, Odyssey-VCS [MCPW08] supports the versioning of UML models. This system performs the conflict detection at a very fine-grained level, hence it is able to merge modifications concerning different model elements or even different attributes of one model element. Odyssey-VCS neither considers composite operations nor does it provide user support in conflict situations. EMF Compare [BP08] is an Eclipse plug-in which is able to match, to compare, and to merge Ecore-based models. In combination with a model repository a model versioning system could be realized. EMF Compare is intended to match any kind of model artifact, hence no language-specific operations and conflicts can be handled. CoObRA [SZN04] is integrated in the Fujaba tool suite and logs the changes performed on an artifact, hence it is tightly-coupled to a modeling tool. The modifications performed by the modeler who did the later commit are replayed on the updated version of the repository. Conflicts occur if an operation may not be applied due to a violated precondition. Unicase [Kög08], an Eclipse-based CASE-tool, integrates different model viewpoints. The Unicase client allows viewing and editing models in a textual, tabular, and diagramming visualization. The models are stored in a repository and can be versioned. The provided three-way merge technique makes use of editing operations, which are obtained from the Unicase client. SMoVer [Alt08] is a VCS for EMF-based model artifacts which is able to detect semantic conflicts. Therefore the parallel evolved model versions are transformed to a semantic view which emphasizes a certain semantic aspect. For each modeling language and each semantic view the transformation has to be manually specified.

Overall, these approaches focus on specific aspects of conflict detection and provide little or no resolution support at all. However, some of these approaches are suitable for the integration in AMOR. A detailed survey on model versioning systems is given in [ASW09].

Refactoring-Aware Versioning Systems. Our approach comprises knowledge on applied refactorings to improve the quality of the merge. Hence, it is related to refactoring-aware versioning systems such as [EA04] and [DMJN08]. Like in these approaches, we replay the applied refactorings after all atomic changes are merged. The check of the refactoring's preconditions before its execution in order to proof its applicability and detect potential refactoring conflicts is also done in [EA04]. In contrast to [EA04] and [DMJN08] we discuss refactoring-aware versioning for models and not code. Beside that, our approach has three main advantages over [EA04] and [DMJN08]. First, we do not restrict the language of the artifacts under version control. In AMOR any language based on Ecore is supported. Second, AMOR allows the user to define custom refactorings by-example and does not only support a fixed range of refactorings. Finally, AMOR does not rely on an editor-specific operation tracking which is done in [EA04] and [DMJN08]. Our approach detects refactorings state-based, i.e., only by analyzing the modified models and their common base version. Consequently, any editor may be applied to change and refactor the models.

Conflict Resolution Support. One of the key challenges in versioning is the resolution of conflicts in an automatic way [Men02]. To the best of our knowledge, automatic conflict resolution has not been achieved so far. As the conflict resolution involves many user decisions in general, total automatic resolution will probably never be possible. Semi-automatic approaches offering resolution suggestions to the user are realized for example in MolhadoRef [DMJN08] for code versioning. PROMPT [NM00], a tool for ontology versioning, uses a manual approach, but supports the user by pointing to conflicts as well as giving some hints how to resolve them.

AMOR provides a recommender component suggesting conflict resolution patterns, which are automatically executed, if the user has selected them. The resolution patterns are learned from conflict resolutions manually done by the users. However, therefore we need solutions of high quality capturing all user intentions. AMOR provides a synchronous and collaborative resolution approach (cf. [BLS+09a]) inspired by the work of Dewan and Hegde [DH07], who consider collaborative merge techniques only for software code but not for models.

6 Conclusion and Future Work

In this paper, we presented the adaptable model versioning system AMOR. AMOR extends a standard optimistic versioning system with respect to three aspects. (1) AMOR provides a conflict detection component which may be enhanced with user-defined composite operations. This additional information allows a more precise conflict detection on the one hand, and a more compact conflict report on the other. (2) AMOR provides a recommender component which offers suggestions how to resolve a previously detected conflict. These suggestions may be either manually defined or they are learned from the situations when the modelers resolve the conflicts by hand. (3) Finally, AMOR provides collaborative conflict resolution features, which allow the implementation of conflict resolution policies. In

order to ensure that the intentions of all modelers are captured in the merged version, it is sometimes necessary that they perform the conflict resolution together.

In future work, we will investigate the impact of different similarity heuristics for the lookup of similar conflicts in the *Resolution Pattern Storage*. These heuristics are of particular importance if legacy modeling tools are used which either have no metamodel or a less object-oriented one. Then it is not possible to make use of inheritance relationships. Also then the Lookup Component should be able to find applicable patterns. This will also involve more sophisticated higher-order transformations in order to adapt the transformations which incorporate the resolution solution in the merged model version.

Furthermore, we will perform an evaluation of the Conflict Detection Component of AMOR by defining multiple refactorings for the *Operation Repository* and running tests based on artificially provoked conflict scenarios. Finally, we will conduct an extensive case study in the context of a real world project.

References

[AKK+08] Kerstin Altmanninger, Gerti Kappel, Angelika Kusel, Werner Retschitzegger, Martina Seidl, Wieland Schwinger, and Manuel Wimmer. AMOR—Towards Adaptable Model Versioning. In *Proceedings of the 1st International Workshop on Model Co-Evolution and Consistency Management @ MoDELS'08*, 2008.

[Alt08] Kerstin Altmanninger. Models in Conflict — Towards a Semantically Enhanced Version Control System for Models. *Models in Software Engineering*, pages 293–304, 2008.

[AP05] Marcus Alanen and Ivan Porres. Version Control of Software Models. *Advances in UML and XML-Based Software Evolution*, 2005.

[ASW09] Kerstin Altmanninger, Martina Seidl, and Manuel Wimmer. A Survey on Model Versioning Approaches. *International Journal of Web Information Systems*, 5(3), 2009.

[BLS+09a] Petra Brosch, Philip Langer, Martina Seidl, Konrad Wieland, and Manuel Wimmer. We Can Work It Out: Collaborative Conflict Resolution in Model Versioning. In *Proceedings of the 11th European Conference on Computer Supported Cooperative Work, ECSCW'09*, pages 207–214, 2009.

[BLS+09b] Petra Brosch, Philip Langer, Martina Seidl, Konrad Wieland, Manuel Wimmer, Gerti Kappel, Werner Retschitzegger, and Wieland Schwinger. An Example is Worth a Thousand Words: Composite Operation Modeling By-Example. In *Proceedings of the 12th International Conference on Model Driven Engineering Languages and Systems, MoDELS'09*, pages 271–285. Springer, 2009.

[BP08] C. Brun and A. Pierantonio. Model Differences in the Eclipse Modeling Framework. *UPGRADE, The European Journal for the Informatics Professional*, 2008.

[CRP08] Antonio Cicchetti, Davide Ruscio, and Alfonso Pierantonio. Managing Model Conflicts in Distributed Development. In *Proceedings of the 11th International Conference on Model Driven Engineering Languages and Systems, MoDELS'08*. Springer, 2008.

[CW98] Reidar Conradi and Bernhard Westfechtel. Version Models for Software Configuration Management. *ACM Computing Surveys*, 30(2):232, 1998.

[DH07] Prasun Dewan and Rajesh Hegde. Semi-Synchronous Conflict Detection and Resolution in Asynchronous Software Development. In *Proceedings of the 10th European Conference on Computer-Supported Cooperative Work, ECSCW'07*. Springer, 2007.

[DJ06] Danny Dig and Ralph Johnson. How Do APIs Evolve? A Story of Refactoring. *Journal of Software Maintenance and Evolution: Research and Practice*, 18(2):83–107, 2006.

[DMJN08] Danny Dig, Kashif Manzoor, Ralph E. Johnson, and Tien N. Nguyen. Effective Software Merging in the Presence of Object-Oriented Refactorings. *IEEE Transactions on Software Engineering*, 34(3):321–335, 2008.

[EA04] Torbjörn Ekman and Ulf Asklund. Refactoring-Aware Versioning in Eclipse. *Electronic Notes in Theoretical Computer Science*, 107:57–69, 2004.

[FBB+99] Martin Fowler, Kent Beck, John Brant, William Opdyke, and Don Roberts. *Refactoring: Improving the Design of Existing Code*. Addison-Wesley, 1999.

[KGE09] Jochen Malte Küster, Christian Gerth, and Gregor Engels. Dependent and Conflicting Change Operations of Process Models. In *ECMDA-FA*, pages 158–173, 2009.

[Kög08] Maximilian Kögel. Towards Software Configuration Management for Unified Models. In *Proceedings of the 2nd International Workshop on Comparison and Versioning of Software Models @ ICSE'08*. ACM, 2008.

[MCPW08] Leonardo Murta, Chessman Corrêa, Joao Gustavo Prudêncio, and Cláudia Werner. Towards Odyssey-VCS 2: Improvements over a UML-based Version Control System. In *Proceedings of the 2nd International Workshop on Comparison and Versioning of Software Models @ ICSE'08*. ACM, 2008.

[Men02] Tom Mens. A State-of-the-Art Survey on Software Merging. *IEEE Transactions on Software Engineering*, 28(5):449–462, 2002.

[NM00] Natalya Fridman Noy and Mark A. Musen. PROMPT: Algorithm and Tool for Automated Ontology Merging and Alignment. In *Proceedings of the 17th National Conference on Artificial Intelligence and 12th Conference on Innovative Applications of Artificial Intelligence*, pages 450–455. AAAI Press / The MIT Press, 2000.

[OS06] Takafumi Oda and Motoshi Saeki. Meta-Modeling Based Version Control System for Software Diagrams. *IEICE Transactions on Information and Systems*, E89-D(4):1390–1402, 2006.

[OWK03] Dirk Ohst, Michael Welle, and Udo Kelter. Differences Between Versions of UML Diagrams. *ACM SIGSOFT Software Engineering Notes*, 28(5):227–236, 2003.

[SZN04] Christian Schneider, Albert Zündorf, and Jörg Niere. CoObRA - A Small Step for Development Tools to Collaborative Environments. In *Proceedings of the Workshop on Directions in Software Engineering Environments @ ICSE'04*, 2004.

[TJF+09] M. Tisi, F. Jouault, P. Fraternali, S. Ceri, and J. Bézivin. On the Use of Higher-Order Model Transformations. In *Proceedings of the 5th European Conference on Model Driven Architecture-Foundations and Applications*, pages 18—33. Springer, 2009.

Strukturbezogener Vergleich von Modellversionen mit graphbasierten Optimierungsalgorithmen

Sabrina Uhrig[1] und Bernhard Westfechtel[2]

Abstract: Der Einsatz von modellgetriebener Entwicklung in der industriellen Praxis setzt insbesondere voraus, dass die Versionskontrolle für Modelle adäquat unterstützt wird. In diesem Zusammenhang spielen Algorithmen zum Vergleich von Modellversionen eine zentrale Rolle. Die bisher entwickelten Algorithmen sind Heuristiken, die teilweise eindeutige Objektbezeichner voraussetzen. In diesem Aufsatz beschreiben wir ein von eindeutigen Objektbezeichnern unabhängiges, strukturbezogenes Verfahren zum Vergleich von Modellversionen. Der Vergleich wird auf ein Optimierungsproblem abgebildet, das mit einem graphbasierten Algorithmus gelöst wird. Der Algorithmus ist als Bestandteil eines EMF-basierten Rahmenwerks implementiert, das die Integration und Evaluation alternativer Vergleichsalgorithmen unterstützt.

1 Einleitung und Abgrenzung zu verwandten Arbeiten

Modellgetriebene Softwareentwicklung zielt darauf ab, den Aufwand zur Erstellung komplexer Softwaresysteme mit Hilfe von ausführbaren Modellen zu reduzieren, die auf einer höheren Abstraktionsebene liegen als Programmcode. Die Anwendung modellgetriebener Softwareentwicklung in der industriellen Praxis wird u.a. dadurch gehemmt, dass die Versionskontrolle für Modelle noch nicht ausreichend unterstützt wird. Defizite bestehen insbesondere hinsichtlich der Verfahren zum Vergleich und zum Mischen von Modellversionen [FW07, BE09].

In diesem Aufsatz beschränken wir uns auf den *Vergleich* von Modellversionen. Traditionelle *textbasierte Verfahren*, wie sie seit langer Zeit in Systemen zur Softwarekonfigurationsverwaltung eingesetzt werden [HS77], haben sich als ungeeignet erwiesen, da sie zeilen- und nicht strukturbasiert arbeiten und ein Vergleich auf der Ebene von Textrepräsentationen, die als Speicherformat dienen, keine sinnvollen Resultate auf konzeptioneller Ebene liefert. Dies hat die Entwicklung von *strukturbezogenen Verfahren* motiviert, die jedoch bisher unter folgenden Beschränkungen leiden:

Viele Verfahren setzen *eindeutige Objektbezeichner* voraus, um “gleiche” Objekte miteinander zu identifizieren [MGH05, OWK03, RW98, AP03]. Dies erleichtert den Vergleich erheblich, da nicht mehr nach Korrespondenzen gesucht werden muss. Auf eindeutigen Objektbezeichnern basierende Verfahren haben jedoch eine Reihe gravierender

[1] Angewandte Informatik I, Universität Bayreuth, 95440 Bayreuth, sabrina.uhrig@uni-bayreuth.de
[2] Angewandte Informatik I, Universität Bayreuth, 95440 Bayreuth, bernhard.westfechtel@uni-bayreuth.de

Nachteile. Sie sind nur innerhalb eines Werkzeugs einsetzbar, das diese Bezeichner vergibt. Beim Austausch von Modellen über Werkzeuggrenzen hinweg können die Objektbezeichner jedoch verloren gehen. Weiterhin hängt das Ergebnis des Vergleichs von der Ediergeschichte ab. Schließlich liefert selbst innerhalb eines Werkzeugs ein Vergleich keine sinnvollen Ergebnisse, wenn die Versionen unabhängig voneinander entstanden sind. Einige Verfahren verwenden zwar keine eindeutigen Objektbezeichner, identifizieren jedoch Objekte über vom Modellierer vergebene *Namen*. Namen werden als *Orientierungspunkte* verwendet, die den weiteren Vergleich steuern [XS05, BP08, KWN05]. Werden die Namen geändert, so schlägt die Identifikation fehl.

Alle bekannten Verfahren sind *Heuristiken* und verfolgen dabei unterschiedliche Strategien: Bei SiDiff [KWN05] sowie UMLDiff [XS05] erfolgt das Matching auf der Basis lokaler Entscheidungen mit Hilfe heuristischer Ähnlichkeitsfunktionen. Die wesentlichen Unterschiede bestehen in der Abarbeitungsreihenfolge und im verwendeten Datenmodell. Der in SiDiff verwendete Baum aus Kompositionsbeziehungen erweitert durch Querverweise wird zunächst bottom-up durchlaufen, wobei die paarweisen Ähnlichkeitswerte für die jeweiligen Elemente eines Typs ermittelt werden. Können Elemente (typischerweise über ihre Namensattribute) eindeutig zugeordnet werden, werden die Ähnlichkeitswerte noch nicht zugeordneter Söhne angepasst, woraus sich weitere Zuordnungen ergeben können. UMLDiff hingegen verfolgt einen Top-Down-Ansatz und arbeitet auf aus Quellcode generierten UML-Klassendiagrammen, wodurch mehr Information in die Ähnlichkeitsberechnung einfließen kann. EMFCompare [BP08] ist aufgrund der Top-Down-Durchlaufstrategie dem UMLDiff-Verfahren nach eigener Auskunft am ähnlichsten, hat diesen Ansatz jedoch auf EMF-Modelle verallgemeinert. Einen völlig anderen Ansatz verfolgt Similarity-Flooding [MGMR02]. In diesem Matching-Verfahren auf Graphen mit Kantenmarkierungen werden die Ähnlichkeitswerte, ausgehend von den Ähnlichkeiten der Namen, über eine Fixpunktiteration an die Nachbarelemente propagiert. Die Ähnlichkeitswerte sind auch hier heuristisch, werden aber global berechnet, die korrespondierenden Elemente werden nach Abschluss der Berechnung über verschiedene Metriken ausgewählt. Das Verfahren wurde unseres Wissens nicht auf Klassendiagramme übertragen.

Alle diese Verfahren basieren nicht auf einem mathematisch definierten Optimierungskriterium, so dass sich über die Optimalität der berechneten Lösung keine Aussage treffen lässt. Dies erschwert die Bewertung der Verfahren erheblich, zumal keine allgemein akzeptierten Benchmarks zur Verfügung stehen. In diesem Aufsatz beschreiben wir ein strukturbezogenes Verfahren zum Vergleich von Modellversionen [Uhr08], das sich durch folgende Eigenschaften auszeichnet:

Unabhängigkeit von eindeutigen Objektbezeichnern Es werden keine eindeutigen Objektbezeichner vorausgesetzt. Das Ergebnis des Vergleichs hängt nicht von werkzeugspezifischen Zusatzinformationen, sondern nur von den logischen Inhalten der zu vergleichenden Modelle ab.

Definiertes Optimalitätskriterium Das Verfahren basiert auf einem *Kostenmodell*, das Edieroperationen auf Modellen und daraus gebildeten *Edierskripten* Kosten zuordnet. Gesucht wird ein Edierskript mit minimalen Kosten, das das Ausgangsmodell in das Zielmodell des Vergleichs überführt.

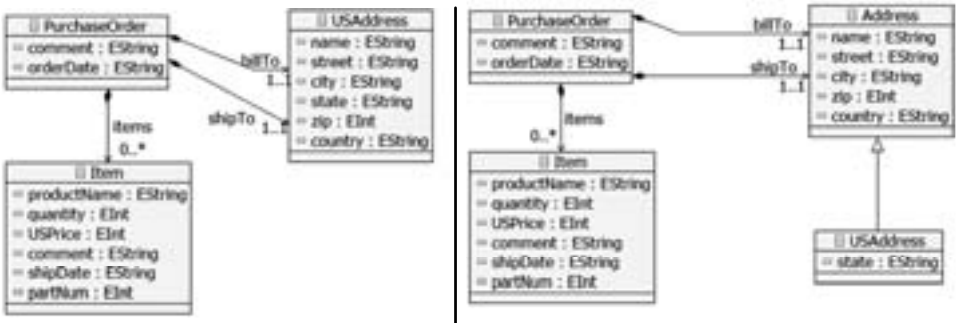


Abb. 1: Beispiel: Vergleich von Klassendiagrammen

Geringe Sensitivität gegenüber Namensänderungen Beim strukturellen Vergleich werden Namen als gewöhnliche Attribute behandelt (nicht als Orientierungspunkte). Bei hinreichend hoher struktureller Übereinstimmung werden auch unterschiedlich benannte Modellelemente miteinander identifiziert.

Verwendung von Optimierungsalgorithmen aus der Graphentheorie Der Vergleich von Modellversionen wird in ein Optimierungsproblem transferiert, das mit bekannten Algorithmen aus der Graphentheorie gelöst werden kann.

Implementierung in einem EMF-basierten Rahmenwerk Das Verfahren ist in einem EMF-basierten Rahmenwerk implementiert, das die Integration alternativer Vergleichsalgorithmen unterstützt und damit das Experimentieren mit und das Evaluieren von Algorithmen zum Vergleich von Modellversionen ermöglicht.

2 Motivation

Wie bereits erwähnt, führt das von uns entwickelte Verfahren einen strukturellen Vergleich durch, durch den Modellelemente auch dann miteinander identifiziert werden können, wenn sie unterschiedliche Namen haben (und zwar ohne Verwendung eindeutiger Objektbezeichner). In diesem Abschnitt betrachten wir nun zwei Beispiele, die den Vorteil dieses Ansatzes verdeutlichen. Abbildung 1 zeigt zwei Ecore-Klassendiagramme zur Modellierung von Kaufaufträgen. In Diagramm *A* werden Liefer- und Rechnungsadresse mit der Klasse USAddress modelliert. In Diagramm *B* wurde die Modellierung von Adressen verallgemeinert. Die Klasse Address enthält fast alle Attribute der alten Klasse USAddress, die nun als Unterklasse von Address definiert wird und das für US-Adressen spezifische Attribut state enthält.

Unser Verfahren berechnet eine minimale Distanz zwischen den beiden Diagrammen. Aufgrund ihrer strukturellen Ähnlichkeiten identifiziert es die Klasse USAddress in Diagramm *A* mit der Klasse Address in Diagramm *B* und ermittelt folgende Differenzen:

1. Umbenennen von USAddress in Address

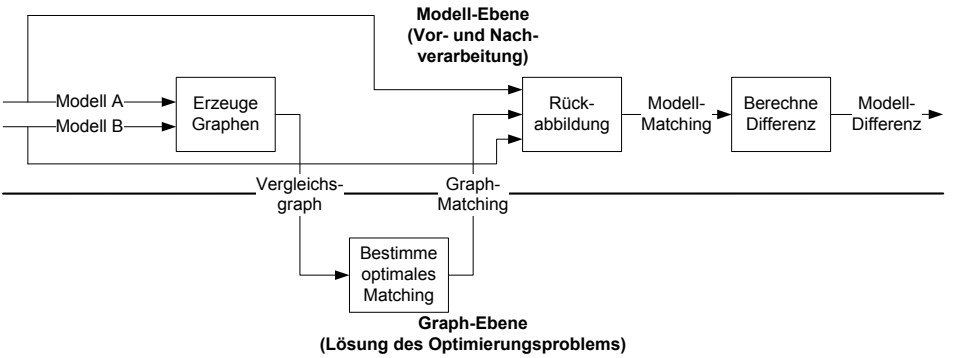


Abb. 2: Graphbasierter Ansatz zum Vergleich von Modellversionen

2. Löschen des Attributs `state`
3. Einfügen der Klasse `USAddress` als Unterklasse von `Address` mit Attribut `state`

Im Gegensatz dazu identifiziert beispielsweise EMF Compare [BP08] Klassen über den Namen und ordnet daher die mit `USAddress` benannten Klassen einander zu. Dies führt zu einer erheblich größeren Modelldifferenz, da fünf Attribute aus der Klasse `USAddress` gelöscht und in die neue Klasse `Address` eingefügt werden müssen.

Um den Effekt von Umbenennungen zu demonstrieren, haben wir in einem weiteren Beispiel ein 8 Klassen umfassendes Diagramm aus dem Englischen ins Französische übersetzt. Unser Verfahren ordnet alle Klassen korrekt einander zu. EMF Compare identifiziert dagegen lediglich drei Klassen, deren Namen zufälligerweise hinreichend ähnlich sind.

3 Ansatz

3.1 Überblick

Abbildung 2 vermittelt einen Überblick über den von uns verfolgten Ansatz zum Vergleich von Modellversionen. Zunächst wird in einem *Vorverarbeitungsschritt* aus den zu vergleichenden Modellen *A* und *B* ein *Vergleichsgraph* konstruiert. Auf den Vergleichsgraphen wird ein *generischer Graphalgorithmus* angewendet, der aus der Graphentheorie bekannt ist [BG61]. Mit Hilfe dieses Algorithmus wird ein *optimales Graph-Matching* berechnet. Die *Rückabbildung* transformiert anschließend das Matching auf der *Graph-Ebene* zurück auf die *Modell-Ebene*. Damit ist bekannt, welche Modellelemente in *A* und *B* miteinander identifiziert werden. Aus den identifizierten Elementen wird schließlich die *Modell-Differenz* berechnet, d.h. es wird ermittelt, welche Elemente aus *A* gelöscht bzw. geändert wurden und welche Elemente in *B* neu eingefügt wurden.

Der generische Graphalgorithmus zur Lösung des Optimierungsproblems hängt nicht von dem *Metamodell* ab, auf dem die zu vergleichenden Modelle basieren. Die auf der Mo-

dellebene angeordneten Schritte des Verfahrens sind dagegen vom Metamodell abhängig. Die dafür benötigten Algorithmen sind für jedes Metamodell spezifisch zu entwickeln.

3.2 Graph-Matching

Es existiert eine Fülle von *graphentheoretischen Algorithmen* für eine große Bandbreite von Problemen. Es stellt sich die Frage, welche Ansätze für den Vergleich von Modellversionen geeignet sind. Der Isomorphiebegriff auf Graphen bzw. die Suche nach einem maximalen gemeinsamen Untergraphen sind für unsere Zwecke zu restriktiv. Für den Vergleich von Modellversionen erweisen sich Ansätze zum *Graph-Matching*[KN05] als besser geeignet. Wir betrachten hier folgende Problemvariante:

Optimierungsproblem 1 (Maximales Graph-Matching mit minimalen Kosten) *Gegeben sei ein bipartiter Graph $G = (V, E)$ mit Knotenmenge $V = V_A \cup V_B$, wobei $V_A \cap V_B = \emptyset$, und Kantenmenge $E \subseteq V_A \times V_B$. Ferner sei $c : E \to \mathbb{R}^+$ eine Kostenfunktion, die jeder Kante nichtnegative reellwertige Kosten zuordnet.*

Gesucht ist ein Matching $M \subseteq E$ mit folgenden Eigenschaften:

1. *Jeder Knoten aus V ist zu höchstens einer Kante aus M inzident.*
2. *M ist maximal mit dieser Eigenschaft, d.h. es gibt kein anderes Matching mit höherer Kardinalität.*
3. *M hat minimale Kosten, d.h. für jedes andere maximale Matching M' gilt $\sum_{e \in M} c(e) \leq \sum_{e \in M'} c(e)$.*

Der Vergleich von Modellen lässt sich auf ein Graph-Matching-Problem abbilden. Dazu konstruiert man für zwei Modelle A und B einen Graphen, dessen Knotenmenge in zwei Partitionen V_A und V_B zerfällt. Zunächst fügt man für die Modellelemente aus A und B entsprechende Knoten in V_A und V_B ein. Für jede mögliche Zuordnung (*Ersetzung*) von Modellelementen fügt man anschließend eine Kante in E ein, deren Kosten sich aus den Operationen errechnet, mit denen man das Quellelement in das Zielelement überführt. *Erzeugungen* von Modellelementen in B werden modelliert, indem man für jedes Modellelement aus B einen ε-Knoten (für ein nicht existierendes Modellelement) in V_A einfügt. Der ε-Knoten wird mit dem Knoten aus V_B durch eine Kante verbunden, der die Kosten für das Erzeugen des Modellelements zugeordnet werden. Schließlich behandelt man das *Löschen* von Modellelementen auf analoge Weise.

3.3 Anwendung auf den Modellvergleich

Ein Algorithmus zum Modellvergleich soll *effizient* sein und ein Ergebnis liefern, das einerseits *optimal* im Sinne von Problem 1 ist und andererseits den Anforderungen des Benutzers entspricht (*Validität*).

Effizienz *Effiziente Algorithmen* zur Lösung des Optimierungsproblems 1 sind aus der Literatur bekannt [Jun08]. Der von uns verwendete Algorithmus hat polynomielle Laufzeit (Abschnitt 3.4).

Optimalität Hinsichtlich der *Optimalität des Ergebnisses* ergibt sich folgendes Problem: Im Optimierungsproblem 1 wird vorausgesetzt, dass sich den Kanten des bipartiten Graphen die Kosten *statisch* zuordnen lassen, d.h. die Kosten werden berechnet, bevor der Algorithmus gestartet wird. Beim Modellvergleich sind jedoch die Kosten insofern *dynamisch*, als sie von der Abbildung des *Kontextes* eines Modellelements abhängen.

Bildet man beispielsweise Klassen aufeinander ab, so hängen die Edierkosten nicht nur von den lokalen Eigenschaften dieser Klassen ab (d.h. von den Attributen und Operationen), sondern auch von der Abbildung der Kontextklassen, die über Assoziationen oder Vererbungsbeziehungen verbunden sind. Diese Beziehungen lassen sich nur dann aufeinander abbilden, wenn ihre Enden miteinander identifiziert werden.

Dieses Problem lässt sich lösen, indem man für das Graph-Matching statt der (unbekannten) *realen Kosten* c_{real} untere Schranken (*Mindestkosten*) c_{lb} verwendet:

$$c_{lb} \leq c_{real} \tag{1}$$

Die Mindestkosten werden bestimmt, indem man zwischen *kontextabhängigen* und *kontextfreien Kosten* unterscheidet und die kontextabhängigen Kosten unter der optimistischen Annahme berechnet, dass der Kontext eines Modellelements "passend" abgebildet wird. Insgesamt führt dies zu folgendem Vorgehen:

1. Das Optimierungsproblem 1 wird für die Mindestkosten gelöst.
2. Für das berechnete Matching werden die realen Kosten und die Mindestkosten berechnet:

$$C^*_{real} = \sum_{e \in M} c_{real}(e) \tag{2}$$

$$C^*_{lb} = \sum_{e \in M} c_{lb}(e) \tag{3}$$

3. Das Optimierungsproblem für die realen Kosten ist gelöst, wenn die realen und die Mindestkosten übereinstimmen:

$$C^*_{real} = C^*_{lb} \tag{4}$$

Gleichung 4 liefert somit ein *hinreichendes Optimalitätskriterium*. Leider ist dieses Kriterium nicht zugleich notwendig: Die optimale Lösung kann höhere Kosten haben als die berechneten Mindestkosten. Falls $C^*_{real} > C^*_{lb}$, ist also nicht bekannt, ob die berechnete Lösung optimal ist. Wir kommen auf dieses Problem in Abschnitt 6 noch einmal zurück.

Validität In unserem Verfahren werden wie in vielen anderen Ansätzen Differenzen mit Hilfe von *Edierskripten* dargestellt. Die Kosten eines Edierskripts werden durch Summation der Kosten der einzelnen Operationen ermittelt. Dabei werden komplexe Operationen (z.B. das Löschen einer Klasse) auf Elementaroperationen (Löschen von Attributen und Methoden) zurückgeführt und — im einfachsten Fall — Elementaroperationen gleich gewichtet (*Einheitskostenmodell*).

Auch bei einer erfolgreichen Lösung des Optimierungsproblems ist nicht sichergestellt, dass das auf diese Weise berechnete Ergebnis die Anforderungen des Benutzers erfüllt (*Validierung*). Um dies zu erreichen, bieten sich folgende Maßnahmen an:

Schwellenwerte für Zuordnungen Ist man an einem Edierskript mit minimalen Kosten interessiert, so ist die Änderung eines Modellelements (z.B. einer Klasse) kostengünstiger, als das Quellelement zu löschen und das Zielelement einzufügen. Sind die Modellelemente jedoch nicht "hinreichend ähnlich", so ist es sinnvoller, eine Zuordnung zu vermeiden, da der Benutzer die Differenz nicht (nur) auf der Ebene von Elementaroperationen bildet. Nicht angemessene Zuordnungen lassen sich z.B. über *Schwellenwerte* für die Ähnlichkeiten ausschließen.

Unterschiedliche Behandlung von Elementaroperationen Es kann sinnvoll sein, von der Gleichgewichtung aller Elementaroperationen) abzuweichen. So spielen etwa *Namensattribute* eine besondere Rolle bei der Zuordnung von Modellelementen. Dies kann beispielsweise berücksichtigt werden, indem man das Ändern von Namensattributen stärker gewichtet. Ferner kann sich eine feingranulare Analyse der Änderungen durch einen Vergleich von Zeichenketten empfehlen, um zu erreichen, dass Modellelemente mit ähnlichen Namen mit größerer Wahrscheinlichkeit identifiziert werden.

3.4 Lösung des Optimierungsproblems

Zur Lösung des Optimierungsproblems 1 reduzieren wir dieses auf die Berechnung eines *maximalen kostenminimalen Flusses* in einem Netzwerkgraphen ([NM02], Kapitel 2). Ein *Netzwerkgraph* ist ein gerichteter Graph $N = (V, E)$, der eine ausgezeichnete Quelle r und eine ausgezeichnete Senke s enthält. Den Kanten $e = (i, j)$ sind als Minimal- und Maximalkapazitäten natürliche Zahlen $\lambda_{ij} \leq \kappa_{ij}$ sowie nichtnegative reellwertige Kosten c_{ij} zugeordnet. Die Flüsse werden mit ϕ_{ij} bezeichnet. Im Folgenden setzen wir $\lambda_{ij} = 0$ voraus (die Minimalkapazität verschwindet bei den von uns konstruierten Netzwerkgraphen). Das Optimierungsproblem lässt sich dann wie folgt formulieren:

Optimierungsproblem 2 (Maximaler kostenminimaler Fluss) *Minimiere für den maximal erreichbaren Fluss f die Kosten*

$$\sum_{(i,j)\in E} c_{ij}\phi_{ij} \tag{5}$$

so, dass der Fluss balanciert ist

$$\sum_{\{j:(i,j)\in E\}} \phi_{ij} - \sum_{\{k:(k,i)\in E\}} \phi_{ki} = \begin{cases} f & \text{if i = r} \\ -f & \text{if i = s} \\ 0 & \forall\, i \in V \backslash \{r,s\} \end{cases} \tag{6}$$

und die Kapazitäten der Flusskanten einhält:

$$0 \leq \phi_{ij} \leq \kappa_{ij},\ (i,j) \in E \tag{7}$$

Es seien n und m die Zahlen der Modellelemente in A und B; o.B.d.A. nehmen wir $n \geq m$ an. Der Netzwerkgraph wird nach folgenden Regeln konstruiert (unter der vereinfachenden Annahme, dass sich alle Modellelemente in A und B aufeinander abbilden lassen):

1. Erzeuge einen Quellknoten r und einen Zielknoten s.
2. Füge für alle Modellelemente aus A und B entsprechende Knoten ein. Die Knotenmengen seien im Folgenden mit V_A und V_B bezeichnet.
3. Füge für jeden Knoten $a \in V_A$ eine Kante (r,a) mit Markierung $(0,1,0)$ ein (Minimalkapazität, Maximalkapazität, Kosten).
4. Füge für jeden Knoten $b \in V_B$ eine Kante (b,s) mit Markierung $(0,1,0)$ ein.
5. Füge für jedes Knotenpaar $(a,b) \in V_A \times V_B$ eine Kante mit Markierung $(0,1,c_{ab})$ ein, die die Ersetzung von a durch b mit Kosten c_{ab} repräsentiert.
6. Füge einen Knoten ε ein und erzeuge Löschkanten $(a,\varepsilon$ mit Markierung $(0,1,c_{a\varepsilon})$ bzw. Erzeugungskanten (ε,b) mit Markierung $(0,1,c_{\varepsilon b})$ für alle $a \in V_A$ und $b \in V_B$. Verbinde den Knoten ε ferner mit dem Zielknoten s mit einer Kante der Markierung $(0,n-m,0)$.

Der Netzwerkgraph ist somit i.W. ein bipartiter Graph, der um einen Quell- und einen Zielknoten ergänzt wird. Ferner enthält er einen Knoten ε, dessen einlaufende Kanten zum Löschen und dessen auslaufende Kanten zum Einfügen von Modellelementen verwendet werden. Ein maximaler Fluss hat die Stärke $|V_A|$. Allen von r ausgehenden Kanten wird ein Fluss der Stärke 1 zugeordnet. Für jedes $a \in V_A$ wird genau eine auslaufende Kante beschickt. Dadurch wird festgelegt, ob a durch ein $b \in V_B$ ersetzt oder gelöscht wird. Die Balancierungsregeln stellen ferner sicher, dass jedes $b \in V_B$ entweder neu erzeugt wird oder ein $a \in V_A$ ersetzt. Als Beispiel zeigt Abbildung 3 den Netzwerkgraphen, der für die Diagramme aus Abbildung 1 konstruiert wird (mit vertauschten Rollen von A und B).

Zur Lösung des Optimierungsproblems haben wir den Algorithmus von *Busacker* und *Gowen* [BG61] implementiert (dargestellt z.B. in [NM02, Jun08]). Die Laufzeitkomplexität liegt in $O(n_V n_E f)$, wobei n_V und n_E für die Zahl der Knoten bzw. Kanten des Netzwerkgraphen sowie f für die Kapazität des berechneten Flusses stehen. Bei Netzwerkgraphen,

die nach obiger Vorschrift konstruiert werden, lässt sich die Komplexität als Funktion von $n_A = |V_A|$ ausdrücken: $|f| = n_A$, $n_V \leq 2n_A + 3$, und $n_E \leq n_A^2 + 4n_A$. Damit ergibt sich insgesamt $O(n_A^4)$ als obere Schranke für die Laufzeitkomplexität.

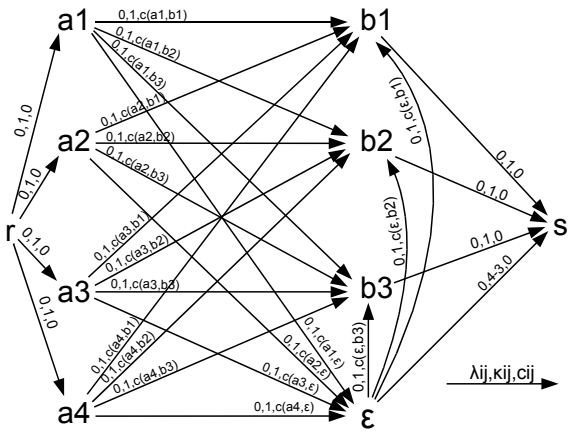


Abb. 3: Netzwerkgraph für den Vergleich der Klassendiagramme aus Abbildung 1

4 Vergleich von Klassendiagrammen

Das beschriebene Verfahren zum Vergleich von Modellversionen wurde für Ecore-Klassendiagramme implementiert. Der Graph eines Klassendiagramms wird dabei als hierarchische Struktur aus Kompositionsbeziehungen mit Querverbindungen betrachtet. Die Attribute, Operationen, Vererbungskanten und Assoziationen einer Klasse bilden jeweils einen Baum, der über die Assoziationen und Vererbungskanten wiederum mit anderen Bäumen verbunden sein kann.

Abbildung 4 zeigt das dabei verwendete Metamodell, eine Teilmenge des Ecore-Metamodells. Um das Problem in seiner Komplexität zu reduzieren, wurde der Typ von Attributen, Rückgabeparametern und Übergabeparametern vereinfacht als String behandelt. Sichtbarkeiten und der Modifizierer *static* sind in diesem Modell nicht erfasst. Ebenso wenig werden derzeit Unterpakete unterstützt. Stattdessen gehen wir davon aus, dass alle Klassen in einem Grundpaket liegen. Im folgenden Abschnitt werden die auf diesem Modell zulässigen Edieroperationen und deren Kosten behandelt.

4.1 2-Ebenen-Ansatz

Um die Kosten für die Zuordnung zweier Klassen zu berechnen, müssen auch die Kosten für die Zuordnungen derer Unterelemente berücksichtigt werden. Dabei wurden folgende Annahmen hinsichtlich der zulässigen Edieroperationen getroffen:

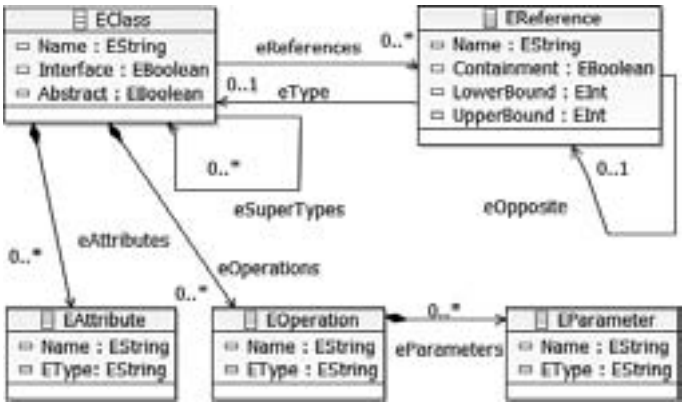


Abb. 4: Implementierter Teil des Ecore-Metamodells

Keine Typänderung Der Typ von Elementen kann nicht geändert werden, d.h. dass ein Attribut nicht in eine Methode umgewandelt werden kann. Eigenschaften jedoch können geändert werden. So kann eine Klasse zum Interface werden, indem das entsprechende *Interface*-Attribut gesetzt wird.

Vollständige Teilbaum-Zuordnung Falls die Klassen *a* und *b* einander zugeordnet werden, werden auch die Unterelemente von Klasse *a* den entsprechenden Unterelementen von Klasse *b* zugeordnet, oder gelöscht. Unterelemente von Klasse *b*, die dabei nicht zugeordnet werden konnten, werden erzeugt.

Mit diesen Annahmen beschränken sich die auf diesem Modell zulässigen Edieroperationen auf die elementaren Operationen Erzeugen und Löschen von Elementen und das Ändern von Eigenschaften. Das Verschieben von Elementen wird nicht als eigenständige Edieroperation betrachtet, sondern wird durch eine Folge von Einfüge- und Löschoperationen erzielt. Auch Assoziationen und Vererbungskanten werden jeweils als gemeinsames Unterelement der beiden beteiligten Klassen betrachtet. Daher kann das Assoziationsende einer Assoziation zwischen den Klassen *a* und *b* nicht von *b* nach *c* verschoben werden. Stattdessen muss die Assoziation zwischen *a* und *b* gelöscht und eine andere Assoziation zwischen *a* und *c* erzeugt werden. Das Gleiche gilt auch für Vererbungskanten.

Jeder zulässigen Edieroperation werden Kosten zugewiesen. Das hier verwendete Kostenmodell enthält nur positive reelle Kosten und ist symmetrisch hinsichtlich inverser Operationen. So verursacht das Löschen eines Elements die gleichen Kosten wie das Einfügen des gleichen Elements. Diese setzen sich aus den Löschkosten des Elements selbst und den Löschkosten aller Unterelemente zusammen. Die Löschkosten eines Elements selbst wiederum sind größer oder gleich der Summe der Kosten, alle Eigenschaften dieses Elements zu ändern. Insgesamt orientieren sich die Kosten an einem *Einheitskostenmodell*, bei dem jede Elementaroperation mit Kosten 1 gewichtet wird. Abweichungen davon ergeben sich nur durch nachträgliche Verfeinerungen, z.B. indem die Edieroperation zum Ändern der Multiplizität einer Assoziation in zwei Operationen zum Ändern der oberen bzw. unteren Schranke aufgeteilt wurde.

Tabelle 1 zeigt einen Ausschnitt aus dem Kostenmodell, anhand dessen nun exemplarisch die Kosten für die Edieroperationen auf Methoden erläutert werden. Falls sich die Namensattribute a und b zweier Methoden unterscheiden, werden die Kosten für die Änderungsoperation mit Hilfe der *längsten gemeinsamen Untersequenz lcs* ([HS77]) berechnet, wobei sich Edierkosten innerhalb des Intervalls $[0,1]$ ergeben (siehe Formel 8).

$$lcs = 1 - \frac{2 * lcs.length}{a.length + b.length} \tag{8}$$

Abweichungen im Rückgabetyp der Methode werden mit Kosten 1 bewertet. Beim Einfügen bzw. Löschen eines Übergabeparameters entstehen Kosten in Höhe von 2 (Addition der Einheitskosten für Name und Typ des Parameters). Wird die komplette Methode gelöscht, entstehen Kosten in Höhe von 2 für das Löschen der Methode selbst und zusätzlich 2 für jeden gelöschten Übergabeparameter dieser Methode.

EOperation	
Änderung des Namensattributs	lcs
Änderung des Rückgabetyps	1
Erzeugen/Löschen einer Methode	2 + Kosten für das Einfügen/Löschen aller Übergabeparameter
EParameter	
Änderung des Namensattributs	lcs
Änderung des Typs	1
Einfügen/Löschen eines Übergabeparameters	2

Tab. 1: Kostenmodell für Methoden

Aus dem 2-Ebenen-Ansatz ergibt sich ein hierarchischer Aufbau von Optimierungsproblemen. Für die Berechnung der Kosten für die Zuordnung zweier Klassen müssen die Kosten für die Zuordnung aller Unterelemente berücksichtigt werden. Dies beinhaltet weitere Zuordnungsprobleme für die Zuordnung der Attribute, Methoden, Assoziationen und Vererbungsbeziehungen, die nach der Übertragung in ein entsprechendes Netzwerk ebenfalls mit dem Busacker-Gowen-Algorithmus gelöst werden können. Erst nachdem die Zuordnungen der Unterelemente berechnet sind, kann auch das Optimierungsproblem auf Klassenebene gelöst werden.

4.2 Kontextabhängige Kosten und deren Abschätzung

Für die Kantenbeschriftungen des Netzwerkgraphen werden statische Kosten benötigt (vgl. Abschnitt 3), d.h. die Kosten der Zuordnung zweier Klassen a und b muss von den Zuordnungen der anderen Klassen entkoppelt sein, da sonst ein kombinatorisches Problem mit exponentiellen Aufwand entstünde. Bei den Attributen und Methoden können die Zuordnungskosten exakt berechnet werden, da es sich in diesen Fällen um lokale Eigenschaften handelt. Für die exakte Berechnung der Kosten für die Zuordnung der Assoziationen

und Vererbungskanten von a und b werden jedoch Informationen über die Kontextklassen benötigt: Da Assoziationen ebenso wie Vererbungskanten in diesem Modell nicht verschoben werden können, dürfen zwei Assoziationen von a und b im Fall der Abbildung von a auf b nur dann identifiziert werden, wenn auch deren Assoziationsenden c und d in der berechneten Lösung identifiziert werden.

Daher arbeiten wir mit abgeschätzten Kosten für die Zuordnung von Assoziationen und Vererbungsbeziehungen, die eine untere Schranke für die tatsächlichen Kosten darstellen. D.h. wir treffen die optimistische Annahme, dass in unserem obigen Beispiel die Klassen c und d einander zugeordnet werden, und berücksichtigen nur eventuelle Kosten für die Namensänderung der Assoziation oder Änderungen der Multiplizitäten. Dieses relaxierte Problem mit den Kostenabschätzungen wird nun mit dem Busacker-Gowen-Algorithmus gelöst. Sobald die Zuordnung der Klassen feststeht, können die vorher abgeschätzten Kosten für die Zuordnung der Assoziationen und Vererbungskanten exakt bestimmt werden. Falls die Klassen c und d aus unserem Beispiel doch nicht miteinander identifiziert werden, liegen die realen Kosten über den abgeschätzten Kosten.

4.3 Schwellenwerte

Wie in Abschnitt 3 bereits erwähnt wurde, kann es sinnvoll sein, auf die Zuordnung der Elemente in Form von Schwellenwerten Einfluss zu nehmen. An dieser Stelle werden nun die beiden Möglichkeiten der technischen Realisierung solcher Schwellenwerte behandelt:

1. Soll die Zuordnung der Klassen a und b auf jeden Fall verhindert werden, kann einfach die Verbindungskante zwischen a und b aus dem Netzwerkgraphen (siehe Abbildung 3) entfernt werden.

2. Falls die Zuordnung der beiden Klassen lediglich erschwert werden soll, ist es möglich, die Kosten c_{ab} der Verbindungskante zwischen a und b zu erhöhen. Wenn die Kosten der Kante erhöht werden, bis $c_{ab} > c_{a\varepsilon} + c_{\varepsilon b}$ gilt, wird die Kante ab sicher nicht in der Zuordnung enthalten sein, da es günstiger wäre, die Klasse a zu löschen und Klasse b zu erzeugen, als die Klassen aufeinander abzubilden.

In einer Implementierung des Verfahrens wurden Schwellenwerte wie in der (allgemeineren) zweiten Variante realisiert. Dabei werden die Kanten der Netzwerkgraphen zunächst mit den Kosten bewertet, die sich aus dem Kostenmodell ergeben. Im Anschluss daran erfolgt eine Iteration über alle Verbindungskanten des bipartiten Graphen auf Klassenebene, die gemäß einem vorgegebenen Kriterium gegebenenfalls die Kosten der Kanten erhöht. Das modifizierte Optimierungsproblem wird ebenfalls mit dem Busacker-Gowen-Verfahren gelöst, und für die modifizierten Kosten c'_{real} gilt:

$$c'_{real} \geq c_{real} \tag{9}$$

5 Rahmenwerk

In Abschnitt 3 haben wir einen allgemeinen graphbasierten Ansatz zum Vergleich von Modellversionen beschrieben, den wir in Abschnitt 4 auf Ecore-Klassendiagramme angewendet haben. Das dort beschriebene Matching-Verfahren wurde als Eclipse-Plugin implementiert und bildet zusammen mit weiteren Plugins des Eclipse Modeling Frameworks ([SBPM09]) ein Rahmenwerk, das die Integration alternativer Vergleichsalgorithmen unterstützt und damit das Experimentieren mit und das Evaluieren von Algorithmen zum Vergleich von Ecore-Klassendiagrammen ermöglicht (Abbildung 5). Um Ecore-Klassendiagramme zu erzeugen, können verschiedene bereits existierende auch graphische Editoren verwendet werden, wie z.B. von Ecore Tools[3]. Über die Benutzerschnittstelle unseres Plugins können die zu vergleichenden Diagramme und das zu verwendende Verfahren ausgewählt werden. Das Verfahren berechnet die Korrespondenzen, d.h. die aufeinander abgebildeten Elemente. Diese werden dann in ein EMF Match Model überführt, das aus einem Baum von Paaren identifizierter Elemente und deren identifizierten Unterelementen sowie aus einer Liste derjenigen Elemente besteht, die nicht zugeordnet werden konnten. Das EMF Match Model dient zusammen mit den Diagrammen als Eingabe für den Differenzberechnungsalgorithmus von EMF Compare, der aus den berechneten Korrespondenzen die Unterschiede ableitet und diese in einem EMF Diff Model speichert. Schließlich können die berechneten Differenzen im EMF Compare UI Editor in der Baumsicht visualisiert werden.

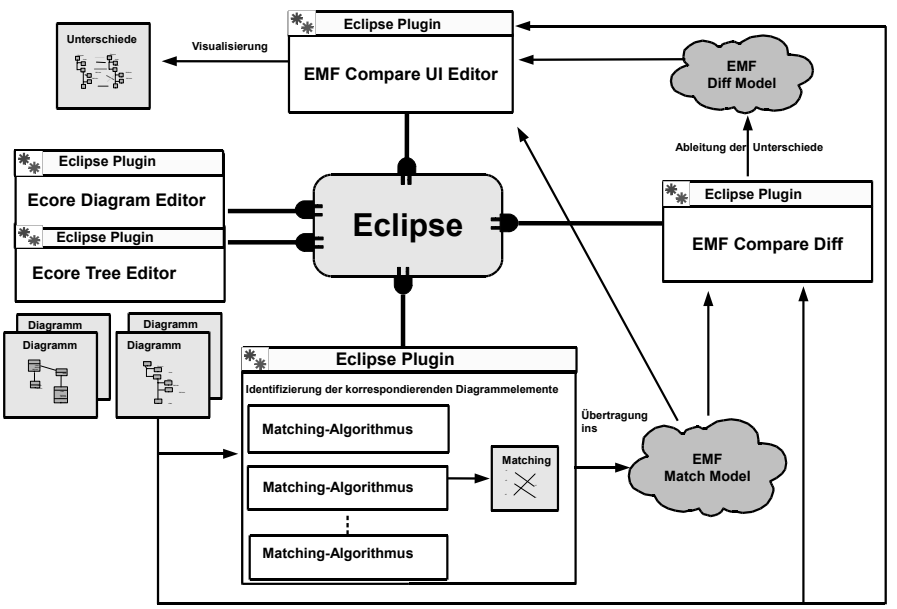


Abb. 5: Rahmenwerk für den Modellvergleich

[3] http://www.eclipse.org/modeling/emft/?project=ecoretools

6 Evaluierung

Die Vorteile des strukturbasierten Vergleichs von Klassendiagrammen, der sich nicht auf eindeutige Objektbezeichner oder die Identifikation von Namen stützt, wurde bereits in Abschnitt 2 erläutert. In diesem Abschnitt werden wir die Fragestellungen untersuchen, wie gut die untere Schranke ist, wie oft die optimale Lösung gefunden wird und in welcher Höhe die Kosten im Durchschnitt von der optimalen Lösung abweichen. Bei den Tests wurde die Version des Algorithmus eingesetzt, bei der die Zuordnung der korrespondierenden Elemente ausschließlich über das Kostenmodell gesteuert wird. Der Einsatz von Schwellenwerten kann aus Sicht des Benutzers zu nachvollziehbareren Lösungen führen, stellt jedoch einen nachträglichen Eingriff ins Kostenmodell dar.

Sind die geschätzten Kosten mit den realen Kosten identisch, so wurde mit dem Verfahren eine optimale Lösung gefunden. Auch in Fällen, in denen diese untere Schranke nicht erreicht wird, kann eine optimale Lösung gefunden worden sein, da die Erreichbarkeit der unteren Schranke nicht garantiert wird (vgl. Abschnitt 3.3). Um in diesen Fällen überprüfen zu können, ob die gefundene Lösung optimal ist, wurde eine Brute-Force-Lösung implementiert, welche die optimalen Kosten mit Aufwand $O(n!)$ ermittelt.

Die für die Evaluierung verwendeten Testmengen *A*, *B* und *C* enthalten jeweils verschiedene manuell erstellte Versionen eines Klassendiagramms. Daraus ergibt sich, dass sich die Diagramme innerhalb einer Testmenge ähnlich sind und sich die Diagramme aus verschieden Testmengen sehr stark unterscheiden. Die Testmenge *AB* ist die Vereinigung der Testmengen *A* und *B*. Die Größe der Klassendiagramme in diesem Test wurde durch die langen Laufzeiten des Brute-Force-Algorithmus limitiert. Die in Testmenge *A* enthaltenen Klassendiagramme sind die kleinsten mit bis zu 4 Klassen, Testmenge *B* enthält Klassendiagramme mit bis zu 8 Klassen, die Diagramme der Testmenge *C* bis zu 11 Klassen.

Die Ergebnisse der Tests sind in Tabelle 6 zu sehen. Die ersten beiden Spalten enthalten die verwendete Testmenge und die Anzahl der paarweise durchgeführten Vergleiche. In der 3. Spalte ist angegeben, in wieviel Prozent der Testfälle die Kosten der unteren Schranke erreicht wurden. In diesen Fällen konnte der Algorithmus eine optimale Lösung finden und verifizieren. Der Prozentsatz der tatsächlich gefundenen optimalen Lösungen steht in der 4. Spalte der Tabelle. Im Fall der Testgruppe B zeigt sich, dass obwohl die hinreichende Optimalitätsbedingung nur in 5,71% der Fälle erfüllt ist, die optimale Lösung dennoch in 96,19% der Fälle erreicht wird. Die 5. Spalte enthält die durchschnittliche Abweichung der Kosten von den Kosten der optimalen Lösung. Dabei entsprechen 100% den Kosten der optimalen Lösung. Hierbei werden in allen Testmengen gute Ergebnisse erzielt, auch in der Testmenge *AB*, in der sehr unterschiedliche Diagramme verglichen werden, liegen die durchschnittlichen Kosten nur bei 1,10 Prozentpunkten über den Kosten der optimalen Lösung. In den letzten beiden Spalten der Tabelle stehen die Laufzeiten, die auf einem Arbeitsplatzrechner (2,53 GHz, 4GB RAM) benötigt werden. In der Testmenge *AB* werden 1035 Tests in 1249 ms ausgeführt, so dass sich eine durchschnittliche Dauer von 1,12 ms pro Test ergeben. Aufgrund des polynomialen Aufwands des Verfahrens benötigt der Vergleich zweier Klassendiagramme mit 65 Klassen bzw. 80 Klassen mit 96 Änderungen im Mittel von 10 Tests immer noch lediglich 2255 ms. Im Gegensatz dazu benötigt die Brute-

Force-Lösung auf dem gleichen Rechner für zwei Klassendiagramme zu je 11 Klassen über 6 Stunden.

Die Ergebnisse dieser Evaluierung lassen sich folgendermaßen zusammenfassen: Die untere Schranke ist zwar ein relativ schlechtes Kriterium für Optimalität, dennoch findet der Algorithmus in sehr vielen Fällen eine optimale Lösung und insgesamt ist die durchschnittliche Abweichung von den optimalen Kosten gering. Darüber hinaus zeigen die Laufzeiten, dass dieses Verfahren auch in der Praxis gut einsetzbar ist. Im interaktiven Einsatz fällt die Laufzeit des Algorithmus im Vergleich zu anderen Aktionen, wie dem Aufbau des EMF Compare Editors, der die Unterschiede visualisiert, kaum ins Gewicht.

Menge	#	EC = RC	RC = BF	RC/BF	Gesamtzeit[ms]	∅[ms]
A	465	36,77%	96,77%	100,45%	191	0,41
B	105	5,71%	96,19%	100,30%	401	3,82
AB	1035	20,44%	70,58%	101,10%	1249	1,12
C	55	3,64%	60,00%	103,18%	361	6,56

Abb. 6: Ergebnisse

7 Zusammenfassung

In diesem Aufsatz wurde ein Verfahren zum Modellvergleich vorgestellt, das die Berechnung der korrespondierenden Modellelemente in ein vereinfachtes Optimierungsproblem überführt, das mit Hilfe eines graphentheoretischen Algorithmus mit polynomiellem Aufwand gelöst werden kann. Es ist möglich, die berechnete Zuordnung über die Ausgestaltung des Kostenmodells und den Einsatz von Schwellenwerten zu beeinflussen. Das Verfahren wurde bereits erfolgreich auf den Vergleich von Ecore-Klassendiagrammen übertragen und in ein EMF-basiertes Rahmenwerk zum Modellvergleich eingebettet. Die Ergebnisse der Evaluierung sind ermutigend sowohl in Bezug auf die Optimalität der berechneten Lösung als auch in Bezug auf die Laufzeiten. Im Rahmen zukünftiger Arbeiten ist es geplant, diesen generischen Ansatz auf weitere Diagrammarten anzuwenden.

Literaturverzeichnis

[AP03] Marcus Alanen und Ivan Porres. Difference and Union of Models. In *Proc. UML 2003*, LNCS 2863, Seiten 2–17, San Francisco, CA, 2003. Springer-Verlag.

[BE09] Lars Bendix und Pär Emanuelsson. Requirements for Practical Model Merge - An Industrial Perspective. In *Proc. (MODELS 2009)*, LNCS 5795, Seiten 167–180, Denver, CO, 2009. Springer-Verlag.

[BG61] R.G. Busacker und P.J. Gowen. A Procedure for Determining a Family of Minimum Cost Flow Networks. Bericht 15, John Hopkins University, Operations Research Office, Baltimore, MD, 1961.

[BP08] Cédric Brun und Alfonso Pierantonio. Model Differences in the Eclipse Modelling Framework. *UPGRADE*, IX(2):29–34, April 2008.

[FW07] Sabrina Förtsch und Bernhard Westfechtel. Differencing and Merging of Software Diagrams - State of the Art and Challenges. In *Proc. ICSOFT 2007*, Seiten 90–99, Barcelona, Spain, Juli 2007.

[HS77] J.W. Hunt und T.G. Szymanski. A Fast Algorithm for Computing Longest Common Subsequences. *Communications of the ACM*, 20(5):350–353, Mai 1977.

[Jun08] Dieter Jungnickel. *Graphs, Networks and Algorithms*. Springer-Verlag, Berlin, 2008.

[KN05] Sven Oliver Krumke und Hartmut Noltemeier. *Graphentheoretische Konzepte und Algorithmen*. Leitfäden der Informatik. Teubner Verlag, Wiesbaden, 2005.

[KWN05] Udo Kelter, Jürgen Wehren und Jörg Niere. A Generic Difference Algorithm for UML Models. In *Software Engineering 2005*, LNI 64, Seiten 105–116, Essen, 2005.

[MGH05] Akhil Mehra, John C. Grundy und John G. Hosking. A generic approach to supporting diagram differencing and merging for collaborative design. In *Proc. ASE 2005*, Seiten 204–213, Long Beach, CA, 2005.

[MGMR02] S. Melnik, H. Garcia-Molina und E. Rahm. Similarity Flooding: A Versatile Graph Matching Algorithm and ist Application to Schema Matching. In *Proc. 18th International Conference on Data Engineering*, Seiten 117–128, San Jose, CA, 2002.

[NM02] Klaus Neumann und Martin Morlock. *Operations Research*. Carl Hanser Verlag, 2002.

[OWK03] Dirk Ohst, Michael Welle und Udo Kelter. Differences between versions of UML diagrams. In *Proc. ESEC/FSE-11*, Seiten 227–236, Helsinki, 2003.

[RW98] J. Rho und C. Wu. An Efficient Version Model of Software Diagrams. In *Proc. Asia Pacific Software Engineering Conference*, Seiten 236–243, Taiwan, 1998.

[SBPM09] Dave Steinberg, Frank Budinsky, Marcelo Paternostro und Ed Merks. *EMF Eclipse Modeling Framework*. Addison-Wesley, Upper Saddle River, NJ, 2009.

[Uhr08] Sabrina Uhrig. Matching Class Diagrams: With Estimated Costs Towards the Exact Solution? In *Proc. CVSM 2008*, Seiten 7–12, Leipzig, 2008.

[XS05] Zhenchang Xing und Eleni Stroulia. UMLDiff: an algorithm for object-oriented design differencing. In *Proc. ASE 2005*, Seiten 54–65, Long Beach, CA, 2005.

Modellgetriebene Ableitung von BPMN-Workflowschemata aus SOM-Geschäftsprozessmodellen

Corinna Pütz[1] und Elmar J. Sinz[2]

Abstract: Die Business Process Model and Notation (BPMN) hat sich in den letzten Jahren zu einer der dominierenden graphischen Modellierungssprachen für Prozesse entwickelt. Ihre Nutzung erfolgt sowohl für die konzeptuelle Workflowmodellierung als auch für die Spezifikation ausführbarer Workflowschemata. Für die Modellierung von Geschäftsprozessen erscheint die BPMN allerdings nur bedingt geeignet. Geschäftsprozessmodelle beschreiben die auf Unternehmensziele ausgerichtete betriebliche Leistungserstellung und ihre Lenkung (Aufgabenebene). Diese Merkmale werden von der BPMN nicht explizit erfasst. Im Gegensatz dazu spezifizieren Workflowschemata Lösungsverfahren für die Durchführung betrieblicher Aufgaben (Aufgabenträgerebene). Der vorliegende Beitrag schlägt einen zweistufigen Modellierungsansatz zur Überwindung der semantischen Lücke zwischen Geschäftsprozessmodellen und Workflowschemata vor: In einem ersten Schritt wird ein Geschäftsprozessmodell gemäß dem Semantischen Objektmodell (SOM) erstellt und schrittweise verfeinert. Im zweiten Schritt wird aus dem hinreichend verfeinerten Geschäftsprozessmodell ein BPMN-Workflowschema auf Basis einer metamodellbasierten Schematransformation abgeleitet. Der Modellierungsansatz wird anhand des Fallbeispiels Online-Auktionshaus illustriert.

1 Einleitung

Die Business Process Model and Notation (BPMN) hat sich in den letzten Jahren zu einer der dominierenden graphischen Modellierungssprachen für Prozesse entwickelt. BPMN wird durch die Object Management Group (OMG) standardisiert und liegt aktuell in der Version 1.2 [OMG09a] vor. Die Version 2.0 ist in Vorbereitung.

Mit dem Einsatz der Sprache BPMN ([WM08, S. 24], [Al08, S. 9 ff]) werden unterschiedliche Ziele verfolgt:

- *Konzeptuelle Modellierung von Workflows*: Ziel ist insbesondere die fachliche Dokumentation von Workflows und die Schaffung einer Kommunikationsgrundlage für die Analyse und Gestaltung von Workflows in einer Organisation. Die Ausführbarkeit der Workflows steht hier nicht im Vordergrund.
- *Spezifikation ausführbarer Workflowschemata*: Ziel ist die vollständige und detaillierte Spezifikation von Workflows, um diese direkt z. B. in BPEL (WS-BPEL, Web-Services Business Process Execution Language [OA07]) transformieren und ausführen zu können. Als Repräsentationssprache für den Export von Workflowschemata aus einem BPMN-Werkzeug in eine BPEL-Engine kommt insbesondere die XML Process Definition Language (XPDL) [WF08] zum Einsatz.

[1] Universität Bamberg, Feldkirchenstr. 21, 96045 Bamberg, corinna.puetz@uni-bamberg.de
[2] Universität Bamberg, Feldkirchenstr. 21, 96045 Bamberg, elmar.sinz@uni-bamberg.de

Häufig wird zunächst eine konzeptuelle Modellierung von Workflows verfolgt, um die Ergebnisse anschließend zu ausführbaren Workflowschemata zu erweitern [Si09, S. 7 f]..

In Wissenschaft und Praxis erfolgt oft keine explizite Unterscheidung zwischen Geschäftsprozessen und Workflows (siehe z. B. [HW08], [Al08, S. 8]). Für die im vorliegenden Beitrag vorgestellte Methodik ist dagegen die Differenzierung zwischen Geschäftsprozessmodell und Workflowschema wesentlich:

- Ein Geschäftsprozessmodell spezifiziert, ausgerichtet auf vorgegebene Unternehmensziele, die betriebliche Leistungserstellung sowie deren Lenkung und referenziert die dafür eingesetzten Ressourcen [FS08, S. 193 f]. Die Beschreibung erfolgt in Form von betrieblichen Aufgaben und Ereignisbeziehungen zwischen Aufgaben. Der Begriff Aufgabe stellt eines der elementaren Konzepte der Betriebswirtschaftslehre dar [Ko76] und bezeichnet eine zielorientierte Verrichtung, welche an einem Aufgabenobjekt durchgeführt wird. Die Aufgabenziele werden dabei aus den Unternehmenszielen abgeleitet.
- Im Gegensatz dazu beschreibt ein Workflowschema ein Lösungsverfahren, welches personelle oder maschinelle Aufgabenträger (Personen bzw. Anwendungssysteme) im Rahmen der Durchführung einer oder mehrerer betrieblicher Aufgaben ausführen. Die Beschreibung erfolgt in Form von Aktivitäten und Beziehungen zwischen Aktivitäten. Der Begriff Aktivität bezeichnet eine elementare oder nicht-elementare Tätigkeit im Rahmen des genannten Lösungsverfahrens.

Die Namensgebung der BPMN legt nahe, dass die Sprache auf die Modellierung von Geschäftsprozessen zielt. Folgt man jedoch der obigen Differenzierung, so wird deutlich, dass die BPMN primär auf die Modellierung von Workflows ausgerichtet ist. Ihre zentralen Sprachelemente sind Aktivitäten und deren Beziehungen (Nachrichtenflüsse und Sequenzflüsse) [OM09a]. Zur Modellierung von Geschäftsprozessen im obigen Sinne ist BPMN dagegen nur bedingt geeignet, da betriebliche Aufgaben in ihrem Bezug zu Unternehmenszielen sowie zur betrieblichen Leistungserstellung und ihrer Lenkung nicht adäquat dargestellt werden können.

Im vorliegenden Beitrag wird vorgeschlagen, BPMN-Workflowschemata mithilfe eines modellgetriebenen Ansatzes aus Geschäftsprozessmodellen abzuleiten. Zur Geschäftsprozessmodellierung wird dabei das Semantische Objektmodell (SOM) eingesetzt [FS08, S. 192ff]. Ein SOM-Geschäftsprozessmodell wird dabei, ausgehend von initialen Leistungsflüssen zwischen Diskurswelt und Umwelt, schrittweise verfeinert. Dabei wird zunehmend die Lenkung der Leistungserstellung in Form von Koordinationsbeziehungen zwischen betrieblichen Objekten und deren Aufgaben sichtbar. Nachdem eine hinreichende Detaillierung erreicht ist, wird in einer modellgetriebenen Ableitung aus dem Geschäftsprozessmodell ein BPMN-Workflowschema entwickelt.

Im Vergleich zu einer direkten Spezifikation von BPMN-Workflowschemata besitzt der Ansatz eine Reihe von Vorteilen: Pools, die Choreographie zwischen den Pools, die Orchestrierung von Aktivitäten im Inneren von Pools und weitere Merkmale von BPMN-Workflows werden systematisch aus dem Geschäftsprozessmodell abgeleitet. Semantische Eigenschaften des Geschäftsprozessmodells werden dabei in die Workflow-

Spezifikation übertragen und reichern diese an. Die systematische Ableitung verfolgt zudem das Ziel, die Modellqualität der Workflowschemata zu verbessern.

Der weitere Beitrag ist wie folgt gegliedert: Kapitel 2 behandelt methodische Grundlagen für die modellgetriebene Ableitung von BPMN-Workflowschemata aus SOM-Geschäftsprozessmodellen. In Kapitel 3 wird als Fallbeispiel ein Online-Auktionshaus eingeführt und in Form eines mehrstufig verfeinerten SOM-Geschäftsprozessmodells beschrieben. Die modellgetriebene Ableitung eines BPMN-Workflowschemas aus dem SOM-Geschäftsprozessmodell ist Gegenstand von Kapitel 4. Kapitel 5 diskutiert den vorgeschlagenen Ansatz. Abschließend beleuchtet Kapitel 6 verwandte Arbeiten und gibt einen Ausblick auf weiteren Forschungsbedarf.

2 Methodische Grundlagen für die modellgetriebene Ableitung von Workflowschemata aus Geschäftsprozessmodellen

Das Semantische Objektmodell [FS08, S. 192ff] ist eine objekt- und geschäftsprozessorientierte Methodik zur ganzheitlichen Modellierung betrieblicher Systeme. Die SOM-Unternehmensarchitektur (Abb. 1a) umfasst drei Modellebenen: (1) Der Unternehmensplan beschreibt die Gesamtaufgabe des betrieblichen Systems aus Außenperspektive und legt dabei insbesondere deren Ziele fest. (2) Das Geschäftsprozessmodell beschreibt das Lösungsverfahren für die Realisierung des Unternehmensplans; es spezifiziert damit die Innenperspektive auf die Aufgaben des Unternehmens. (3) Das Ressourcenmodell beschreibt die Aufgabenträger zur Durchführung der betrieblichen Aufgaben ebenfalls aus der Innenperspektive des Unternehmens.

Korrespondierend zu den Modellebenen der SOM-Unternehmensarchitektur spezifiziert das SOM-Vorgehensmodell (Abbildung 1b) die Sichten zur Darstellung der drei Modellebenen. Im Rahmen des vorliegenden Beitrags werden das Interaktionsschema (IAS) und das Vorgangs-Ereignis-Schema (VES) zur Spezifikation des Geschäftsprozessmodells aus Struktur- bzw. Verhaltenssicht sowie das Vorgangsobjektschema (VOS) für die Beschreibung der Verhaltenssicht des Ressourcenmodells betrachtet. Die letztere Verhaltenssicht wird im vorliegenden Beitrag in Form eines BPMN-Workflowschemas spezifiziert.

Die in diesem Beitrag vorgeschlagene Methodik umfasst zwei Schritte:

1. Erstellung und schrittweise Verfeinerung eines SOM-Geschäftsprozessmodells (IAS und VES).
2. Modellgetriebene Ableitung eines BPMN-Workflowschemas aus dem VES der detailliertesten Zerlegungsstufe.

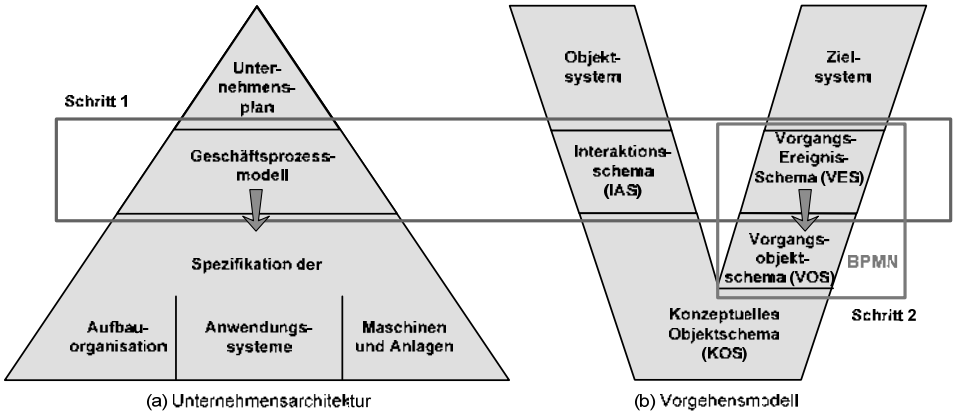


Abb. 1: Unternehmensarchitektur und Vorgehensmodell der SOM-Methodik [FS08, S. 193, 195]

Schritt 1: Das Metamodell für SOM-Geschäftsprozessmodelle ist in Abb. 9 (Metamodell SOM-GP) dargestellt. Abb. 2 zeigt exemplarisch die Struktursicht eines SOM-Geschäftsprozessmodells. Das IAS enthält die beiden betrieblichen Objekte *Käufer* (Umweltobjekt) und *Online-Auktionshaus* (Diskursweltobjekt). Ein betriebliches Objekt umfasst eine Menge von Aufgaben, die auf einem gemeinsamen Aufgabenobjekt durchgeführt werden, mit zugehörigen Sach- und Formalzielen. Die Koordination dieser betrieblichen Objekte erfolgt durch Transaktionen. Es werden zwei Koordinationsformen unterschieden, eine hierarchische Koordination unter der Nutzung von Steuer- und Kontrolltransaktionen (S, K) und eine nicht-hierarchische Koordination mithilfe von Anbahnungs-, Vereinbarungs- und Durchführungstransaktionen (A, V, D) [FS08, S. 66 ff]. In Abb. 2 wird die Ersteigerung eines Gutes durch *Information* des (potenziellen) Käufers über verfügbare Auktionen angebahnt (Anbahnung). Dieser gibt ein oder mehrere verbindliche *Gebote* ab (Vereinbarung) und erhält ggf. den *Gebotszuschlag* (Durchführung).

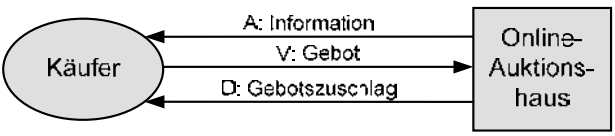


Abb. 2: Nicht-hierarchische Koordination zwischen Online-Auktionshaus und Käufer

Die zugehörige Verhaltenssicht ist in Abb. 3 in Form eines VES dargestellt. Inhalt des VES ist der ereignisgesteuerte Ablauf von Aufgabendurchführungen (vgl. Abb. 9, Metamodell SOM-GP). Dem VES liegt das Systemparadigma „Petri-Netz“ (vgl. z. B. [Re86], [Pe77]) zugrunde (Abb. 3a). Die Ausführung eines Petri-Netzes erfolgt durch Schalten zulässiger Übergänge (Transitionen). Zum Beispiel ist der Übergang *Info empfangen* zulässig, wenn der Zustand (Stelle) *Infoübertragung* markiert ist. Das Schalten des Übergangs führt zur Markierung des Zustands *Info liegt vor*.

Ein VES wird als erweitertes Petri-Netz mit folgenden Merkmalen verstanden: Es handelt sich um ein gefärbtes Petri-Netz (vgl. z. B. [JK09, S. 13 ff]) mit unterscheidbaren Marken (Unterscheidung von Käufern und Geboten). Die Übergänge können um Pre- und Post-Conditions ergänzt werden, welche das Schaltverhalten genauer spezifizieren. Schließlich wird festgelegt, dass die beiden Übergänge, welche die an einer betrieblichen Transaktion beteiligten Aufgaben repräsentieren, synchron schalten. Der die beiden Übergänge verbindende Zustand wird im VES daher nicht repräsentiert. Das mit dem Petri-Netz korrespondierende VES ist in Abbildung 3b dargestellt.

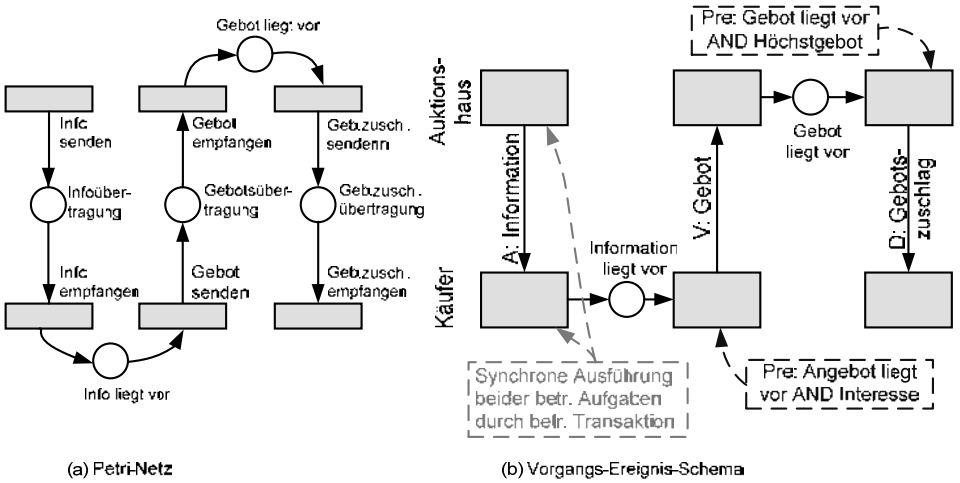


Abb. 3: Vorgangs-Ereignis-Schema Online-Auktionshaus

Schritt 2: Die für das abzuleitende Workflowschema relevanten Konstrukte der BPMN sind in Abb. 9 in Form eines Metamodells zusammengefasst. Gemäß OMG werden die Konstrukte in die drei Hauptkategorien *Swim Lane*, *Flow Object* und *Connecting Object* eingeteilt [OM09a]. Eine *Swim Lane* spezifiziert entweder einen Pool oder eine Lane. Ein Pool stellt einen Teilnehmer dar, der gegebenenfalls durch Lanes in Rollen unterteilt wird. *Flussobjekte* werden in (1) Activities (atomarer Task oder nicht-atomarer Subprocess), (2) Events (Start-, Zwischen- oder Endereignis) oder (3) Gateways (Teilung, bzw. Synchronisation des Sequenzflusses) unterschieden. *Connecting Objects* sind zum einen Sequenzflüsse (Sequence Flows), welche die Reihenfolge beschreiben, in der Flussobjekte innerhalb eines Pools ausgeführt werden. Zum anderen sind sie Nachrichtenflüsse (Message Flows), welche den Nachrichtenaustausch zwischen zwei Pools spezifizieren.

Während einem VES das Systemparadigma „Petri-Netz“ zugrunde liegt, folgt ein BPMN-Schema dem Systemparadigma „Algorithmus“. Die Instanziierung eines Petri-Netzes wird durch die den einzelnen Zuständen zugeordneten Marken beschrieben. Im Gegensatz dazu wird ein BPMN-Workflowschema mehrfach instanziiert. Jede Instanz entspricht einem Ablauf des Schemas, dessen aktueller Ausführungszustand durch ein Token markiert ist.

Die Ableitung eines initialen BPMN-Workflowschemas aus einem detaillierten VES erfolgt als metamodellbasierte Schematransformation. Diese wird in Kapitel 4 beschrieben.

3 Fallbeispiel Online-Auktionshaus

Als Fallbeispiel dient ein Online-Auktionshaus, wie es z. B. in Form von eBay[3] weite Bekanntheit erreicht hat. Für dieses Online-Auktionshaus wird im Folgenden ein SOM-Geschäftsprozessmodell entwickelt. Dabei erfolgt eine mehrstufige, schrittweise Zerlegung des IAS und des korrespondierenden VES.

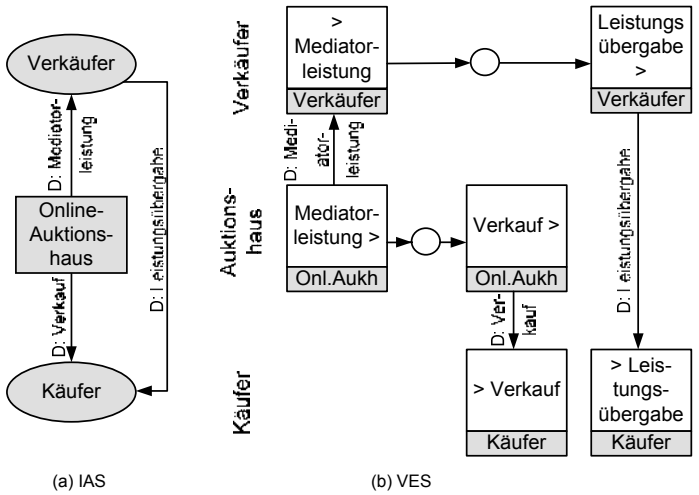


Abb. 4: IAS und VES für das Online-Auktionshaus (initiales Modell)

Das initiale IAS (Abb. 4a) zeigt die aggregierten Leistungsflüsse zwischen den betrieblichen Objekten aus Struktursicht. Das Diskursweltobjekt *Online-Auktionshaus* generiert für das Umweltobjekt *Käufer* eine Mediatorleistung und für das Umweltobjekt *Verkäufer* eine Verkaufsleistung. Beide Leistungen werden in korrespondierenden Durchführungstransaktionen übergeben. Die Leistungsübergabe des vermittelten Gutes erfolgt direkt zwischen *Verkäufer* und *Käufer*. Das korrespondierende VES ist in Abbildung 4b dargestellt. Die Bezeichnungen der Aufgaben im VES werden aus den Transaktionsbezeichnungen abgeleitet. So bezeichnet *Mediatorleistung*> die Aufgabe „Erstellen und Übergeben der Mediatorleistung“, >*Mediatorleistung* die korrespondierende Empfangsaufgabe.

Die Koordination zwischen Online-Auktionshaus und Verkäufer bzw. Käufer erfolgt nach dem Prinzip der nicht-hierarchischen Koordination. In einer ersten Zerlegung (Ab-

[3] http://www.ebay.de/

bildung 5) werden die beiden vom Online-Auktionshaus ausgehenden Durchführungstransaktionen nach dem AVD-Prinzip verfeinert. Nach erfolgtem Kontozugang (Anbahnung) erteilt der Verkäufer einen Auktionsauftrag (Vereinbarung). Nach Abschluss der Auktion wird er über deren Ausgang informiert und es erfolgt die Abrechnung (Durchführung). Entsprechend erhält der Käufer Informationen über eingerichtete Auktionen (Anbahnung), er kann für eine Auktion Gebote abgeben (Vereinbarung) und erhält ggf. den Gebotszuschlag (Durchführung).

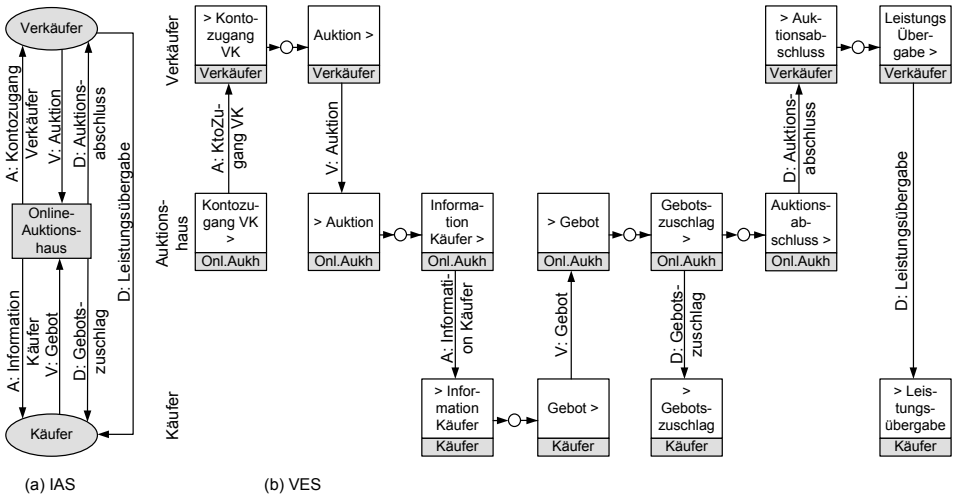


Abb. 5: IAS und VES für das Online-Auktionshaus (erste Zerlegung)

In der zweiten Zerlegung (Abbildungen 6 und 7) werden die Koordinationsprotokolle zwischen *Online-Auktionshaus* und *Verkäufer* bzw. *Käufer* weiter zerlegt. Darüber hinaus erfolgt eine Zerlegung des betrieblichen Objekts *Online-Auktionshaus*. Letztere führt zu je einem Teilobjekt für die Koordination auf Käufer- und auf Verkäuferseite sowie zu einem Teilobjekt, welches den jeweiligen Bestand an Geboten und Auktionen verwaltet. *Gebots- und Auktionsverwaltung* wird von *Gebotshandhabung* und von *Auktionsbeauftragung* im Rahmen nicht-hierarchischer Koordinationen beauftragt. Das gesamte Protokoll der Objekt- und Transaktionszerlegungen ist in Abbildung 7 zusammengefasst.

Im VES der zweiten Zerlegung (Abbildung 8) wurden – soweit notwendig – Aufgaben um Pre- und Postconditions ergänzt. Diese sind vor bzw. nach der jeweiligen Aufgabenbezeichnung eingetragen und mit dem Schlüsselwort PRE bzw. POST gekennzeichnet. Zum Beispiel fordert der Käufer nur dann eine Auktionsübersicht an (*Anf.Auktionsübersicht*>), wenn er angemeldet ist und grundsätzliches Interesse an der Teilnahme an einer Auktion hat.

Im Weiteren wird nun davon ausgegangen, dass das VES der zweiten Zerlegung hinreichend detailliert ist, um daraus ein BPMN-Workflowschema ableiten zu können. Diese Ableitung wird im nächsten Kapitel erläutert.

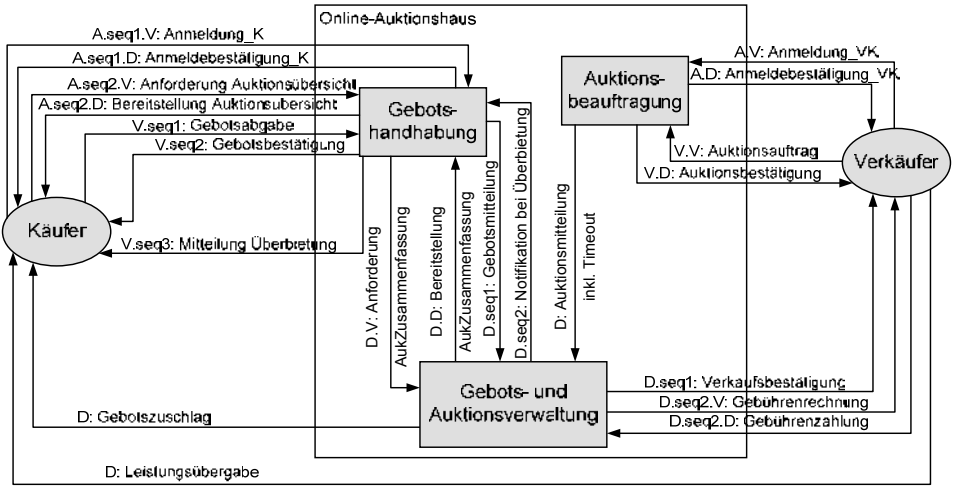


Abb. 6: IAS für das Online-Auktionshaus (zweite Zerlegung)

Objektzerlegung	**Transaktionszerlegung**	
Online-Auktionshaus Auktionsbeauftragung D: Auktionsmitteilung inkl. Timeout Gebots- und Auktionsverwaltung D: AuktionsZusammenfassung D.V: Anforderung AukZusammenfassung D.D: Bereitstellung AukZusammenfassung D: Gebotsmeldungen D.seq1: Gebotsmitteilung D.seq2: Notifikation bei Überbietung Gebotshandhabung Käufer Verkäufer	D: Verkauf A: Information Käufer A.seq1: Kontozugang Käufer A.seq1.V: Anmeldung_K A.seq1.D: Anmeldebestätigung_K A.seq2: Auktionsübersicht A.seq2.V: Anf. Auktionsübersicht A.seq2.D: Bereitst.. Auktionsübersicht V: Gebot V.seq1: Gebotsabgabe V.seq2: Gebotsbestätigung V.seq3: Mitteilung Überbietung D: Gebotszuschlag	D: Mediatorleistung A: Kontozugang Verkäufer A.V: Anmeldung_VK A.D: Anmeldebestätigung_VK V: Auktion V.V: Auktionsauftrag V.D: Auktionsbestätigung D: Auktionsabschluss D.seq1: Verkaufsbestätigung D.seq2: Abrechnung D.seq2.V: Gebührenrechnung D.seq2.D: Gebührenzahlung D: Leistungsübergabe

Abb. 7: Objekt- und Transaktionszerlegung Online-Auktionshaus

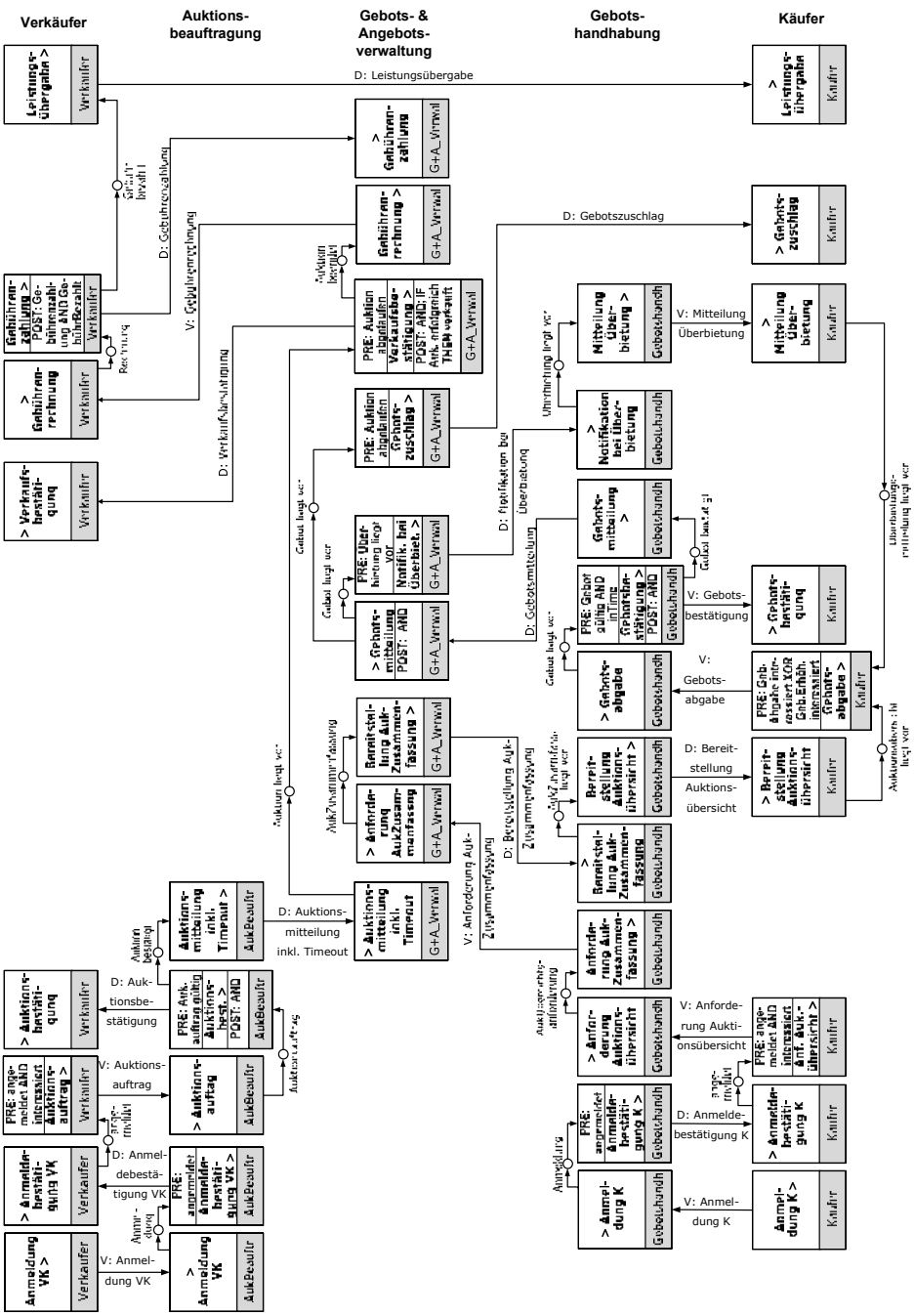


Abb. 8: Vorgangs-Ereignis-Schema für das Online-Auktionshaus (zweite Zerlegung)

4 Vom SOM-Geschäftsprozessmodell zum BPMN-Workflowschema

Methodische Grundlage der Ableitung eines BPMN-Workflowschemas aus einem SOM-Geschäftsprozessmodell ist die metamodellbasierte Schematransformation gemäß dem MDA-Pattern der Model Driven Architecture ([OM03, S. 3-9], siehe auch [Fr03], [GPR06]) (Abbildung 9, links). Ausgangspunkt der Ableitung ist die Verhaltenssicht, d. h. das VES, eines hinreichend verfeinerten SOM-Geschäftsprozessmodells, welches gemäß dem zugehörigen Metamodell spezifiziert ist. Das Ergebnis der Ableitung ist ein initiales BPMN-Workflowschema gemäß dem erstellten BPMN-Metamodell. Die Ableitung selbst wird anhand einer Abbildung von Modellbausteinen des SOM-Metamodells auf Modellbausteine des erstellten BPMN-Metamodells beschrieben (siehe gestrichelte Linien in Abbildung 9, rechts).

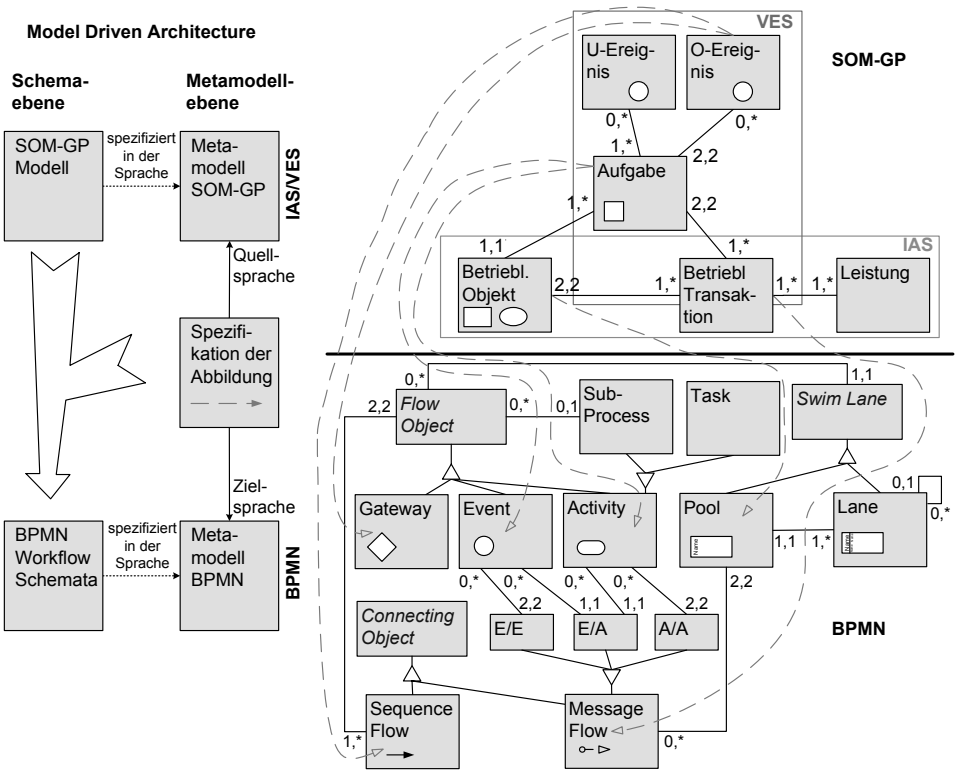


Abb. 9: Metamodell-Abbildung SOM → BPMN (vgl. [OM03, S. 3-9], [FS08, S. 210])

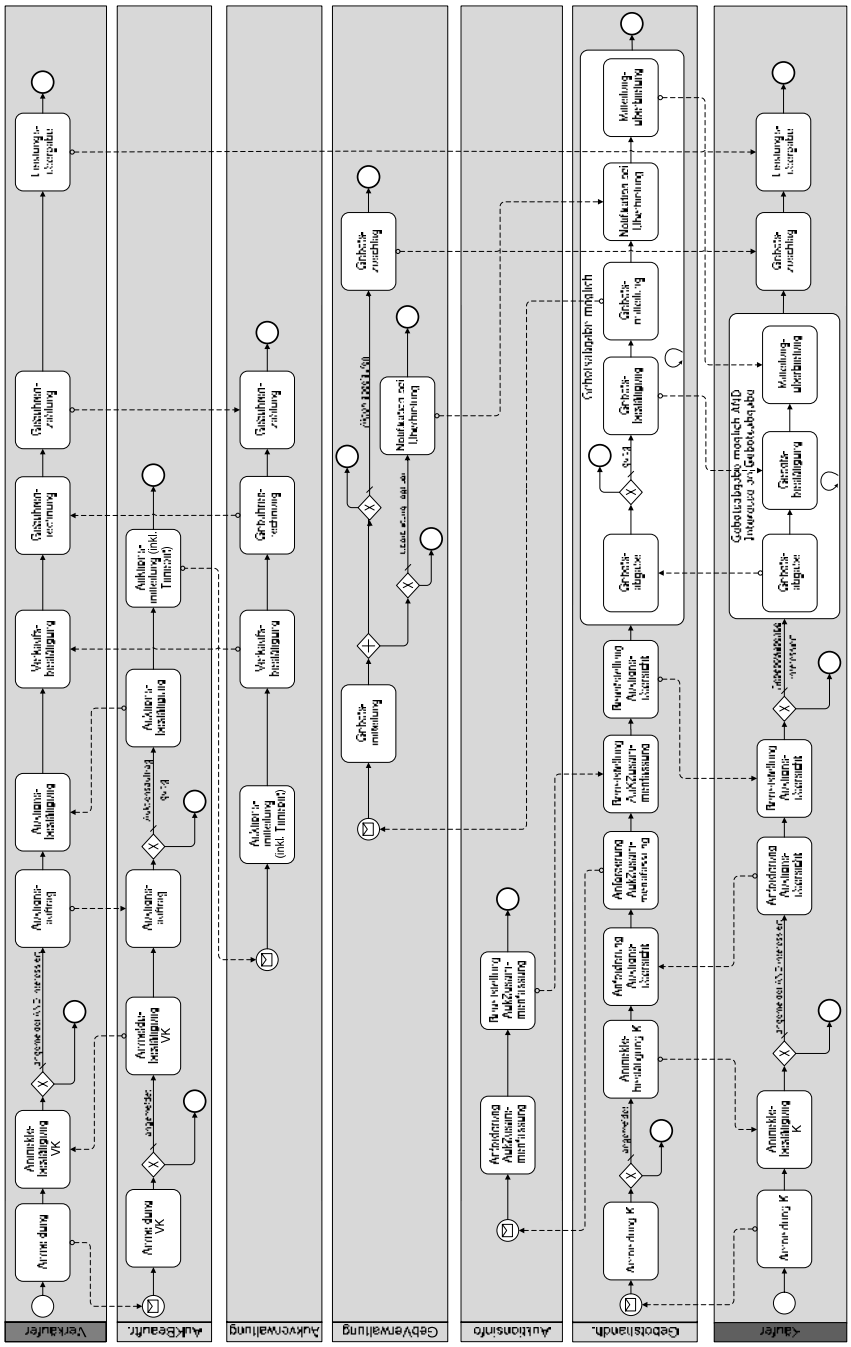


Abb. 10: BPMN-Workflowschema für das Online Auktionshaus

Die wichtigsten Beziehungselemente der Abbildung zwischen einem SOM-Geschäftsprozessmodell und einem BPMN-Workflowschema werden nun kurz erläutert: Jedes betriebliche Objekt wird in einen Pool transformiert. Betriebliche Transaktionen zwischen den Objekten führen zu Nachrichtenflüssen zwischen Pools. Aufgaben werden in Aktivitäten und ggf. Events transformiert. Ein objektinternes Ereignis, welches zwei Aufgaben im Inneren eines betrieblichen Objekts verknüpft, entspricht einem Sequenzfluss, der ebenfalls nur innerhalb eines Pools existiert. Aus objektinternen Ereignissen in Verbindung mit Pre- oder Post-Conditions von Aufgaben werden Gateways abgeleitet. Aufeinander folgende Flussobjekte innerhalb eines Pools werden zusätzlich durch Sequenzflüsse miteinander verbunden.

Durch die Ableitung wird erreicht, dass Abläufe im Inneren eines betrieblichen Objekts in Abläufe innerhalb eines Pools abgebildet werden. Die Koordination zwischen betrieblichen Objekten führt zu einer Choreographie zwischen Pools, die in Form von Nachrichtenflüssen spezifiziert wird (siehe hierzu [OM09a]).

Das aus der Ableitung resultierende BPMN-Workflowschema ist in Abb. 10 dargestellt. Während sich die betrieblichen Objekte *Verkäufer*, *Auktionsbeauftragung*, *Kaufverhandlung* und *Käufer* des SOM-Geschäftsprozessmodells direkt in korrespondierende Pools des BPMN-Schemas überführen lassen, bedarf die Abbildung des betrieblichen Objekts *Gebots- und Auktionsverwaltung* einer speziellen Behandlung. Diese ist dem Übergang vom Systemparadigma „Petri-Netz“ beim VES zum Systemparadigma „Algorithmus“ beim BPMN-Schema geschuldet. Während der VES-Ausschnitt des betrieblichen Objekts *Gebots- und Auktionsverwaltung* drei Einstiegspunkte in Form der Aufgaben *>Auktionsmitteilung*, *>Anforderung Auktionszusammenfassung* und *>Gebotsmitteilung* aufweist, darf ein Pool eines ausführbaren BPMN-Schemas jeweils nur ein einziges Startereignis aufweisen, damit der zugehörige Ablauf algorithmisch determiniert ist.

Die drei Einstiegspunkte des betrieblichen Objekts *Gebots- und Auktionsverwaltung* resultieren aus der Tatsache, dass sich *Auktionsmitteilungen*, *Anforderungen von Auktionszusammenfassungen* und *Gebotsmitteilungen* auf ein gemeinsames Aufgabenobjekt, d. h. Gebote und Auktionen, beziehen. Aus dem Blickwinkel der objektorientierten Modellierung ist das betriebliche Objekt daher nicht mehr weiter zerlegbar. Im ablauforientierten BPMN-Schema spielt diese Objektbindung jedoch keine Rolle. Bei der Ableitung wird damit das betriebliche Objekt *Gebots- und Auktionsverwaltung* in mehrere Pools mit je einem Startereignis abgebildet (*Auktionsinfo*, *Gebotsverwaltung*, *Auktionsverwaltung*).

5 Diskussion des vorgeschlagenen Ansatzes

Ausgangspunkt des vorgeschlagenen Ansatzes ist ein initiales SOM-Geschäftsprozessmodell, welches die mit dem Unternehmensplan abgestimmten Leistungsbeziehungen des Geschäftsprozesses mit seiner Umwelt beinhaltet. Dieses Geschäftsprozessmodell wird in einer Struktursicht (IAS) sowie einer korrespondierenden Verhaltenssicht (VES) dargestellt und durch Zerlegung von betrieblichen Transaktionen und Objekten in meh-

reren Schritten verfeinert. Dabei wird sukzessive die transaktionsbasierte Koordination zwischen den betrieblichen Objekten aufgedeckt. Nach Erreichen einer geeigneten Verfeinerung des Geschäftsprozessmodells erfolgt die modellgetriebene Ableitung eines initialen BPMN-Workflowschemas aus dem VES der detailliertesten Zerlegungsstufe. Eine geeignete Verfeinerung liegt vor, wenn für alle Aufgaben des Geschäftsprozessmodells das Lösungsverfahren auf genau eine Aktivität des Workflowschemas abbildbar ist.

Im Vergleich zu einer direkten Modellierung eines BPMN-Workflowschemas ohne ein vorgeschaltetes SOM-Geschäftsprozessmodell ergibt sich durch die modellgetriebene Ableitung eine Reihe von Vorteilen:

- Pools, die Choreographie zwischen Pools sowie die Aktivitäten und Beziehungen des initialen BPMN-Schemas sind Ergebnis der Ableitung und müssen nicht auf der Ebene des Workflowschemas konstruiert oder rekonstruiert werden [vgl. hierzu auch Al08, S. 53 ff].
- Die „Herkunft aus dem Geschäftsprozessmodell“ kann für eine semantische Annotation der Artefakte des Workflowschemas genutzt werden. Diese Anreicherung erhöht die semantische Aussagekraft eines BPMN-Schemas. Zum Beispiel kann ein Message-Flow um die Information ergänzt werden, welcher betrieblichen Transaktion er im Geschäftsprozessmodell entspricht. Die Choreographie zwischen Pools spiegelt das gesamte Koordinationsprotokoll zwischen den korrespondierenden betrieblichen Objekten wider.
- Die semantische Lücke zwischen dem mit dem Unternehmensplan abgestimmten initialen Geschäftsprozessmodell und dem Workflowschema, welches die Lösungsverfahren für die Durchführung der detaillierten Geschäftsprozessaufgaben spezifiziert, wird in überschaubaren und damit prüfbaren Komplexitätsspannen überwunden.
- Das durch Modelltransformation erzeugte initiale BPMN-Schema setzt den Rahmen für dessen weitere Bearbeitung. Das Workflowschema kann verfeinert werden, um z. B. Varianten von Aufgabendurchführungen zu spezifizieren. Eine Abänderung der durch die Ableitung erzeugten Grundstruktur des BPMN-Schemas ist jedoch nicht möglich, da dadurch die Abstimmung zwischen Geschäftsprozessmodell und Workflowschema verletzt würde.

Insgesamt zielt der Ansatz damit auf eine Verbesserung der Modellqualität und der semantischen Aussagekraft von Workflowschemata.

Voraussetzung für die Anwendung des Ansatzes ist die konzeptuelle Unterscheidung zwischen Geschäftsprozessmodell und Workflowschema (siehe Kapitel 1). Das Geschäftsprozessmodell ist der Aufgabenebene eines betrieblichen Systems zugeordnet und beschreibt das „Was“ der zielorientierten Aufgabenerfüllung. Das Workflowschema hingegen ist Teil der Aufgabenträgerebene und beschreibt das „Wie“ der Aufgabendurchführung. Der vorgeschlagene Ansatz führt zu einer Abstimmung der beiden Ebenen, macht aber gleichzeitig die Freiheitsgrade beim „Wie“ zum Erreichen eines vorgegebenen „Was“ deutlich. Zum Beispiel können für einen gegebenen Geschäftsprozess

unterschiedliche Varianten von Workflows durch Verfeinerung des initialen Workflowschemas erzeugt werden.

6 Verwandte Arbeiten und weiterer Forschungsbedarf

Verwandte Arbeiten betreffen insbesondere Modelltransformationen von und nach BPMN. Dabei nutzen relativ wenige Ansätze BPMN als Zielsprache für die Modelltransformation. Beispiele sind u. a. die Arbeiten [Al07] und [KV06], wo Ereignisgesteuerte Prozessketten bzw. UML-Aktivitätsdiagramme in eine BPMN-Darstellung transformiert werden.

Im Vergleich dazu verwenden viele Arbeiten BPMN als Quellsprache für eine Modelltransformation. Zum Beispiel befassen sich [De08a] mit der Überführung von BPMN in die Workflowsprache YAWL, [DDO08] mit der Transformation in Petri-Netze und [Ci09] mit der Abbildung in UML-Aktivitätsdiagramme. Eine Reihe von Arbeiten beschäftigt sich mit der Generierung von BPEL-Spezifikationen aus BPMN (z. B. [OM09a] [Ou06a], [Ou06b]). Der Tatbestand, dass für BPMN leistungsfähige Editoren, Transformatoren in unterschiedliche Zielsprachen und BPEL-Generatoren verfügbar sind, spricht für die Wahl von BPMN als Sprache zur Spezifikation von Workflowschemata.

Fragen der Choreographiemodellierung mit BPMN werden u. a. in [DB08] behandelt. Zudem wird der neue Sprachstandard BPMN 2.0 eigene Sprachelemente zur Choreographiemodellierung enthalten [OM09b]. Mit der Transformation von BPMN in BPEL4Chor befassen sich [De08b].

Eine Reihe von Arbeiten verwendet das „Auktionsbeispiel“ für die Veranschaulichung einer direkten Modellierung von Geschäftsprozessen bzw. Workflows. Beispiele, die zum Vergleich mit den hier erzielten Ergebnissen anregen sollen, sind [DB08], [DP07] und [PGW09].

In der vorgestellten Form weist der Ansatz zur modellgetriebenen Ableitung von BPMN-Workflowschemata einige Restriktionen auf. Diese betreffen insbesondere die Berücksichtigung der Datensicht von Workflows (z. B. Bedingungen für Gateways), die Ausnahmebehandlung bei Workflows (z. B. Blockieren einer Workflowinstanz aufgrund des Verhaltens einer korrespondierenden Workflowinstanz) sowie die explizite Berücksichtigung des Automatisierungsgrades von Geschäftsprozessaufgaben. Aspekte der Einbeziehung von nichtautomatisierten Aktivitäten werden auch im neuen Sprachstandard BPMN 2.0 und in BPEL4People [OA09] behandelt.

Darüber hinaus ergibt sich weiterer Forschungsbedarf im Hinblick auf die Generalisierung und Formalisierung der Ableitungsregeln vom VES zum BPMN-Schema, bezüglich der Verfeinerung der Aktivitäten und hinsichtlich einer geeigneten Werkzeugunterstützung der Transformation. Damit soll eine durchgängige und weitgehend modellgetriebene Entwicklung von lauffähigen BPEL-Prototypen erreicht werden. Voraussetzung hier-

für ist die Einbeziehung der Aufgabenobjekte gemäß der SOM-Methodik (siehe Abbildung 1b, Konzeptuelles Objektschema). Diese sind in Form von Diensten zu realisieren, welche die von den Aktivitäten des Workflowschemas benötigten persistenten Daten verwalten.

Literaturverzeichnis

[Al07] Allweyer, T.: Erzeugung detaillierter und ausführbarer Geschäftsprozessmodelle durch Modell-zu-Modell-Transformationen. In (Nüttgens, M.; Rump, F. J.; Gadatsch, A. Hrsg.): EPK 2007 - Geschäftsprozessmanagement mit Ereignisgesteuerten Prozessketten. 6. Workshop der GI, St. Augustin, 2007, S. 23-39.

[Al08] Allweyer, T.: BPMN - Business Process Modeling Notation. Einführung in den Standard für die Geschäftsprozessmodellierung. Books on Demand, Norderstedt, 2008.

[Ci09] Cibrán, M. A.: Translating BPMN Models into UML Activities. In (Ardagna, D. Hrsg.): Proc. Business Process Management. BPM 2008. International Workshops. Springer, Berlin, 2009; S. 236–247.

[DB08] Decker, G.; Barros, A.: Interaction Modeling Using BPMN. In (Hofstede, A. H. M. ter Hrsg.): Proc. on Business Process Management.. BPM 2007. International Workshops. Springer, Berlin, 2008; S. 208–219.

[DDO08] Dijkman, R. M.; Dumas, M.; Ouyang, C.: Semantics and analysis of business process models in BPMN. In Information and Software Technology, 2008, 50; S. 1281-1294.

[De08a] Decker, G. et al.: Transforming BPMN Diagrams into YAWL Nets. In (Dumas, M. Hrsg.): Proc. 6th Int. Conf. on Business Process Management. BPM 2008, Springer, Berlin, 2008; S. 386-389.

[De08b] Decker, G. et al.: Modeling Service Choreographies Using BPMN and BPEL4Chor. In (Bellahsène, Z. Hrsg.): Proc. 20th Int. Conf. on Advanced Information Systems Engineering. CAiSE 2008; Springer, Berlin, 2008; S. 79–93.

[DP07] Decker, G.; Puhlmann, F.: Extending BPMN for Modeling Complex Choreographies. In (Meersman, R.; Tari, Z. Hrsg.): Proc. OTM Confederated International Conferences CoopIS, DOA, ODBASE, GADA, and IS 2007. Springer, Berlin, 2007; S. 24–40.

[Fr03] Frankel, D. S.: Model Driven Architecture. Applying MDA to Enterprise Computing. Wiley, Indianapolis, Ind., 2003.

[FS08] Ferstl, O. K.; Sinz, E. J.: Grundlagen der Wirtschaftsinformatik. Oldenbourg. 6. Auflage, München, 2008.

[GPR06] Gruhn, V.; Pieper, D.; Röttgers, C.: MDA®. Effektives Software-Engineering mit UML2® und Eclipse. Springer, Berlin, 2006.

[HW08] Huth, S.; Wieland, T.: Geschäftsprozessmodellierung mittels Software-Services auf Basis der EPK. In (Nissen, V.; Petsch, M.; Schorcht, H. Hrsg.): Service-orientierte Architekturen. Chancen und Herausforderungen bei der Flexibilisierung und Integration von Unternehmensprozessen. Deutscher Universitäts-Verlag, Wiesbaden, 2008; S. 61–76.

[JK09] Jensen, K.; Kristensen, L. M.: Coloured Petri Nets. Modelling and Validation of Concurrent Systems. Springer, Berlin, 2009.

[Ko76] Kosiol, E.: Organisation der Unternehmung. Gabler. 2. Auflage, Wiesbaden, 1976.

[KV06] Kalnins, A.; Vitolins, V.: Use of UML and Model Transformations for Workflow Process Definitions. In TECHNIKA, 2006, 3.

[OA07] OASIS: Web Services Business Process Execution Language Version 2.0. OASIS Standard. 11.04.2007. http://docs.oasis-open.org/wsbpel/2.0/OS/wsbpel-v2.0-OS.html, Abruf am 31.10.2009.

[OA09] OASIS: WS-BPEL Extension for People (BPEL4People) Technical Committee. http://www.oasis-open.org/committees/bpel4people/charter.php, Abruf am 02.11.2009.

[OM03] OMG: MDA Guide Version 1.0.1. Document Number: omg/2003-06-01. http://www.omg.org/cgi-bin/doc?omg/03-06-01.pdf, Abruf am 28.10.2009.

[OM09a] OMG: Business Process Model and Notation (BPMN) - Version 1.2. OMG Document Number: formal/2009-01-03. http://www.omg.org/spec/BPMN/1.2/PDF, Abruf am 28.10.2009.

[OM09b] OMG: Business Process Model and Notation (BPMN). FTF Beta 1 for Version 2.0. OMG Document Number: dtc/2009-08-14. http://www.omg.org/spec/BPMN/2.0/Beta1/PDF, Abruf am 28.10.2009.

[Ou06a] Ouyang, C. et al.: Translating BPMN to BPEL. BPM Center Report BPM-06-02, BPMcenter.org, 2006.

[Ou06b] Ouyang, C. et al.: From BPMN Process Models to BPEL Web Services: IEEE. Int. Conf. on Web Services. ICWS 2006. IEEE Computer Society, 2006; S. 285–292.

[Pe77] Peterson, J. L.: Petri Nets. In ACM Computing Surveys, 1977, 9; S. 223–252.

[PGW09] Pascalau, E.; Giurca, A.; Wagner, G.: Validating Auction Business Processes using Agent-based Simulations, Proc. 2nd Int. Conf. on Business Process and Services Computing. BPSC2009. Leipzig, 2009.

[Re86] Reisig, W.: Petrinetze. Eine Einführung. Springer. 2. Auflage, Berlin, 1986.

[Si09] Silver, B.: BPMN method and style. Cody-Cassidy, Aptos, 2009.

[WF08] WFMC: Process Definition Interface. XML Process Definition Language. Document Number WFMC-TC-1025. Version 2.1a. 2008. http://www.wfmc.org/index.php?option=com_docman&task=doc_download&Itemid=72&gid=132, Abruf am 02.11.2009.

[WM08] White, S. A.; Miers, D.: BPMN Modeling and Reference Guide. Understanding and Using BPMN. Future Strategies Inc., Lighthouse Point, Florida, 2008.

Webbasierte Modelltransformation in der Geoinformatik

Andreas Donaubauer[1], Tatjana Kutzner[2], Hans Rudolf Gnägi[3], Stefan Henrich[4] und Astrid Fichtinger[5]

Abstract: In der Geoinformatik, welche sich mit der Modellierung raumbezogener Strukturen und Prozesse in Geoinformationssystemen (GIS) beschäftigt, spielt die Modelltransformation eine große Rolle. Neu ist dabei aus Sicht der Geoinformatik, dass die Modelltransformation nicht auf Format-/Datenebene durch bilaterales Umformatieren implementiert wird, sondern auf die Ebene der konzeptionellen Modelle (UML) gehoben wird. Dort kann die Modelltransformation anschaulich grafisch definiert werden und die (physische) Datentransformation aufgrund der konzeptionell definierten Modelltransformation anschließend automatisch erfolgen. In diesem Beitrag wird eine Methode vorgestellt, wie sowohl Modell- als auch Datentransformation webbasiert durchgeführt werden können. Die Methode besteht aus einer Sprache zur Beschreibung von Transformationsregeln (UMLT), einer Webschnittstelle für den Aufruf von Modelltransformationen (mdWFS) sowie einem Zugriff auf die transformierten Geodaten innerhalb einer serviceorientierten Architektur. Die prototypische Implementierung dieses webbasierten Modell- und Datentransformationsdienstes erfolgte vor dem Hintergrund der EU-Richtlinie INSPIRE, welche einen europaweit einheitlichen Zugriff auf die Geodaten der Mitgliedstaaten erfordert. Die positiven Ergebnisse wie auch beim Prototyping erkannte Probleme werden im letzten Abschnitt erläutert. Sie sind z.T. Themen für weitergehende Forschung.

1 Einführung

Die Geoinformatik beschäftigt sich mit der Modellierung von Strukturen und Prozessen mit Raumbezug. Wesentlicher Nutzen von Geoinformationssystemen (GIS) ist es, Daten aus unterschiedlichsten Quellen aufgrund ihres Raumbezugs zusammenzuführen, um daraus neue Informationen zu gewinnen. Das Zusammenführen von Daten unterschiedlicher Quellsysteme in ein Zielsystem erfordert den Transfer von Daten. Dieser Datentransfer ist jedoch aufgrund der charakteristischen Eigenschaften von Geodaten (Geometrie als Nicht-Standard-Datentyp, Mehrfachrepräsentation von Objekten in unterschiedlichen Detaillierungsgraden, Unschärfe infolge nicht klar abgrenzbarer räumlicher Phänomene, etc.) sowie aufgrund heterogener und gewachsener Strukturen bei den Anbietern raumbezogener Daten nicht trivial. Beim Datentransfer ist deshalb oft notwendig, zusätzlich eine Modelltransformation durchzuführen.

[1] Institut für Geodäsie und Photogrammetrie, Gruppe Geoinformationssysteme, ETH Zürich, Wolfgang-Pauli-Strasse 15, CH-8093 Zürich, donaubauer@geod.baug.ethz.ch
[2] Fachgebiet Geoinformationssysteme, Technische Universität München, Arcisstr. 21, D-80333 München, tatjana.kutzner@bv.tum.de
[3] Institut für Geodäsie und Photogrammetrie, Gruppe Geoinformationssysteme, ETH Zürich, Wolfgang-Pauli-Strasse 15, CH-8093 Zürich, gnaegi@geod.baug.ethz.ch
[4] Institut für Geodäsie und Photogrammetrie, Gruppe Geoinformationssysteme, ETH Zürich, Wolfgang-Pauli-Strasse 15, CH-8093 Zürich, henrich@geod.baug.ethz.ch
[5] Fachgebiet Geoinformationssysteme, Technische Universität München, Arcisstr. 21, D-80333 München, astrid.fichtinger@bv.tum.de

Im Rahmen so genannter Geodateninfrastrukturen sollen Daten und Zugriffsdienste so harmonisiert werden, dass das Zusammenführen der Daten auf einfache Weise möglich ist. Die INSPIRE-Richtlinie der Europäischen Union [EU07] zum Aufbau einer europaweiten Geodateninfrastruktur verlangt von den Mitgliedstaaten, Geodaten über interoperable Geodienste bereitzustellen. Hierfür werden in den INSPIRE-Durchführungsbestimmungen standardisierte Schnittstellen erarbeitet, die einen einheitlichen Zugriff auf die Geodienste aller Mitgliedstaaten ermöglichen und die Daten in einheitlichen Datentransferformaten bereitstellen sollen. Damit für diese Daten auch auf semantischer Ebene, also auf Ebene der Datenmodelle Interoperabilität gegeben ist, werden im Rahmen der INSPIRE-Durchführungsbestimmungen zudem harmonisierte, europaweit einheitliche Datenmodelle (INSPIRE Data Specifications) für bestimmte Themen erstellt, z.B. für Verwaltungsgrenzen oder für Schutzgebiete. Aus Sicht der Anbieter derartiger Daten (z.B. Vermessungsverwaltungen der EU-Mitgliedstaaten) bedeutet dies, dass eine Transformation der Daten aus ihren eigenen Datenmodellen in die von der EU vorgegebenen Modelle durchzuführen ist. Die INSPIRE-Richtlinie sieht für diese Aufgabe sogenannte Transformationsdienste vor. Für derartige Dienste gibt es jedoch noch keine gültigen Standards [Mü08] und es herrscht Forschungsbedarf in diesem Bereich (vgl. z.B. [DSS07], [SGM08], [Le07], [Ba07], [OC08]). Genau diese Lücke schließt die in diesem Beitrag vorgestellte Methode zur webbasierten Modelltransformation.

2 Modellbasierter Datentransfer in der Geoinformatik

Die hier beschriebene Methode zur Modelltransformation stützt sich auf das modellbasierte Vorgehen für den Datentransfer, dessen Grundsatz und Anwendung im Folgenden erörtert werden.

Das modellbasierte Vorgehen erweist sich in der Geoinformatik als die zweckmäßigste Möglichkeit, Geodaten zwischen verschieden strukturierten Systemen bzw. Schnittstellen auszutauschen. Grundsatz und Vorteile des modellbasierten Vorgehens in diesem Bereich werden deutlich beim Vergleich mit anderen Möglichkeiten.

Die naheliegende Lösung ist, ein Programm zu schreiben, welches das Austauschformat des Quellsystems in das Austauschformat des Zielsystems transformiert. Gemäß Abbildung 1 begibt man sich damit auf die Ebene der Instanzen und programmiert von Hand eine *Instance Translation*. Wenn diese bilaterale Transformationsmethode aber auf n verschiedene Systeme angewendet wird, dann benötigt jedes dieser Systeme $n - 1$ verschiedene Export-Prozessoren. Für $n(n - 1)$ Programme ist jedoch der Entwicklungs- und Wartungsaufwand unverhältnismäßig groß.

Dieser Aufwand kann reduziert werden, wenn als Austauschformat ein Standardformat verwendet wird. Dann benötigt jedes System nur noch einen Import- und einen Exportprozessor. In der Geoinformatik wird solch ein Standardformat als formatbasierter bzw. formatgesteuerter Datentransferstandard bezeichnet. Wenn sich dieses Format ändert, erhält man einen neuen Standard. Um die Daten zu verstehen, die mit diesem Standardformat transferiert werden, müssen dessen Datenfelder beschrieben werden. Wird hierfür

natürliche Sprache eingesetzt, z.B. Deutsch oder Englisch, so kann dies zu vielen Missverständnissen führen. Die meisten Missverständnisse können jedoch ausgeschlossen werden, wenn die Beschreibung des Formats mit einer formalen Sprache wie einer Programmiersprache durchgeführt wird. Der weitere Vorteil dabei ist, dass diese Beschreibung auch durch einen Computer bearbeitet werden kann. Die Formatbeschreibung entspricht in Abbildung 1 dem *Physical Schema* oder Formatschema. Das Formatschema ist abhängig von einem speziellen Format, d.h. plattform-abhängig, und somit ein plattformspezifisches Modell (PSM).

Der nächste entscheidende Schritt liegt darin, nicht die Felder des Transferformats mittels formaler Sprache zu beschreiben, sondern die Struktur der zu transferierenden Daten und aus diesem so genannten Datenmodell auf konzeptioneller Ebene die Beschreibung eines entsprechenden Transferformats automatisch herzuleiten. Unterschiedlichste Datenmodelle können so mit derselben Sprache präzise beschrieben und deren entsprechenden Transferformate mit denselben Regeln hergeleitet werden. Dieser modellbasierte Standard (Sprache plus Format-, Herleitungs- oder Encodingregeln) muss auch nicht geändert werden, wenn in einem anderen Anwendungsgebiet eine andere Datenstruktur zu transferieren ist.

Die exakte Beschreibung der Datenstruktur heißt Datenmodell oder konzeptionelles Schema der Daten, *Conceptual Schema* in Abbildung 1. Dieses Datenmodell ist nicht mehr abhängig von einem bestimmten Format, also plattform-unabhängig, ein plattformunabhängiges Modell (PIM) [OMG03 und OMG05].

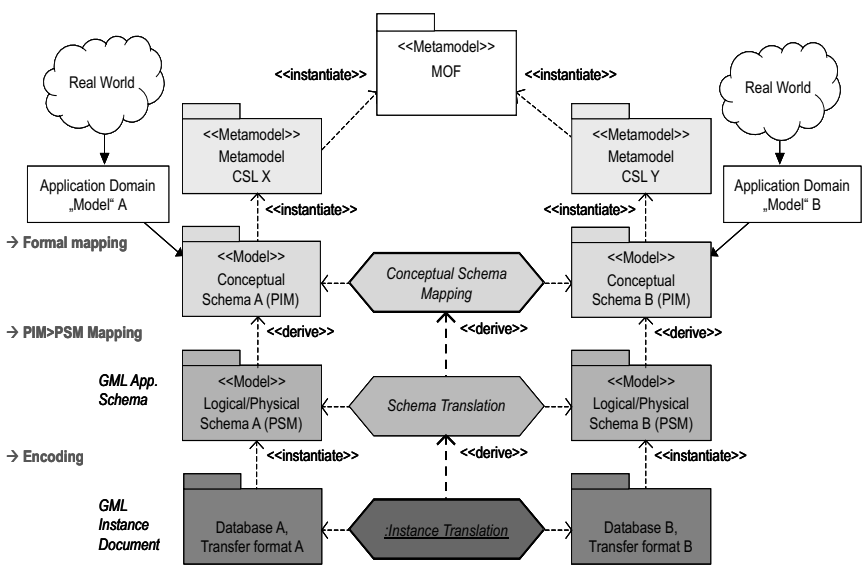


Abb. 1: Das modellbasierte Vorgehen [St07]

Für den modellbasierten Datentransfer muss nun einerseits eine Beschreibungssprache, die sogenannte konzeptionelle Schemasprache (CSL) festgelegt werden (*Metamodel* in Abbildung 1), wovon jedes Datenmodell eine Instanz ist. Andererseits müssen Regeln definiert werden, um die entsprechende Formatbeschreibung aus dem Datenmodell herleiten zu können. Diese Regeln für die Herleitung des Formatschemas sind in Abbildung 1 als *PIM>PSM Mapping* dargestellt.

Dieses modellbasierte Vorgehen dient nicht nur der Standardisierung des Datenaustauschs. Es ermöglicht auch, Werkzeuge zur Verfügung zu stellen für verschiedene Dienste, die aufgrund ihrer Unabhängigkeit von den betreffenden Datenstrukturen systemunabhängig sind. Beispiele für solche Werkzeuge sind:

a) *Nachhaltige Datensicherung:* Daten, die im Standardformat zusammen mit ihrem Datenmodell abgespeichert sind, bleiben jederzeit interpretierbar, zweifelsfrei und automatisch.

b) *Geometrische Datentypen:* Unabdingbar für die Modellierung von Anwendungsgebieten der Geoinformatik sind Datentypen für die wesentlichen geometrischen Objekte wie Punkte, Linien, Flächen, Pflasterungen etc. sowie die Möglichkeit, Einheiten und Koordinatensysteme zu definieren. Derartige Datentypen sind in CSL aus der allgemeinen IT wie UML meist nicht vorhanden und werden in CSL-Profilen definiert.

c) *Automatische Datenprüfung:* Es gibt verschiedene systemunabhängige Checker-Werkzeuge, welche Daten im Standardformat mit dem entsprechenden Datenmodell vergleichen und Inkonsistenzen melden, insbesondere auch geometrischer Art.

d) *Konzeptionelle Datenbankkonfiguration:* Ein konzeptionell entworfenes Datenmodell kann auf Konsistenz geprüft und unmittelbar in ein konsistentes Datenbankschema übersetzt und damit implementiert werden. Werden Daten aus dieser Datenbank exportiert und wird das Transferformat ebenfalls mittels Regeln aus dem konzeptionellen Schema hergeleitet, so ist sichergestellt, dass die Daten aus der Datenbank im Transferformat verlustfrei abgebildet werden können.

e) *Modelltransformation (Semantische Transformation):* Die Datentransformation von einem Quelldatenmodell in ein Zieldatenmodell erfolgt hierbei nicht durch Implementierung auf Datenebene sondern durch Konfiguration einer Modelltransformation auf (konzeptioneller) Modellebene mit Hilfe geeigneter vordefinierter Abbildungselemente.

Während es für die Beispiele a) bis d) bereits Standards und Werkzeuge gibt, müssen im Bereich der Modelltransformation noch grundlegende Forschungsarbeiten geleistet werden. Der Beitrag widmet sich dabei den folgenden Forschungsfragen:

- Wie kann eine Modelltransformation zwischen Datenmodellen für Geodaten auf konzeptioneller Ebene (PIM → PIM) formal sauber beschrieben werden?
- Gibt es dazu passende PIM → PSM-Abbildungen von den Modellen auf die Transferformate und von der Modellabbildung auf die Datentransformation?

- Kann der Datentransfer einschließlich der Modelltransformation als Web Service zur Verfügung gestellt werden?

3 Die Modellabbildungssprache UMLT

Zur Definition von Modelltransformationen auf Ebene der konzeptionellen Datenmodelle wird eine konzeptionelle Abbildungssprache (UMLT genannt) eingeführt. Diese Sprache muss verschiedenen Anforderungen bezüglich ihrer Verwendbarkeit genügen. So dürfen Transformationsschemata nicht nur von Systemen interpretierbar, sondern sie müssen auch für Menschen verständlich sein. Deshalb wurde sowohl ein UML 2-Metamodell als auch eine Syntax für eine *Human Useable Textual Notation* (HUTN) entwickelt. Die Transformationsschemata können dadurch in grafischer Form mit UML-Aktivitätsdiagrammen dargestellt, in textueller Form beschrieben (abgeleitet aus INTERLIS [KG06]) und in XMI (einem XML-basierten Transferformat für Metamodelle) kodiert werden. Einschlägige Normen und Standards der OMG (Object Management Group), des OGC (Open Geospatial Consortium) und der ISO (Internationale Organisation für Normung) wurden bei der Entwicklung berücksichtigt.

3.1 Konzept von UMLT

Bei der Entwicklung der Modellabbildungssprache UMLT wurden insbesondere zwei Formalismen der OMG untersucht. Zum einen die Meta-Object-Facility Query/Views/Transformations-Sprache (MOF-QVT) [OMG05]. Diese Sprache wurde für die Transformation von Metamodellen entwickelt, jedoch sind MOF-QVT-Modelle im Allgemeinen schwer verständlich. Der Standard ist komplex, da eigentlich drei Sprachen spezifiziert werden: Relations, Core und Operational. Darüber hinaus wurde der MOF-QVT-Standard in erster Linie für PIM-PSM-Abbildungen entwickelt und wird bei der Implementierung von Software eingesetzt, wie beispielsweise der Übersetzung von UML nach Java, jedoch weniger für PIM-PIM-Abbildungen.

Der andere untersuchte Formalismus ist das UML 2-Aktivitätsdiagramm. Dieses kann verwendet werden, um Transformationen mittels Aktivitäts-Sequenzen zu beschreiben. Eine klare Beschreibung der Semantik wie auch des Transferformates (XMI 2.1) ist in der Superstructure für UML-Modelle gegeben [OMG07]. UML 2-Modelle sind grundsätzlich leicht verständlich und es sind eine Vielzahl von Implementierungen und Open Source-APIs verfügbar.

Aufgrund dieser Überlegungen wurde das UML 2 Aktivitätsdiagramm als Ausgangspunkt gewählt. Ein wesentlicher Teil der Forschung bestand darin, diese bestehende Technologie um spezielle Abbildungsaktivitäten zu erweitern. Die konzeptionelle Abbildungssprache UMLT ist definiert durch eine unabhängige Erweiterung des UML 2-Metamodells. Unabhängig ist diese Erweiterung in dem Sinne, dass keine Elemente von

UML verändert werden, sondern neue Sprachelemente durch Vererbung hinzugefügt werden (vgl. nachfolgenden Abschnitt).

3.2 UMLT-Sprachelemente

Die Sprachelemente von UMLT sind eine Spezialisierung von UML 2-Aktivitäten [OMG07]. Es werden folgende Sprachelemente eingeführt, welche auch in Abbildung 2 dargestellt sind [St07]. Beispiele für die Verwendung dieser Sprachelemente sind in Kapitel 5 beschrieben:

a) *TransformationActivity:* Gliederungs- oder Strukturierungselement für semantische Transformationen. Umfasst alle Komponenten für eine gesamte Modelltransformation.

b) *StructuredTransformation:* Gliederungs- oder Strukturierungselement für semantische Transformationen. Entspricht einer Transformation bestehend aus Objekt- und Kontrollfluss.

c) *SelectionCriteria:* Auswahl von Input-Daten durch einen logischen Ausdruck.

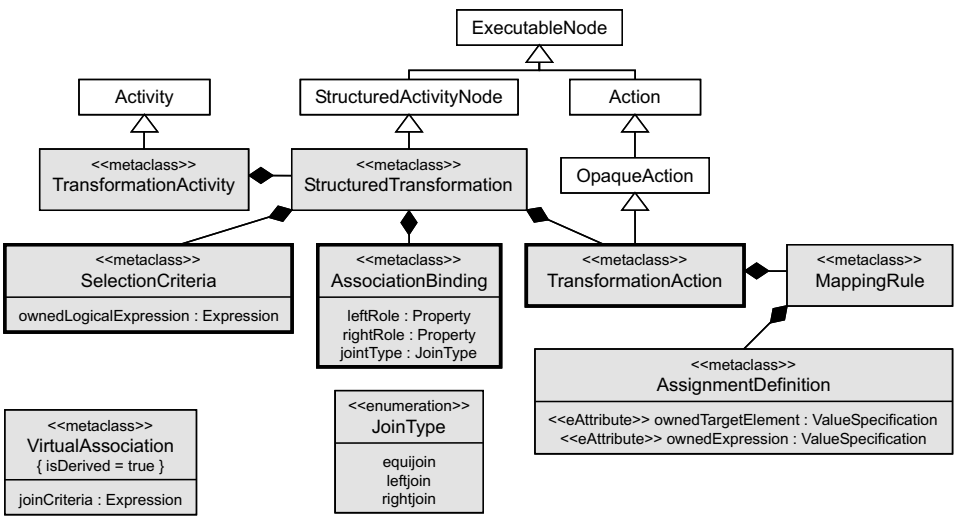


Abb. 2: Sprachelemente des UMLT-Metamodells (weiß: UML 2, grau: UMLT)

a) *VirtualAssociation:* Verwaltet Input-Daten, die nicht explizit durch eine Assoziation verknüpft sind, aber dennoch einen (engen) Zusammenhang aufweisen. Solche Input-Daten können beispielsweise Referenzattribute oder Fremdschlüsselattribute aufweisen, welche zur Laufzeit ausgewertet werden, um berechnete Relationen zu erhalten. Auf diese Weise können während einer semantischen Transformation solche Objekte virtuell assoziiert werden. Auf welche Weise Objektpaare erzeugt werden, kann mit einem logischen Ausdruck (joinCriteria) explizit defi-

niert werden. Darin unterscheidet sich die virtuelle Assoziation von einer herkömmlichen abgeleiteten Assoziation in UML 2. JoinCriteria können auch auf die geometrischen Attribute einer Klasse angewendet werden. So können z.B. aus einer Klasse „Flüsse“ und einer Klasse „Straßen“, die in einer virtuellen Assoziation über ein Join-Kriterium „crosses“ assoziiert sind, Objektpaare einander kreuzender Flüsse und Straßen ermittelt werden.

b) *TransformationAction:* Vererbung einer UML OpaqueAction. Bietet ein Aktivitätselement, welches nicht weiter strukturiert werden kann. Eine TransformationAction bildet den Behälter für MappingRules, die auf eine ausgewählte Menge von Objekten angewandt werden sollen.

c) *AssignmentDefinition:* Spricht Primitivtypen oder Ausdrücke als Wertzuweisung einer TransformationAction an. Dabei kann auch auf Funktionen zurückgegriffen werden. So lassen sich beispielsweise Attributwerte miteinander verknüpfen (siehe Anwendungsfall in Kapitel 5), Value Maps anwenden (z.B. der Wert „18.5“ des Attributs „Wassertemperatur“ im Quellmodell entspricht dem Wert „high“ im Zielmodell), oder Geometriedatentypen ineinander umwandeln (z.B. Typ „Fläche“ in Typ „Linie“).

d) *MappingRule:* Die eigentliche Objektabbildung. Wird als Komposition von AssignmentDefinitions spezifiziert und bildet eine TransformationAction.

e) *AssociationBinding:* Bei der Selektion der Input-Objekte kann spezifiziert werden, wie assoziierte Objekte während des Inputs ausgewählt werden sollen.

f) *JoinType:* Ein Aufzähltyp, um die Verbindungsregel einer AssociationBinding zu spezifizieren.

4 Webschnittstelle für die Modelltransformation

Ein Geo Web Service ist ein Dienst, der Funktionalitäten für die Nutzung von Daten mit Raumbezug, sog. Geodaten, bereitstellt, wie z.B. den Zugriff auf Geodaten sowie deren Verarbeitung und Präsentation, und der mittels Schnittstellen über das Internet ansprechbar ist [Do04]. Geo Web Services ermöglichen Interoperabilität, wofür jedoch standardisierte Schnittstellen zum Zugriff auf Geo Web Services notwendig sind. Diesem Ziel widmet sich das Open Geospatial Consortium (OGC) [OGC09] durch Entwicklung frei zugänglicher Schnittstellenspezifikationen für den Zugriff auf Geo Web Services. Nachfolgend wird zuerst die Spezifikation des OGC Web Feature Service erläutert. Anschließend wird diskutiert, inwieweit dieser Dienst eingesetzt werden kann, um die Modelltransformation mittels einer Webschnittstelle zur Verfügung zu stellen.

4.1 OGC Web Feature Service

Ein OGC Web Feature Service (WFS) [VP05] ermöglicht es einer Clientanwendung, über standardisierte Schnittstellen auf entfernte Geodaten zuzugreifen, siehe Abbildung 3. Die Datenhaltung kann dabei auf beliebige Weise erfolgen, z.B. in Datenbanken, was jedoch gegenüber dem Client verborgen bleibt. Die Kommunikation zwischen Client und WFS findet über HTTP statt durch Senden von Anfragen an den WFS. Die von einem Client angefragten Geodaten werden stets in GML (Geography Markup Language) kodiert als Geoobjekte (Features) an den Client ausgegeben. GML ist ein XML-basiertes Transferformat, welches die Formatbeschreibung, Übertragung und Speicherung geographischer Daten ermöglicht. Im Jahre 2007 ist GML von der Internationalen Organisation für Normung (ISO) als internationale Norm ISO 19136 Geographic information – Geography Markup Language (GML) [ISO07] verabschiedet worden.

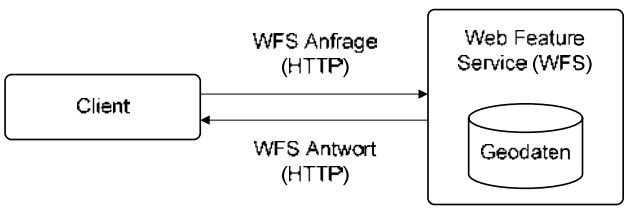


Abb. 3: Web Feature Service (Quelle: [VP05], verändert)

Für den Abruf von Geodaten werden in der WFS-Spezifikation drei Operationen definiert, die jeder WFS verpflichtend bereitstellen muss:

- *GetCapabilities*: Bei Aufruf dieser Operation erhält der Client ein XML-Dokument, in dem die Fähigkeiten des Dienstes durch sogenannte Service-Metadaten beschrieben werden. Zudem wird in diesem Dokument dem Client mitgeteilt, welche Features der Dienst ausgeben kann.
- *DescribeFeatureType:* Mit dieser Operation kann der Client die Formatbeschreibung einzelner Features abrufen.
- *GetFeature:* Über diese Operation werden schließlich die Geodaten als GML-Features in Form einer GML-FeatureCollection an den Client ausgegeben.

Neben dem Abruf von Geodaten (lesender Zugriff) bietet ein WFS auch die Möglichkeit, entfernte Daten zu ändern oder zu löschen (schreibender Zugriff), wofür weitere Operationen zur Verfügung stehen.

4.2 Model Driven Web Feature Service

Der WFS weist jedoch Einschränkungen bezüglich der semantischen Interoperabilität auf. So werden die Daten stets bezogen auf das Datenmodell des Quellsystems geliefert, welches sich i.d.R. vom Modell des Zielsystems unterscheidet. Modellbeschreibungen

auf konzeptioneller Ebene sind in einem WFS – wenn überhaupt vorhanden – für Zielsysteme nicht sichtbar und Modelltransformationen werden nicht unterstützt.

Aus diesem Grund wurde untersucht, inwieweit Modelltransformationen mittels eines Web Service zur Verfügung gestellt werden können. Damit sich solch ein Service bestmöglich in eine OGC Web Services Infrastruktur integrieren lässt, wurde als Grundlage die im vorigen Abschnitt beschriebene OGC-Spezifikation des WFS herangezogen und entsprechend den Anforderungen an Modelltransformationen erweitert. Dieser erweiterte WFS wird im Folgenden als mdWFS (model driven Web Feature Service) bezeichnet. Die Nutzung der mdWFS-Schnittstelle zur Konfiguration einer Modelltransformation und die Nutzung der WFS-Schnittstelle zum Abruf von transformierten Geodaten sind in Abbildung 4 dargestellt und wird im Folgenden näher erläutert.

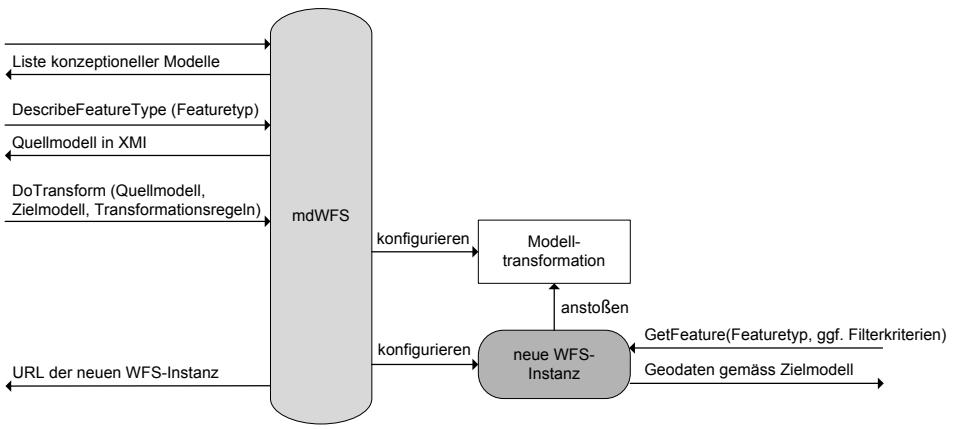


Abb. 4: Nutzung der mdWFS-Schnittstelle

Die WFS-Operation *DescribeFeatureType* wurde für den mdWFS dahingehend erweitert, dass der mdWFS konzeptionelle Datenmodelle im Format XMI bereitstellen kann [St09]. Darüber hinaus wurden für den mdWFS zwei neue Operationen definiert:

- *GetCapabilities:* In Abwandlung der WFS-Spezifikation des OGC enthält die Antwort auf eine GetCapabilities-Anfrage keine Liste der verfügbaren Features, sondern eine Liste mit den Namen aller konzeptionellen Modelle, auf die die jeweilige mdWFS-Instanz einen Zugriff erlaubt.
- *DoTransform:* Durch Aufruf dieser Operation werden die in UMLT auf konzeptioneller Ebene beschriebenen Transformationsregeln auf ein plattform-spezifisches Modell übertragen, um letztendlich Daten transformieren zu können. Jeder erfolgreiche Aufruf der Operation resultiert in einer neuen WFS-Instanz, über die die Daten entsprechend dem Zielschema abrufbar sind, siehe Abbildung 4. Dies ist eine grundlegende Designentscheidung, deren großer Vorteil es ist, dass dadurch ein herkömmlicher WFS-Client ohne jegliche Erweiterung für den Zugriff auf die Daten verwendet werden kann. Aufrufparameter der Operation sind: das Quellsche-

ma bzw. die DescribeFeatureType-URL jener WFS-Instanz, die das Quellschema repräsentiert, das Zielschema im Format XMI, die UMLT-Transformationsregeln im Format XMI sowie optional die Wahl von Regeln für das Encoding der Daten.

Die Antwort des mdWFS auf eine DoTransform-Anfrage besteht aus einer Statusmeldung sowie aus einer GetCapabilities-Request-URL als Identifikator für die neu angelegte WFS-Instanz. Ein herkömmlicher WFS-Client kann diese URL nutzen, um auf Funktionalität und Daten dieser WFS-Instanz zuzugreifen. Bei jedem Abruf von Daten über die WFS-GetFeature-Operation wird die Modelltransformation angestoßen und die Daten werden in das Zielmodell transformiert, bevor sie ausgeliefert werden.

5 Anwendungsfall

Zur Demonstration des hier beschriebenen Konzepts wird ein Prototyp entwickelt, welcher eine Modelltransformation zwischen dem deutschen AAA-Modell (Quellschema) und dem von der EU vorgegebenen INSPIRE-Modell (Zielschema) durchführen kann. Als Ausschnitt aus dem Quellmodell wird der Themenbereich *Schutzgebiete* gewählt, welcher in das entsprechende Zielmodell *ProtectedSites* transformiert wird.

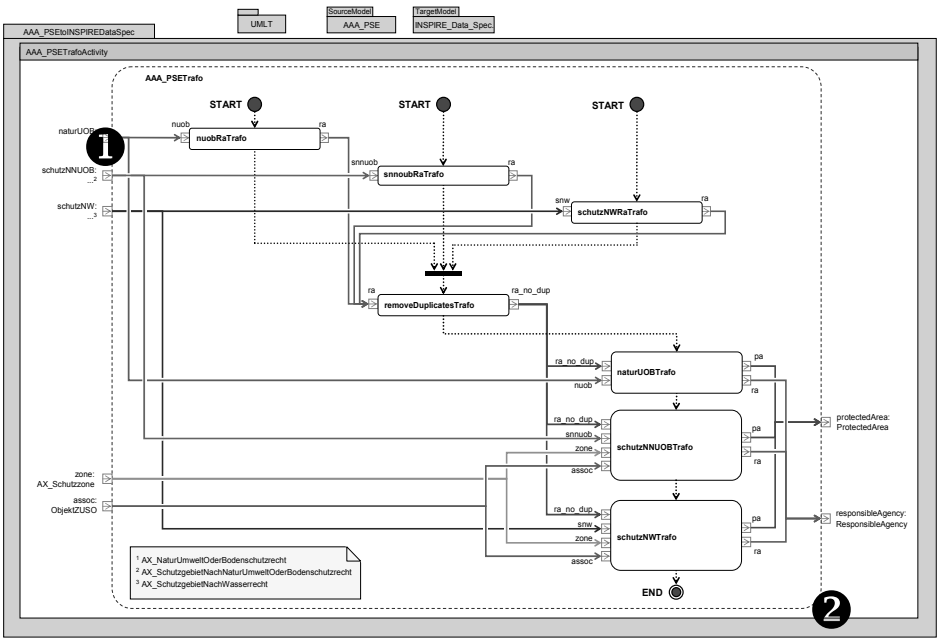


Abb. 5: Darstellung der Modelltransformation in UMLT (Übersicht)

Für die Modelltransformation wurden Transformationsregeln zwischen Quell- und Zielmodell auf konzeptioneller Ebene definiert. Abbildung 5 zeigt in einer Übersicht die Transformation der Schutzgebiete aus dem AAA-Modell in das INSPIRE-Modell unter

Verwendung von UMLT.

Das Package AAA_PSEtoINSPIREDataSpec beinhaltet die gesamte Transformation als *TransformationActivity* AAA_PSETrafoActivity (entspricht dem Punkt a) aus Kapitel 3.2). Zusätzlich wird der Zugriff auf die Packages UMLT, das das Metamodell von UMLT enthält, das Quellmodell und das Zielmodell ermöglicht (Detail ❶).

Innerhalb der TransformationActivity wird die *StructuredTransformation* AAA_PSETrafo aufgebaut (Punkt b) aus Kapitel 3.2), in welcher der Objektfluss von links nach rechts (*Data Flow*, durchgehende Linien) und der Kontrollfluss von oben nach unten (*Control Flow*, gepunktete Linien) festgelegt werden. An den Input-Pins der StructuredTransformation (z.B. naturUOB, schutzNNUOB; siehe Detail ❶) liegen die Objekte aus den benötigten Klassen des Quellmodelles an. Über logische Ausdrücke werden die Objekte gefiltert (*SelectionCriteria*, nicht dargestellt, Punkt c) aus Kapitel 3.2) und den einzelnen *TransformationActions* zugeführt (Punkt e) aus Kapitel 3.2). Der Kontrollfluss steuert dabei die Reihenfolge der auszuführenden TransformationActions.

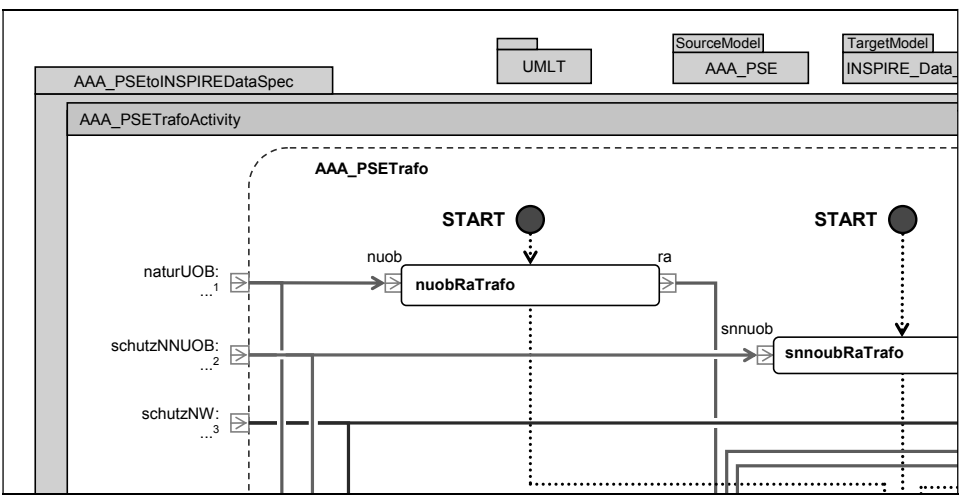


Abb. 6: Detail ❶: Ausschnitt aus der Transformation mit den Startpunkten des Control Flows (START), TransformationActions sowie Input- und Output-Pins des Data Flows

Durch die TransformationAction werden die am Input-Pin anliegenden Objekte sequenziell verarbeitet und anschließend an den Output-Pin weitergeleitet. Die Verarbeitung erfolgt aufgrund der *MappingRules* (Punkt g) aus Kapitel 3.2) für jedes einzelne Objekt. Abbildung 7 zeigt den Einblick in die TransformationAction nuobRaTrafo aus Detail ❶ mit den entsprechenden MappingRules, die ihrerseits aus *AssignmentDefinitions* (Punkt f) aus Kapitel 3.2) aufgebaut sind. Die Notation der MappingRules ist an diejenige von INTERLIS angelehnt. Der Pfeil-Operator „->“ dient der Angabe eines Pfades, welcher zum gewünschten Attribut eines Objektes führt. Lesebeispiel: Die Funktion `String.concatenator` erzeugt einen neuen String aus den zwei Attributwerten

`land` und `stelle` des Quellmodells und weist diesen dem Attribut `localID` aus dem Zielmodell zu.

nuobRaTrafo

```
ra->objectIdentifier->localID := String.concatenator(nuob->
      ausfuehrendeStelle->land, nuob->ausfuehrendeStelle->stelle);

ra->objectIdentifier->namespace := "de";

ra->objectIdentifier->versionId := "Unpopulated";

ra->responsibleAgencyName := "Unpopulated";
```

Abb. 7: Eine der sieben TransformationActions des Beispiels

Nach dem Durchlaufen der verschiedenen TransformationActions werden die umgewandelten Objekte am Ende der Transformation entsprechend den Klassen des Zielmodells ausgegeben (Detail ❷).

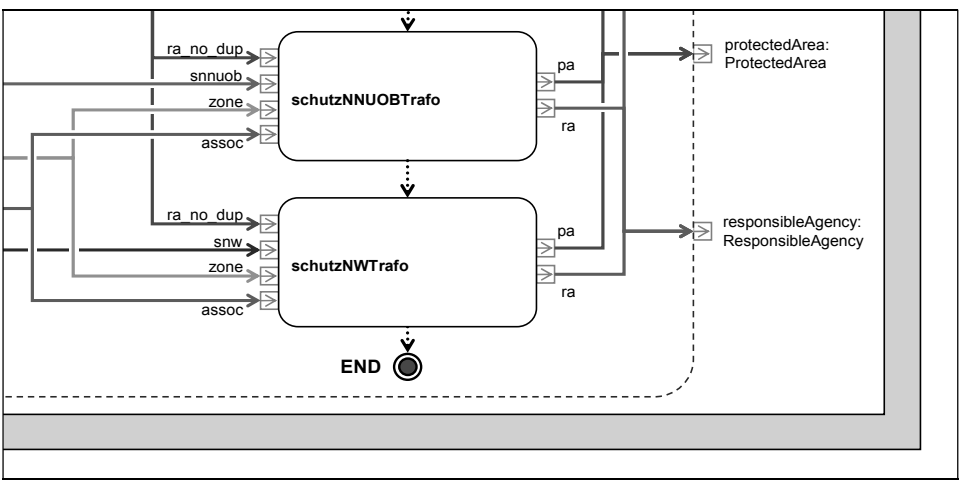


Abb. 8: Detail ❷: Ausschnitt aus der Transformation mit dem Endpunkt des Control Flows (END) sowie den Output-Pins der gesamten Transformation

Die in UMLT beschriebenen Transformationsregeln sind nun auf ein plattformspezifisches Modell zu übertragen, um letztendlich Daten transformieren zu können. Wünschenswert ist es, die Transformation zwischen Quell- und Zielmodell *on-the-fly* durchzuführen, d.h., die Transformation wird jedes Mal dann ausgeführt, wenn ein Benutzer Daten über die WFS-Schnittstelle abruft. Dies hat den Vorteil, dass die transformierten Daten immer aktuell sind, da Kopien der Daten bezogen auf das Zielmodell nicht abgespeichert werden.

Da eine komplette Eigenentwicklung eines solchen Transformationsmoduls sehr aufwändig ist, wurde entschieden, eine bereits vorhandene Softwarelösung in den Prototyp zu integrieren. Hierfür wurde die Spatial ETL Software FME (Feature Manipulation Engine) [FME09] der kanadischen Firma Safe Software Inc. ausgewählt, welche die Möglichkeit bietet, Mappings zwischen Daten auf Ebene des Datenformats zu definieren und entsprechende Transformationen durchzuführen. Diese Software musste jedoch dahingehend erweitert werden, dass sie sowohl konzeptionelle Datenmodelle als auch die in UMLT formulierten Modelltransformationsregeln lesen und interpretieren kann.

6 Fazit und Ausblick

Für das Schlüsselproblem der – zumindest im Geoinformatikbereich – neuen Methode der Modelltransformation mit Definition der Transformation von Quellmodell nach Zielmodell auf konzeptioneller Ebene sowie der automatisch daraus folgenden Datentransformation, konnte eine erfolgreiche Lösung vorgestellt werden, die zudem in eine serviceorientierte Architektur integrierbar ist. Diese webbasierte Modelltransformation besteht aus folgenden Komponenten (in der Reihenfolge, in der sie zum Einsatz kommen):

- Web-Schnittstellen für den Zugriff auf die Liste der verfügbaren konzeptionellen Quelldatenmodelle und für die Bereitstellung des vom Nutzer gewünschten Quelldatenmodells. Das Zieldatenmodell ist beim Nutzer vorhanden.
- Sprache (UMLT) zur Definition der Modelltransformation auf konzeptioneller Ebene mit automatisch folgender Datentransformation auf physischer (Format-) Ebene.
- Web-Schnittstelle zur Übergabe von Quellmodell, Zielmodell und Modelltransformationsregeln sowie Ausführung der Datentranformation und Rückgabe der Web-Adresse eines Dienstes für den Zugriff auf die transformierten Daten entsprechend des Zieldatenmodells.

Alle drei Komponenten wurden erfolgreich in einem Prototyp implementiert und anhand eines Anwendungsfalls evaluiert. Dieses Ergebnis kann als wesentlicher Fortschritt für den Geodatentransfer mit Modelltransformation und dessen Verfügbarkeit im Web gewertet werden.

Die in Kapitel 2 genannten zentralen Forschungsfragen können somit wie folgt beantwortet werden:

1. Es ist möglich, eine Modelltransformation zwischen Datenmodellen für Geodaten auf konzeptioneller Ebene (PIM → PIM) formal sauber zu beschreiben. Die hier beschriebene Sprache UMLT wurde durch wesentliche Erweiterung einer vorhandenen Technologie (UML 2 Aktivitätsdiagramm) entwickelt. Für die Evaluierung des Prototyps wurden in UML modellierte Datenmodelle aus Deutschland, der Schweiz und von INSPIRE als Quell- bzw. Zielmodelle bei der Modelltransforma-

tion verwendet. Dabei zeigte sich, dass in allen drei Fällen ein unterschiedliches UML-Profil zum Einsatz kommt. Dies bedeutet, dass nicht nur unterschiedliche Datenstrukturen umzubauen sind (Datentransformation, PIM→PIM-Transformation), sondern auch die verwendeten Datenbeschreibungssprachen verschieden sind und unterschiedliche Metamodelle besitzen. Außerdem zeigte sich, dass die Datenmodelle aus der Praxis zum Teil unnötig komplex sind und Mehrdeutigkeiten enthalten.

2. Die zweite wesentliche Komponente der hier beschriebenen Methode der Modelltransformation ist die PIM→PSM-Abbildung. Sie wird sowohl für den automatischen Übergang von der konzeptionellen Ebene des Datenmodells zur physischen Ebene von Format bzw. Daten als auch von der konzeptionellen Ebene der Modelltransformationsregeln zur physischen Ebene der Datentransformation verwendet. Dazu braucht es eindeutige Kodierungsregeln. Bei der Evaluierung des Prototyps stellte sich heraus, dass für die drei untersuchten Modelle und die zugeordneten Transferformate jeweils andere Codierungsregeln gelten, was für die Umsetzung der Methode ein Problem darstellt. Im Prototyp wurde dieses Problem gelöst, in dem alle Modelle, teilwiese durch aufwändige Nachmodellierung, auf das Metamodell der Schweizer Norm INTERLIS [KG06] gebracht wurden. INTERLIS definiert eindeutige PIM→PSM-Abbildungsregeln (Datenmodell→Transferformat). Für die PIM→PSM-Abbildung der Modelltransformationsregeln konnte durch die Implementierung des Prototyps ein Machbarkeitsnachweis erbracht werden. Da diese Regeln von den internen Strukturen der jeweiligen Transformationssoftware abhängig sind, wurde auf sie in diesem Beitrag nicht näher eingegangen.

3. Es ist möglich, Datentransfer einschließlich Modelltransformation als Web Service zur Verfügung zu stellen (vgl. Kapitel 4). Hierzu mussten vorhandene Standards und Technologien (WFS, Spatial ETL Software FME) um wesentliche Komponenten erweitert und die Realisierbarkeit durch Implementierung eines Prototyps (vgl. Kapitel 5) nachgewiesen werden.

Die hier beschriebenen Probleme wurden zum Anlass genommen, 2010 im Rahmen einer Studie zu untersuchen, ob und wie die Modelltransformation zukünftig unabhängig von speziellen UML-Profilen und Kodierungsregeln eingesetzt werden kann.

Dank

Der Dank für die Finanzierung des Forschungsprojekts mdWFS gilt den Auftraggebern – dem Deutschen Bundesamt für Kartographie und Geodäsie (BKG) sowie dem Schweizer Bundesamt für Landestopographie swisstopo.

Literaturverzeichnis

[AdV06] Arbeitsgemeinschaft der Vermessungsverwaltungen der Länder der Bundesrepublik

Deutschland (AdV) (Hrsg.): Dokumentation zur Modellierung der Geoinformationen des amtlichen Vermessungswesens (GeoInfoDok) - Version 5.1 - 31.03.2006

[Ba07] Balley, S.: Aide à la restructuration de données géographiques sur le Web. Dissertation an der Université de Marne-la-Vallée, Frankreich, 2007.

[Do04] Donaubauer, A.: Interoperable Nutzung verteilter Geodatenbanken mittels standardisierter Geo Web Services, Dissertation, Technische Universität München, 2004.

[DSS07] Donaubauer, A.; Straub, F.; Schilcher, M.: mdWFS: A Concept of Web-enabling Semantic Transformation. Proceedings of the 10th AGILE Conference on Geographic Information Science, Aalborg, 2007.

[EU07] Europäisches Parlament und Europäischer Rat (Hrsg.): Richtlinie 2007/2/EG des Europäischen Parlaments und des Rates vom 14. März 2007 zur Schaffung einer Geodateninfrastruktur in der Europäischen Gemeinschaft (INSPIRE). Amtsblatt der Europäischen Union Nr. 108/1 vom 25.4.2007.

[FME09] Safe Software Inc.: FME, http://www.safe.com/products/overview.php

[IDT08] INSPIRE Drafting Team Data Specifications: D2.7: Guidelines for the encoding of spatial data, Version 3.0.

[ISO07] Norm ISO 19136:2007, Geographic Information – Geography Markup Language (GML), 2007.

[KG06] KOGIS(Hrsg.): INTERLIS Referenzhandbuch, Ausgabe vom 13.04.2006 (deutsch). Bundesamt für Landestopographie. Wabern, Schweiz.

[Le07] Lehto, L.: Real-time content transformations in a web service-based delivery architecture for geographic information. Dissertation am Finnischen Geodätischen Institut, 2007.

[Mü08] Müller, M.: INSPIRE – Bedeutung für Dienste und Portale. In: Runder Tisch GIS e.V. (Hrsg.): Tagungsband zum 13. Münchner Fortbildungsseminar Geoinformationssysteme, München, 2008.

[OC08] ORCHESTRA Consortium (Hrsg.): Implementation Specification of the Translating Feature Access Service, Version 0.4/1.3, 2008.

[OGC09] Open Geospatial Consortium: http://www.opengeospatial.org/

[OMG03] Object Management Group (Hrsg.): MDA Guide Version 1.0.1. OMG specification omg/2003-06-01, 2003.

[OMG05] Object Management Group (Hrsg.): MOF 2.0 Query/Views/Transformations Specification. OMG specification ptc/05-11-01, 2005.

[OMG07] Object Management Group (Hrsg.): UML Unified Modeling Language: Superstructure, version 2.1.1. OMG specification formal/2007-02-05, 2007.

[SGM08] Staub, P.; Gnägi, H.R.; Morf, A.: Semantic Interoperability through the Definition of Conceptual Model Transformations. Transactions in GIS 12(2): 193-207, 2008.

[St07] Staub, P.: A Model-Driven Web Feature Service for Enhanced Semantic Interoperability. OSGeo Journal 1(3):38-43, 2007.

[St09] Staub, P.: Über das Potenzial und die Grenzen der semantischen Interoperabilität von

Geodaten. Dissertation, Institut für Geodäsie und Photogrammetrie, ETH Zürich, 2009.

[VP05] Vretanos, Panagiotis A.: Web Feature Service Implementation Specification, Version 1.1.0, Open Geospatial Consortium, 2005.

A Change Metamodel for the Evolution of MOF-Based Metamodels

Erik Burger[1] and Boris Gruschko[2]

Abstract: The evolution of software systems often produces incompatibilities with existing data and applications. To prevent incompatibilities, changes have to be well-planned, and developers should know the impact of changes on a software system. This consideration also applies to the field of model-driven development, where changes occur with the modification of the underlying metamodels. Models that are instantiated from an earlier metamodel version may not be valid instances of the new version of a metamodel. In contrast to other metamodeling standards like the Eclipse Modeling Framework (EMF), no classification of metamodel changes has been performed yet for the Meta Object Facility (MOF).

The contribution of this paper is the evaluation of the impact of metamodel changes on models. For the formalisation of changes to MOF-based metamodels, a *Change Metamodel* is introduced to describe the transformation of one version of a metamodel to another. The changes are then classifed by their impact on the compatibility to existing model data. The classification is formalised using OCL constraints. The *Change Metamodel* and the change classifications presented in this paper lay the foundation for the implemention of a mechanism that allows metamodel editors to estimate the impact of metamodel changes semi-automatically.

1 Introduction

The divide between the developer and user of model driven tools becomes wider with broader adoption of model driven techniques in software engineering. While providing higher quality tools and increasing the productivity of their application, this divide also introduces the typical development cycles observed in the evolution of any software development tool. The tool developer provides the user with a discrete version of his product, and the user produces content in accordance with the version of the tool available to him. The problem of evolution does not arise if the user applies the same version of the tool throughout a project's lifecycle, or if the user and the tool developer are the same person. In the first case, the non-relevance of the evolution problem is obvious. In the second case the problem does not arise, because the tool user has intimate understanding of tool's inner workings and evolution impact of his doing on the persisted content. In recent years, a number of toolkits for the development of model driven tools have emerged, simplifying the task of tool creation and modification. This development escalates the problem of tool evolution.

[1] Chair for Software Design and Quality, Karlsruhe Institute of Technology, Germany, erik.burger@kit.edu
[2] SAP AG, Walldorf, Germany, boris.gruschko@sap.com

The most visible symptom of the evolution problem in model driven tools is the breaking of model content due to metamodel evolution. This problem arises when the developer of the tool changes the underlying metamodel, rendering the existing content (created by an earlier version of the tool) incompatible to the new metamodel version.

While the problem of metamodel evolution can be handled via active migration, most of the cases could be handled in an automatic manner. Most of the available toolkits for the development of model driven tools have an underlying meta-metamodel to which the metamodels have to adhere. The restrictions imposed by the meta-metamodel can be used to derive a catalogue of all possible metamodel changes and their classification into changes which could lead to the breakage of model content and those which can be introduced in an additive manner. A classification of this kind has been performed for the Ecore metamodel [BGGK07], but is still missing for MOF [MOF05].

In this paper we present an classification scheme of metamodel changes. Furthermore, we present an approach to the attachment of metamodel changes to the actual metamodels (*Change Metamodel*). Another contribution of this paper is the classification of all possible metamodel changes of a MOF 1.4 metamodel according to the proposed classification scheme. This classification also accounts for sequences of metamodel changes and is formalised using OCL constraints.

The work has been performed in scope of the MOIN (MOdeling INfrastructure) project at the SAP AG. MOIN is a MOF 1.4 based repository. [AHK07]

The structure of this paper is as follows: First, an overview of the foundations of metamodel evolution is given. Then, the *Change Metamodel* is introduced and its structure explained. The most interesting cases for the classification of metamodel changes are discussed by example. The complete classification of MOF metamodel change types is presented as an overview table in section 4. Afterwards, the whole process of metamodel evolution description and classification is shown within a common example. The assumptions and limitations of our approach are then mentioned, together with related work before the paper concludes.

2 Foundations

2.1 Difference of Metamodels

The process of metamodel evolution requires to deal with two versions of a metamodel. The difference between two metamodels can be described by a sequence of elementary change operations [AP03]. These operations contain addition and deletion of elements and links, and the modification of element properties. In general, there are two approaches of determining the set of operations describing a change: Either by tracing of single changes or by direct comparison. [Gir06]

In the first approach, tool support is needed in order to record single changes during the editing of the metamodel. The result is a sequence of change operations that reference the original metamodel. Since changes can revert each other, this sequence is not necessarily minimal unless these redundancies are eliminated. Furthermore, if users edit a metamodel through a tool, the operations tool must be matched with the elementary operations mentioned above.

In the second approach, if two metamodels are compared directly, two prerequisites have to be met: Firstly, the underlying infrastructure must support loading two versions of the same metamodel. On the one hand, the matching of elements is simplified by the fact that every element has a unique MOF-ID; on the other hand, this rises the problem that the infrastructure has to deal with different elements that have the same unique identifier. Secondly, an algorithm must be chosen hat calculates a sequence of elementary changes.

2.2 Metamodel Evolution

The process of metamodel evolution can make models inconsistent with the new version of the metamodel. These inconsistencies must be resolved by migrating the models, which is a process that cannot be fully automated. In [GKP07], Gruschko et al. introduce a classification scheme that categorises changes to Ecore-based metamodels into three classes, considering the impact on metamodel instances. We adapt this categorisation for MOF-based metamodels, resulting in the following three *change severities*:

non-breaking < *breaking and resolvable* < *breaking and not resolvable*

- A *non-breaking* change does not require any adaptation of existing models, which is mostly true for additive changes to the metamodel.
- For *breaking and resolvable* changes, an algorithm can be defined to migrate existing instances to the new metamodel version.
- If a *breaking and not resolvable* change occurs, manual interaction is required to make existing models conform to the new metamodel, if possible at all.

When talking about *conformance* of a model to a metamodel, we mean *instantiation conformance* as defined by Steel [SJ04].

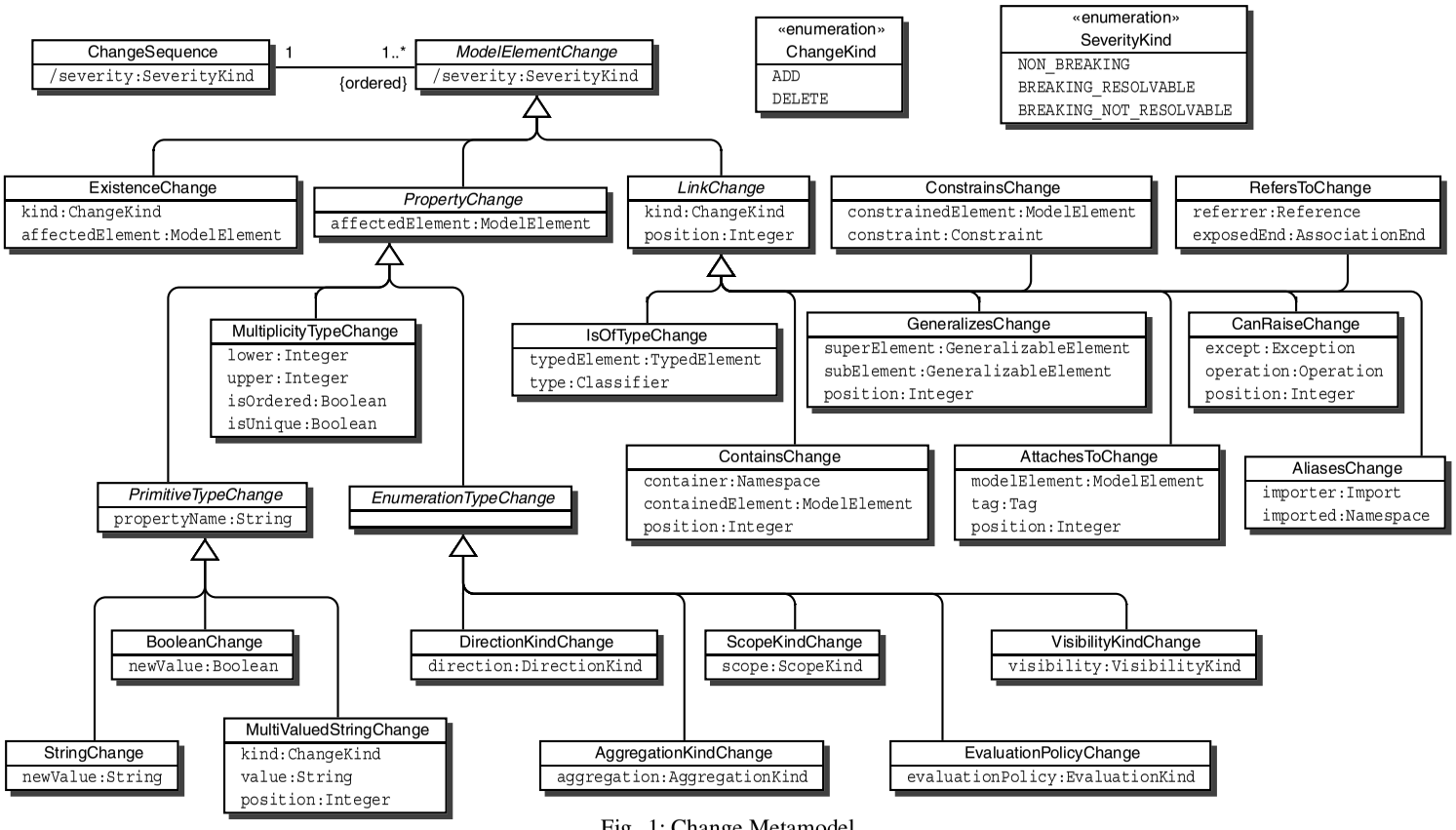


Fig.. 1: Change Metamodel

3 The Change Metamodel

3.1 Definition

For the description of metamodel changes, we introduce a *Change Metamodel* (see figure 1). Instances of this metamodel describe an actual change to a metamodel as a sequence of single change operations, contained in a *ChangeSequence* element. As mentioned in subsection 2.1, all changes in a metamodel can be expressed as a sequence of either additions or deletions of elements and links, or modifications of a property. The *Change Metamodel* is also based on this assumption; the three base classes of the change metamodel cover the addition or deletion of elements *(ExistenceChange)*, the modification of properties *(PropertyChange)* and the deletion or addition of links *(LinkChange)*. These single change types will be described and classified in the following subsections.

3.1.1 ExistenceChange

This change type describes the addition or deletion of an element in the metamodel, i.e. an instance of a MOF class. This covers e.g. classes, attributes, associations etc. In the case of deletion, the *ExistenceChange* instance references an element in the old metamodel; in the case of addition, it references a new element which must be contained in the current Change Metamodel instance.

3.1.2 PropertyChange

A property of a metamodel element represents an attribute in the MOF class of which the element is an instance. This could, for example, be the name of an element or the cardinality of an association end.

Since in the MOF model itself, the attributes of the classes are only typed with *DataType*s (and not with classes), we can specify an actual class in the *Change Metamodel* for every property type. Furthermore, the *propertyName* field is only needed for the *PrimitiveTypeChange*, and not for multiplicity or enumeration types, since there is only one attribute of these types in each MOF class.

3.1.3 LinkChange

This class represents all changes in features that are associations in the MOF model. This includes containment, inheritance (generalisation) and typing. Of these, some are not represented as (visible) associations in a model diagram. However, since they are associations in the MOF model, links exist in the metamodels for these concepts, and thus they are covered by this class. For every association that is part of the MOF model, there is a class in the *Change Metamodel*. They contain each two attributes for the ends of the association,

which are named after the roles in the MOF model. Changing a link is always expressed as either a deletion or addition. This approach is common practise in other fields, e.g. database schema migration [Mon93, p. 42].

For associations that are ordered, the field *position* describes the position of a newly added element. For deletions, this value is ignored.

3.2 Classification

The classification of a change is expressed by the derived attribute *severity* in the respective classes of the *Change Metamodel*. The semantics of this attribute are formally described by OCL constraints that cover all the cases shown in the table in section 4. A complete listing of all constraints can be found in [Bur08].

3.2.1 Overall Severity

Since changes are contained in change sequences, not only the severity of a single change has to be regarded, but also the overall severity of a sequence of changes. Intuitively, the overall severity can be determined by finding the maximum (according to the order presented in subsection 2.2) of the severities of all singular changes in the change sequence.

However, if the change severity of a single change is only based on the effect of that single change, the overall severity is only equal to the maximum of the sequence's severities in trivial cases. If a sequence of changes is applied to a metamodel, it is possible that a combination of changes has a lower severity than any of the single changes. A simple example is the consequent addition and deletion of the same element; the latter change reverts the first one, and there is no effect on the metamodel at all.

As a consequence of this, the OCL constraints that describe the severity of a *ModelElementChange* also account for other changes in the same *ChangeSequence*. The constraints express the severity of a single change in the context of the complete change. If a change sequence only contains one element, the constraint expresses the severity for a singular change. Thus, the maximum severity yields the correct result for the overall severity of a sequence of changes.

3.2.2 Interesting Cases by Example

In this section, we will show two of the more difficult cases, where the determination of change severities is non-trivial.

Deletion of a Class (cf. Figure 2) If classes are deleted in the metamodel, the corresponding M1 instances must also be deleted. This change is normally resolvable, but can

be *breaking and not resolvable* if a situation similar to the one shown in Figure 2 occurs: The instance of *Type2* has a link to an instance of *Subtype1*. If *Subtype1* is deleted, the association still exists in the metamodel, since it is connected to the superclass *Type1* of the deleted class. As a consequence of the deletion of *Subtype1*, all its instances and the links to these instances are also deleted, leaving only the instance of *Type2*. Now the model data is invalid, since there is no link to an instance of *Type1*, which is required by the minimum cardinality of *1* in the metamodel.

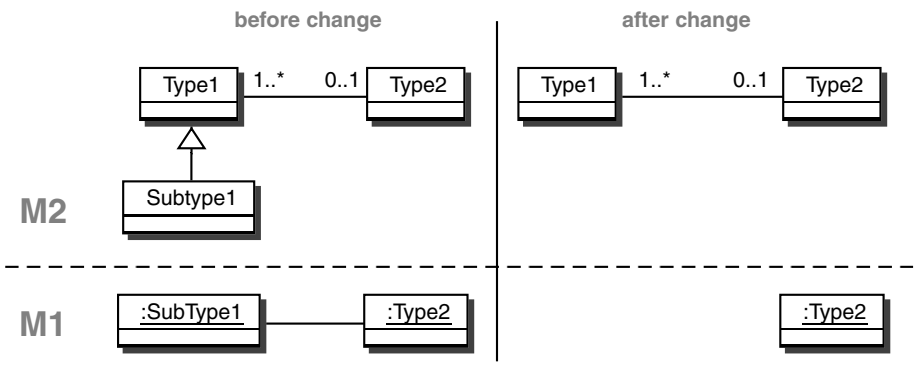


Fig.. 2: Example: Deletion of a class makes M1 data invalid

Association end moved to superclass (cf. Figure 3) If one end of an association is moved to another type, the change consists of a type change of the affected *AssociationEnd*. If the situation is similar to Figure 3, existing M1 data will become invalid. In the example, *EndB* is changed to a supertype. In the M1 data, the instances of *Type1* and *SubType2* will still be valid after the change, since the association still exists. Only the instance of *Type2* is invalid since there is no association to an instance of *Type1* despite the minimum cardinality *1* of *EndA*. This makes the change *breaking and not resolvable*.

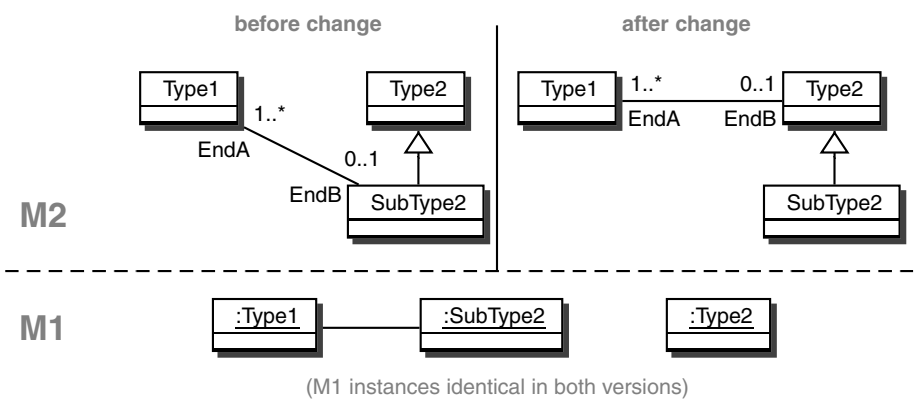


Fig.. 3: Example: Type change on an association end

4 Classification of MOF Metamodel Changes

The table shows the change severities of the single cases, grouped by the classes of the *Change Metamodel*, which can take the values *non-breaking* (nb), *breaking and resolvable* (br) and *breaking and not resolvable* (bn). For a comprehensive explanation, examples and the OCL constraints of all special cases, please refer to [Bur08].

MOF change type		nb	br	bn	condition
ExistenceChange					
`Class`	add	✓			additive
	delete			✓	supertype has mandatory association end
			✓		other cases
`Attribute`	add			✓	minimum cardinality > 0, no initializer
			✓		minimum cardinality > 0, initializer exists
		✓			minimum cardinality $= 0$
	delete		✓		
`Association`	add			✓	minimum cardinality > 0
		✓			minimum cardinality > 0, no instances
		✓			minimum cardinality $= 0$
	delete		✓		
`AssociationEnd`					same as for `Association`
`Reference`	add	✓			
	delete	✓			
`Package`	add	✓			
	delete		✓		contents resolvable
				✓	contents not resolvable
`Import`	add	✓			
	delete		✓		elements resolvable otherwise
				✓	elements not resolvable
`Tag`	add	✓			
	delete	✓			
`Constraint`	add			✓	
	delete	✓			
`Operation`	add/delete	✓			
`Exception`	add/delete	✓			
`Parameter`	add/delete	✓			
`DataType`	add/delete	✓			
`StructureField`	add/delete	✓			

PropertyChange

ModelElement	name		✓		refactoring, unique ids
	annotation	✓			
Generalizable Element	isRoot	✓			
	isLeaf	✓			
	isAbstract	✓			true → false
			✓		false → true
	visibility	✓			increase
				✓	decrease
Association	isDerived	✓			true → false
				✓	false → true
Class	isSingleton	✓			true → false
				✓	false → true
Constraint	expression			✓	
	language			✓	
	evaluationPolicy	✓			
Feature	scope			✓	instance_level → classifier_level
		✓			classifier_level → instance_level
	visibility	✓			increase
				✓	decrease
StructuralFeature	multiplicity	✓			widening
				✓	narrowing
	isChangeable	✓			
Attribute	isDerived	✓			true → false
				✓	false → true
Operation	isQuery	✓			
Constant	value	✓			
Parameter	direction	✓			
	multiplicity	✓			
AssociationEnd	isNavigable	✓			
	aggregation	✓			composite → none
				✓	none → composite, composition closure rule violated
		✓			closure rule not violated
	multiplicity	✓			widening
				✓	narrowing
	isChangeable	✓			
EnumerationType	labels	✓			addition
				✓	delete/modify

LinkChange

`Contains`		✓			not mandatory; to supertype
			✓		not mandatory; to other type
		✓			mandatory; to new or abstract supertype
				✓	mandatory; to other type
`Generalizes`	add			✓	supertype contains mandatory features or associations
			✓		supertype contains non-mandatory features or associations
		✓			supertype contains no features or associations
	delete		✓		no associations to supertype or type compatible; new features in supertype
		✓			no associations to supertype or type compatible; no new features in supertype
				✓	supertype has adjacent mandatory association ends; change not type compatible
			✓		supertype has adjacent non-mandatory association ends; change not type compatible
`IsOfType`		✓			new type is supertype; general case
				✓	new type is supertype; special case for association end with mandatory other end
		✓			new type is subtype and instances are type compatible
				✓	new type is no sub- or supertype
`RefersTo`		✓			
`CanRaise`		✓			

5 Example

The example seen in Figure 4 shows a common case of metamodel evolution. For a given type (*Type1*), a supertype is introduced (*Type2*). The attribute *attr1* is a mandatory feature, since it has a lower cardinality of 1. It is moved to the new supertype.

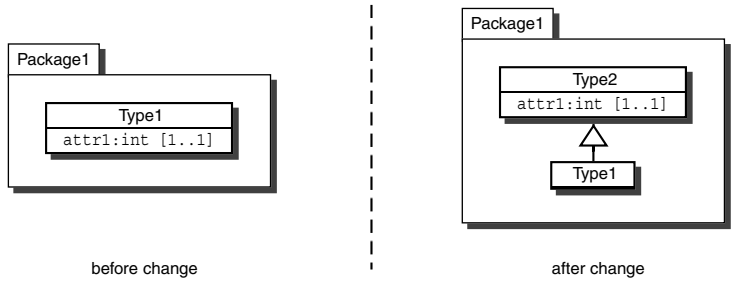


Fig.. 4: Example metamodel

5.1 Change Description

The transition from Version 1 to Version 2 is described by instances of the *Change Metamodel*, as depicted in Figure 5. First, the new type is created in *Change1*. Then, the generalization between the types is added (*Change2*). The move operation for the attribute *attr1* is described as the deletion and addition of the *Contains* link between the attribute and the types in *Change3* and *Change4*.

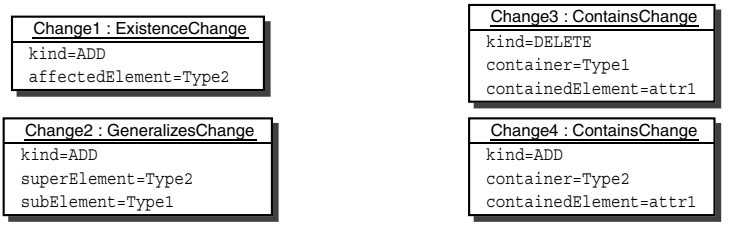


Fig.. 5: Change steps as Change Metamodel instances

5.2 Classification

Since the first change is purely additive, it is *non-breaking*. In the second change, a generalisation link is added. If we looked at the change steps until here, *Type2* would not contain any features, so one could assume that the change were trivially *non-breaking*. However, in the OCL constraints, all other changes of a sequence are also taken into account, as stated in subsubsection 3.2.1. This also includes the changes which are performed after the current change. In the resulting metamodel, *Type2* will contain a mandatory attribute, which would make the change *breaking and not resolvable* because there are no values for it in the instances of *Type1*. But since all of the subtypes of *Type2* (in this case, only *Type1*) contained an attribute of the same name and of the same type in the former metamodel version, existing valid M1 instances contain values for this attribute. This is why the addition of the generalisation is *non-breaking* in this case.

Finally, the mandatory association is moved to the new supertype. This way, all instances of *Type1* are still valid. For *Type2*, the situation is more complex, since a mandatory feature was introduced, which could cause instances to become invalid, as they do not have a value for this feature. But since *Type2* was newly created in the course of this change sequence, there cannot be any instances in existing M1 data, and so the change is also *non-breaking*.

For the overall change, this means that the change is *non-breaking*. The interesting thing about this example is the fact that the single changes would have different severities if they were analysed as singular changes without the context of the change sequence. In total, this would yield a wrong estimation of the overall severity. Only if all four change steps are regarded, the severities can be determined correctly.

5.3 Example OCL constraint

In this section, we take a look at the OCL constraint for *Change3*. As mentioned in the paragraph above, the deletion of the containment link between *attr1* and *Type1* is *non-breaking* because the attribute is moved to a supertype. This is expressed in lines 6–13 of the following constraint.

```
1  context ContainsChange
2  inv: (self.container.oclIsTypeOf(Class) and
3     self.containedElement.oclIsKindOf(StructuralFeature))
4  implies
5  if (self.kind=DELETE) then
6     if self.changeSequence.changes -> exists(c|
7        c.oclIsTypeOf(ContainsChange) and c.kind = ADD and
8        (c.containedElement = self.containedElement or
9        c.containedElement.similar(self.containedElement)) and
10       self.newSupertypesExtended(self.container)
11          -> contains(c.container))
12       -- feature moved to supertype
13       then self.severity = NON_BREAKING
14       -- feature deleted or moved elsewhere
15       else self.severity = BREAKING_RESOLVABLE
16    endif
17    else -- (self.kind=ADD)
18    -- (omitted here)
19 endif
```

Listing 1: Severity Constraint for Change3 (excerpt)

Here, the interesting part is the function *newSupertypesExtended* (line 11), which is the transitive closure of the *newSupertype* helper function shown below. The function calculates the supertype structure in the new version of the metamodel. Functions of this type are necessary since we cannot reference elements in the new version of the metamodel.

```
1  context ModelElementChange::newSupertypes(GeneralizableElement
2                      element) : Set(GeneralizableElement)
3  post: result = element.supertypes
4  -> including(this.changeSequence.changes -> select(ch|
5     ch.oclIsTypeOf(GeneralizesChange) and
6     ch.subtype = element and ch.kind = ADD) -> collect(supertype)
7     )
8  -> excluding(this.changeSequence.changes->select(ch|
```

```
9     ch.oclIsTypeOf(GeneralizesChange) and
10    ch.subtype = element and
11    ch.kind = DELETE) -> collect(supertype)
12    )
```

Listing 2: Helper function

6 Assumptions/Limitations

6.1 Unique Identifiers

As seen in the example above, changes in containment or generalisation are expressed as two operations (delete and add) in the *Change Metamodel*. This makes it more difficult to recognize the semantics of such a change. However, the description is unambiguous since in MOF 1.4, every element has a unique identifier [MOF05].

In order to detect a change in containment, the change sequence has to be searched for elements of the same type (e.g. *ContainsChange*) with different change kind (delete, add) and the identical affected element, which can be determined by the element's unique identifier. The decision to describe changes in this way is backed up by the fact that containment, typing and other relations are expressed by instances of MOF associations like *Contains* and *IsOfType*, i.e. links in a MOF-based metamodel. Since links do not have element identity, it is not possible to distinguish the "modification" of a single link from its deletion and the creation of a new link that connects to one of the former link's ends. Only the MOF constraints which state that an element must be contained in a container or that a typed element must have a type ensure that for every deleted link of these types, a new one is created. Otherwise, the metamodel would not be a valid MOF instance. Since we did not want to hard-code these contraints in the structure of our metamodel, all modifications to associations in MOF are mapped to subtypes of *LinkChange*.

6.2 References to Metamodels

A *Change Metamodel* instance only references elements in one version of the metamodel. The *Change Metamodel* instance is applied like a patch and generates the new version. If new elements are added during the progress of modification, they must be contained with the *Change Metamodel* instance.

This method has the advantage of not having to deal with two versions of the same element, which would be the case if the change description referenced elements from both evolutionary stages. A *Change Metamodel* instance has an inverse that references only elements of the newer metamodel version.

6.3 M1-agnostic analysis

The severity of a *Change Metamodel* instance depends on its own structure and the structure of the metamodel it references, i.e. the M2 level. In general, the nature of M1 instances also has an

influence on the severity of changes to a M2 model. For example, if a class does not have instances, all changes to this class would be non-breaking.

But since metamodels and M1 instances are not necessarily used by the same person, even in the same environment, it may be impossible for the editor of a metamodel to know about all or any M1 instances. For this reason, the classification of this paper is a worst-case assumption: All severities in section 4 have been calculated for the worst-case of possible M1 instances.

7 Related Work

The problem of finding a minimal edit script for hierarchically structured information is adressed in [CRGMW96]. The algorithm presented there calculates a minimal edit script for different version of a graph, under the assumption that no unique identifiers are present. The edit script is composed of atomic editing operations on graphs. For the process of finding a delta of two metamodels without the support of traces, this algorithm could be used to determine a sequence of atomic change steps.

This algorithm is also used by Girschick in [Gir06] to compare UML class diagrams. Girschick introduces the UMLDiff$_{cld}$ algorithm that is based on elementary transformation options. The purpose of this algorithm lies mainly in difference visualisation for graphical UML tools. The visualisation could be generalised to be used with any kind of MOF models.

A metamodel for the description of model deltas has been presented by David Hearnden in the *Deltaware* project [Hea07, pp. 72]. The *Delta* model is based on MOF 2.0 and allows the description of model deltas with the help of identity maps. While the addition and deletion of elements and changes of properties are described thoroughly, the *Delta* model lacks descriptive methods for changes in the model structure, like containment or generalisation. The focus of this work lies on metamodel transformations, for which evolutionary aspects are formalised using the Tefkat language.

Ciccheti et al. [CRP07] propose a metamodel-independent approach for the description of model deltas that is also based on atomic change operations. They also describe model deltas through instances of a difference metamodel, which is derived from a concrete model delta. In contrast, the change metamodel in this paper is fixed, since the meta-model is always the MOF model, and so the semantics of changes can be described more specifically, regarding the effects of metamodel changes to existing instances.

The migration of model instances under metamodel evolution is denoted as adaptation and co-adaptation by Wachsmuth in [Wac07]. The steps of adaptation are described with QVT Relations with respect to instance preservation on the model side. For this purpose, a set of metamodel relations is defined, which allow the estimation of the impact of a metamodel adaption, based on the layout of model data.

For EMF, there are several approaches for the description of metamodel evolution. Becker et al. suggest a process model for semi-automatic evolution, including the change classification that this paper is based on [BGGK07]. The idea behind this process model is that a metamodeling infrastructure should assist the user with the transformation of existing model data by distinguishing automatically adaptable changes from those that need manual interaction.

Herrmannsdörfer defines a language for metamodel evolution and model tranformation called *COPE* [HBJ08], which is used for the evolution of Ecore models. The approach includes *coupled trans-*

actions for the M2 and M1 level which are reusable, as well as tool support. Coupled evolution scenarios are also described by Vermolen et al. in [VV08]. While these approaches focus on the transformation of existing model data, the change classification of this paper is primarily directed at the analysis of the impact of changes to the metamodel.

8 Conclusion

The change metamodel presented in this paper allows a description of the metamodel evolution process of MOF-based metamodels using MOF itself as a description language. Based on this, the severity of changes to a metamodel can be determined for single changes, and, more importantly, for complex change sequences using the change severity classification of section 4. This classification can be used to make statements about the compatibility between arbitrary existing model instances and new versions of the metamodel. The severity classification of changes expressed in the *Change Metamodel* has been noted formally using OCL, so that the results of an automatic or manual classification can be checked by the respective modeling infrastructure. This makes it possible to use the results of the change classifications presented here in a platform-independent way. For manual evaluation of change severities, an overview table has been created.

In a metamodeling infrastructure which allows the comparison of different versions of the same metamodel, the classification presented in this paper could be used to automatically determine the impact of a change to a certain metamodel on existing model data. This would require an automatic calculation of a change metamodel instance from either two versions of a metamodel or from the current editing process of a persisted metamodel. The model editing tool could then determine the severity of the changes, which would enable the metamodel editor to decide about changes to the metamodel regarding the projected impact on existing data as well as interface compatibility of the generated software. The implementation such guidance tools for model editing tools remains subject to future work. For users of the modeling tools, the formal description of metamodel evolution changes would ease the migration of their existing model data to new versions of that software.

The incorporation of techniques alleviating metamodel evolution into modeling infrastrcutures would allow for faster adoptation of modeling tools to user needs.

References

[AHK07] Michael Altenhofen, Thomas Hettel, and Stefan Kusterer. OCL Support in an Industrial Environment. In *Models in Software Engineering. Workshops and Symposia at MoDELS 2006, Genoa, Italy, October 1-6, 2006, Reports and Revised Selected Papers*, volume 4364 of *Lecture Notes in Computer Science*, pages 169–178, Berlin/Heidelberg, 2007. Springer Verlag.

[AP03] Marcus Alanen and Ivan Porres. Difference and Union of Models. In Perdita Stevens, Jon Whittle, and Grady Booch, editors, *"UML 2003" – The Unified Modeling Language, Modeling Languages and Applications 6th International Conference, San Francisco, CA, USA, October 20–24, 2003, Proceedings*, volume 2863 of *Lecture Notes in Computer Science*, pages 2–17, Berlin/Heidelberg, 2003. Springer Verlag.

[BGGK07] Steffen Becker, Thomas Goldschmidt, Boris Gruschko, and Heiko Koziolek. A Process Model and Classification Scheme for Semi-Automatic Meta-Model Evolution. In *Proc. 1st Workshop "MDD, SOA und IT-Management" (MSI'07)*, pages 35–46. GiTO-Verlag, April 2007.

[Bur08] Erik Burger. Metamodel Evolution in the Context of a MOF-Based Metamodeling Infrastructure. Master's thesis, Universität Karlsruhe (TH), September 2008. http://sdqweb.ipd.kit.edu/publications/pdfs/burger2008a.pdf.

[CRGMW96] Sudarshan S. Chawathe, Anand Rajaraman, Hector Garcia-Molina, and Jennifer Widom. Change Detection in Hierarchically Structured Information. In H. V. Jagadish and Inderpal Singh Mumick, editors, *Proceedings of the 1996 ACM SIGMOD International Conference on Management of Data, Montreal, Quebec, Canada, June 4-6, 1996*, pages 493–504. ACM Press, 1996.

[CRP07] Antonio Cicchetti, Davide Di Ruscio, and Alfonso Pierantonio. A Metamodel Independent Approach to Difference Representation. *Journal of Object Technology*, 6(9):165–185, 2007. http://www.jot.fm/issues/issue_2007_10/paper9/index.html.

[Gir06] Martin Girschick. Difference Detection and Visualization in UML Class Diagrams. Technical Report TUD-CS-2006-5, Technische Universität Darmstadt, 2006.

[GKP07] Boris Gruschko, Dimitrios S. Kolovos, and Richard F. Paige. Towards Synchronizing Models with Evolving Metamodels. In *In Proc. Int. Workshop on Model-Driven Software Evolution held with the ECSMR*, 2007.

[HBJ08] Markus Herrmannsdörfer, Sebastian Benz, and Elmar Jürgens. COPE: A Language for the Coupled Evolution of Metamodels and Models. In *1st International Workshop on Model Co-Evolution and Consistency Management*, 2008. http://www.info.fundp.ac.be/mccm/2008/wp-content/uploads/2008/09/9-herrmannsdoerfer.pdf.

[Hea07] David Hearnden. *Deltaware: Incremental Change Propagation for Automating Software Evolution in the Model-Driven Architecture*. PhD thesis, University of Queensland, School of ITEE, October 2007.

[MOF05] Object Management Group. *Meta Object Facility (MOF) Specification, Version 1.4*, July 2005. http://www.omg.org/docs/formal/05-05-05.pdf.

[Mon93] Simon Monk. *A Model for Schema Evolution in Object-Oriented Database Systems*. PhD thesis, Lancaster University, February 1993.

[SJ04] Jim Steel and Jean-Marc Jézéquel. Typing Relationships in MDA. Technical Report 17, University of Kent at Canterbury Computing Laboratory, 2004.

[VV08] Sander Vermolen and Eelco Visser. Heterogeneous Coupled Evolution of Software Languages. In *Model Driven Engineering Languages and Systems, 11th International Conference, MoDELS 2008, Toulose, France, September 28 - October 3*, volume 5301 of *Lecture Notes in Computer Science*, pages 630–644, Berlin/Heidelberg, 2008. Springer Verlag.

[Wac07] Guido Wachsmuth. Metamodel Adaptation and Model Co-adaptation. In *ECOOP'07 – Object-oriented programming*, volume 4609 of *Lecture Notes in Computer Science*, pages 600–624, Berlin/Heidelberg, 2007. Springer Verlag.

A Scalable Approach to Annotate Arbitrary Modelling Languages

Mathias Fritzsche[1], Wasif Gilani[2], Michael Thiele[3], Ivor Spence[4], T. John Brown[5] and Peter Kilpatrick[6]

Abstract: Refinement via annotations is a common practice in Model-Driven Engineering (MDE). For instance, in the case of our Model-Driven Performance Engineering (MDPE) architecture, we are required to annotate different types of process models with performance objectives, constraints and other information. This is used to enable domain experts, such as business analysts, to benefit from an automated performance prediction based decision support.

Currently, the process models are annotated manually, element by element. This approach is not scalable, for instance, in the case where numerous model elements in large model repositories need to be annotated with the same information. Thus, a scalable annotation mechanism is needed which can be used for arbitrary modelling languages. In this paper we propose an architecture which uses a specialized modelling language to express annotations in an efficient way. This language is transformed to model transformation scripts in order to generate annotation models, which separate the annotated information from the target models and, therefore, supports scalable model annotations for modelling languages of choice.

1 Introduction

Refinement of models is a vital component of Model Driven Engineering (MDE) in order to decrease the level of abstraction vertically, for instance, by refining a business process model as an Enterprise Java Bean (EJB) based application. Such vertical refinements enable separation of platform-specific information, such as the middleware used, from platform-independent concerns. Moreover, horizontal refinement is required in order to move from one view point of the system to another. In our previous work [FPG+09] we presented an example of horizontal refinement. That example showed a stepwise transformation chain from business process models representing the business behaviour, to simulation models which combine business behaviour with performance related behaviour.

For all kinds of refinements, in the first step, an annotation of additional information to the target model, e.g., a Business Process Modelling Notation (BPMN) model [Obj06],

[1] SAP Research CEC Belfast, mathias.fritzsche@sap.com
[2] SAP Research CEC Belfast, wasif.gilani@sap.com
[3] Technische Universität Dresden, michael.thiele@inf.tu-dresden.de
[4] Queen's University Belfast, i.spence@qub.ac.uk
[5] Queen's University Belfast, tj.Brown@qub.ac.uk
[6] Queen's University Belfast, p.kilpatrick@qub.ac.uk

is required. In the second step, the annotated (enriched) target model is used, for instance, as input for a code generator or other transformation steps. The annotation can conform to a profile definition, such as proposed for the Unified Modelling Language (UML). Profile definitions are extensions of the target meta-model as described in the UML standard [Obj07]. However, some authors have presented annotation technologies [VCFM, VGK07, FJA+09] where the meta-models of annotations are defined without extending the target meta-model.

We experienced the problem that a systematic and generic approach to the manually performed annotation task itself has hardly been considered yet if annotations are cross-cutting. Instead, target model elements are normally annotated manually, element by element. If numerous model elements in large model repositories need to be annotated with the same information, this becomes a vast manageability problem. Moreover, a solution needs to support arbitrary target modelling languages.

In this paper we propose the combined use of a generic meta-model for model annotations, which can be applied to annotate modelling languages of choice, and a Domain Specific Language (DSL) describing the model annotations at a high level of abstraction, i.e. the user is unaware of the actual underlying employed annotation technology. This annotation script is translated to the Atlas Transformation Language (ATL) [JABK08] in order to use the ATL execution engine to interpret the script and to generate annotation models. This approach complements existing manual annotation editors for cases of cross-cutting annotations.

This paper is structured as follows: Section 2 demonstrates the need for model annotations based on an industrial case study called Model-Driven Performance Engineering (MDPE). In Section 3 we motivate the need for an efficient and scalable way of carrying out model annotation. Section 4 gives an overview of a generic annotation meta-model that we utilized for the solution which is provided in Sections 5. In Section 6 we describe the experience gained with our approach in the context of MDPE. A comparison with the current state of the art is provided in Section 7. Section 8 concludes the paper.

2 Use Case: Model-Driven Performance Engineering

It is the goal of Model-Driven Performance Engineering (MDPE) [FPG+09,FJ07,FGF+08, FBV+09] to provide performance related decision support to domain experts (e.g. business analysts) based on the models they understand, such as BPMN models [Obj06]. The decision support provided by MDPE is currently based on a simulation engine and an optimization engine. It enables answers to questions such as:

- Can available staff cope with the future business growth? This can be simulated by increasing the planned number of business process instances that are intended to be executed in a given time.

- Will training of employees be sufficient to achieve certain business performance targets? This can be simulated by decreasing the net working time for certain process steps.
- Is the hiring of additional employees inevitable? This can be simulated by increasing the resource capacities.
- How many employees are needed at which point of time? This can be answered by an optimization engine which automatically simulates a number of combinations and selects the best combination based on given requirements, objectives and constraints [FGS+08].

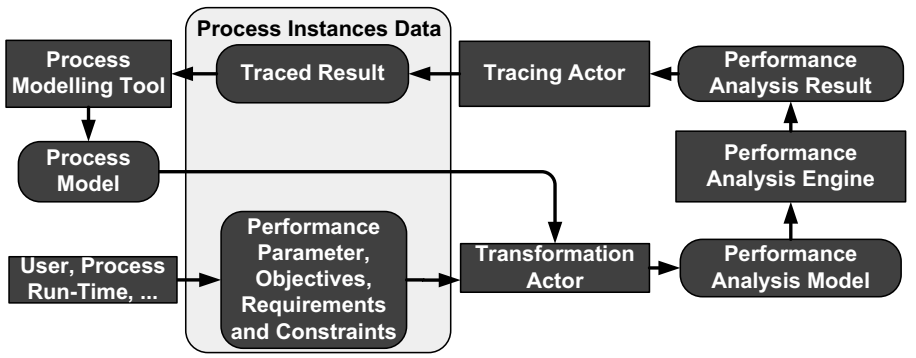


Fig. 1: Model Driven Performance Engineering as a Block Diagram [KGT06]

As shown in Figure 1, MDPE enables automatic generation of *Performance Analysis Models* from different types of *Process Models* conforming to different meta-models. Thus, MDPE extends existing *Process Modelling Tools*, such as the BPMN [Obj06] based tool NetWeaver BPM [SRMS08], other SAP proprietary process modelling tools and JPASS [jCO09]. Different model repositories are employed for storing the *Process Models* and their meta-models. Therefore, in the current MDPE implementation the Eclipse Modelling Framework (EMF) [BBM03] and the SAP Modelling Infrastructure (MOIN) [AHK06] are supported. The meta-models in the EMF repository conform to the ECORE meta-modelling language whereas meta-models in the MOIN conform to the MOF 1.4 [Obj02] meta-modelling language.

Generated *Performance Analysis Models* are used as input for the *Performance Analysis Engine*, such as the AnyLogic simulation engine [XJ 09]. *Performance Analysis Results* are traced back and visualized based on the original *Process Models* the domain expert understands, as described in [FJZ+08, FJA+09].

The generation of the *Performance Analysis Models* is done via model to model transformations. It is shown by Figure 1 that these transformations also require *Performance Parameters*, which are values specifying the resource related behaviour of the modelled process instances (see *Process Instance Data* in Figure 1) over a period of time. Examples are planned or historic workload of a process (e.g., number of arriving Sales Orders in a

Sales Order Processing business process). In order to use not only simulation engines as Performance Analysis Engine, but also other ones such as the optimization engine OptQuest [Rog02], MDPE additionally takes *Objectives, Requirements and Constraints* into account as proposed in [FGS+08].

All this data, together with the *Performance Parameter* and *Traced Results*, are added as annotations to the *Process Models* as discussed in the following section.

3 Identified Problem

At this point of the paper, MDPE has been introduced as an example where model annotations are heavily used.

Current editors for model annotations, such as the editors used for UML profiles, require users to manually select each single model element to enrich it with additional data, such as planned performance parameters, objectives, requirements or constraints in the case of MDPE. This is a time consuming and error prone task.

This manual annotation becomes a significant problem in large models [GLZ06, MF07] in cases where multiple model elements have to be annotated with the same cross-cutting information. For instance, the same business performance requirements in MDPE are often valid for a number of business process steps within a process model. An example is the requirement that all approval steps in a complex business process need to be processed within 2 days. All business process model elements named “Approval” need to be annotated with this requirement.

Concluding mechanism is needed that complements existing manual annotation approaches to carry out cross-cutting model annotation in an efficient, automated and scalable way. This mechanism needs to support modelling languages of choice, such as BPMN, UML, JPASS, etc.

In the following two sections a solution for this problem is provided. First, our approach for non-intrusive annotation of arbitrary modelling languages is summarized. Second, in Section 5, our approach to express annotations in a scalable manner, and to automatically generate annotation models is presented.

As an example we use a BPMN model of a sales business process which is executed by a specific sales unit. We perform the annotation in order to simulate a scenario in which all “Approval” steps in this business process have to be processed by the “ManagerYang”.

4 An Annotation Approach for the Modelling Language of Choice

Like others [D’A05, GP02] we initially used UML profiles [Obj07] for the annotation of additional data to UML Activity Diagrams in our initial version of the MDPE architecture. This enabled us to gain experience with the automatically generated performance models

[FGF+08]. The benefit of this approach was that the tool support, which is available for most UML editors generically, could be utilized for our MDPE specific profiles.

We, however, had to deal also with modelling languages other than UML, such as BPMN [Obj06] conformant models like JPASS [Fle95, jCO09] models and SAP proprietary models. Additionally, because profiles are a meta-model extension mechanism, one needs "write access" to the related meta-models, which is sometimes not possible. This particularly became obvious when we applied the MPDE approach to SAP proprietary models where we were not able to modify the meta-models, as described in [FJA+09]. Thus, the application of a profile-like concept for annotation in the case of SAP proprietary models is not possible. Therefore, in [FJA+09] we defined an alternative solution to UML profiles which can be applied to all modelling languages of choice.

Our approach for model annotations is based on so-called annotation models, as described in [VGK07]. Annotation models enable the definition of separate models containing references to target models. Since the annotation information is encapsulated in its own space, this approach enables the enrichment of arbitrary (process) modelling languages with additional information without polluting them. Moreover, in cases where multiple target model elements are annotated with the same information, only one annotation model element needs to be created.

In order to provide generic tool support, e.g., for editing annotations, we were required to define a basic structure which is refined by concrete annotation meta-models. Therefore, we defined a generic annotation meta-model called *Annotation Meta-Model (AMM)* [FJA+09], as seen in the middle part of Figure 2. The AMM itself inherits from the weaving meta-model *MWCore* provided by the ATLAS group [FBV06] (see upper part of Figure 2). This weaving meta-model enables the definition of links between models independent of any specific modelling language, such as UML.

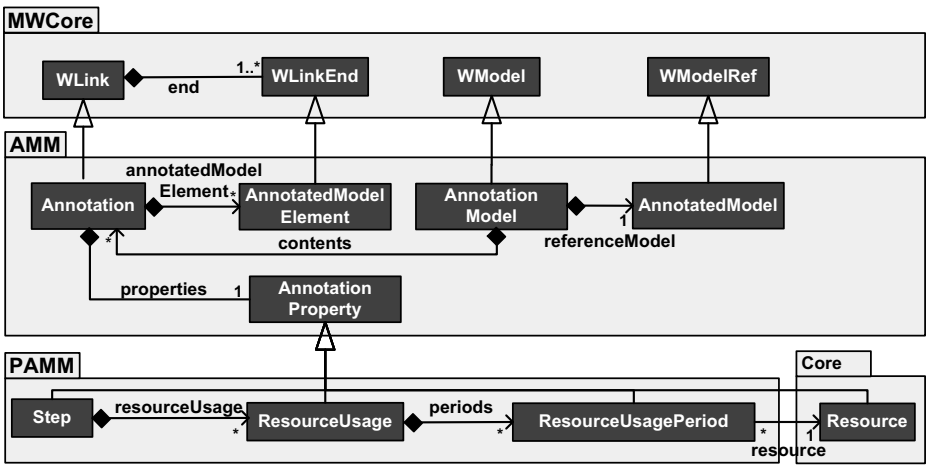


Fig. 2: Generic Annotation Meta-Model (AMM, middle part), and extract of a specific annotation meta-model for performance parameters (Parameter Annotation Meta-Model (PAMM), lower part)

The AMM refines the meta-class *WLink* from the *MWCore* meta-model with the meta-class Annotation. The *WLink* class is indirectly associated with an attribute *"ref"* of type String, which is used to represent references in the enriched models, such as Process Models used in MDPE. Hence, the annotation model elements represent the weaving link to the target model elements. Additionally, an annotation contains multiple *AnnotationProperties*, which represent the annotated information to the source model elements.

An *AnnotationProperty* can be further refined for the needs of a specific annotation meta-model for a certain domain, as shown by the lower part of Figure 2. In the visualized example we refined the AMM for the specific needs of MDPE. A *Step* references a *ResourceUsage* which further references a *ResourceUsagePeriod*. Such a period is linked with a specific *Resource*, such as "ManagerYang". This simple example enables annotation of resource usages to steps over a period of time in a Process Model.

In order to achieve reusability in our annotation meta-model, it is possible to distribute *AnnotationProperties* among packages, such as the "PAMM" (Parameter Annotation Meta-Model) for performance parameters and the "Core" package as shown in Figure 2. In our example, the "Core" package encapsulates the meta-classes, which are referenced within other packages.

To provide generic tool support based on the AMM within the Eclipse environment, the extension of the Annotation Meta-Model can be done by implementing an Eclipse Extension-Point. Thus, a user of the generic AMM based annotation approach can apply annotations to selected model elements in a similar way to the use of UML profiles within model editors like Topcased [The09].

However, with the generic graphical annotation editor it is still necessary to manually select each target model element in order annotate it with additional information.Thus, even if the AMM separates model annotations from the target models and therefore enables annotation of arbitrary modelling languages, a mechanism is still needed to introduce scalability and automation in order to reduce the time needed by the user to define numerous model annotations. This mechanism therefore need to complement the existing annotation editor for cases of cross-cutting annotations.

5 Scalable Model Annotations based on the AMM

For the automated annotation of multiple model elements we define annotations within a separate script. Figure 3 shows that this script includes a *query* phase that defines which model elements should be annotated, and an *execution* phase that defines and creates the annotations for the queried elements (see "Script"). Triggered through user input, both phases are executed (see "Query and Annotation Execution"). As a result, the new annotations with references to the target model are created, such as AMM based annotations (see "Annotation Model").

In the following subsections we outline an approach to implement of the proposed solution.

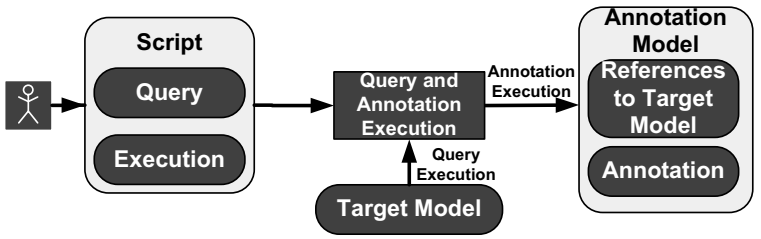


Fig. 3: Proposed solution for non-intrusive model annotation as Block Diagram [KGT06]

5.1 Use of General Purpose Model Transformation Languages

Gray et al. [GLZ06] also proposes to address scalable change evolution for MDE via scripts containing information for refining existing models. He suggests general purpose transformation languages to be used as scripting languages for such refinements. Utilization of such languages for scalable model annotations would result in the following benefits:

- Some general purpose transformation languages abstract from the underlying model repository as long as there is an adapter for the repository. This is required for the MDPE case (see Section 2).
- To implement the *query* phase, transformation languages support an element selection mechanism (most languages offer OCL-based guards or selections on collections of possible elements). Since we support multiple repositories, we cannot expect that all of them offer such a high-level query mechanism.

Summarising, the functionality offered by model transformation languages is sufficient for an annotation language. However, compared to Gray's work, we only wish to create annotation models that conform to meta-models which are based on the generic AMM meta-model.

When we tried to write transformation scripts to create AMM based annotations for arbitray modelling languages, we experienced severe problems with the transformation script based approach. This was due to the fact that the user has to know the generic meta-model of the AMM and create all of its elements such as the *Annotation* elements by hand until the actual *AnnotationProperty* can be created. This is a repetitive and therefore tedious and error-prone task.

Moreover, the syntax of transformation languages is tailored to the general task of model transformations and not for the rather specific task of model annotations. This different purpose again results in intricate code. For instance, ATL's syntax is much broader than what the annotation problem requires and therefore complicates the task of finding a solution.

Concluding, a purely transformation script based approach is not efficient. For instance, in order to specify that all *ManualActivities* with the name "Approval" and which are executed in a specific sales unit have to be annotated with a specific resource consumption from the resource "ManagerYang", a number of *AnnotationProperties* have to be created, namely a number of *Steps* which have references to the *Resource* "ManagerYang". Implementing this script in the Atlas Transformation Language (ATL) requires nearly 150 lines of ATL code and we had to make use of highly ATL-specific functions. A more detailed description of the experiences that we gained with ATL is provided in the next section.

However, a specialized DSL for AMM based model annotations on top of the general purpose transformation language could be employed based on certain assumptions about the AMM based structure of the annotation. Therefore, the effort required by a user for model annotations can be significantly reduced, by still providing an AMM based model annotation approach which permits usage for arbitray modelling languages. In the following section such a DSL is provided.

5.2 A DSL based approach: EQUAL

The benefits of DSLs are manifold. Consel [CM98] and Czarnecki [CVS+06] mention that a DSL is needed in those cases where an isolated problem, such as the automated model annotation, needs to be addressed which could occur again in the future. Furthermore, Jouault [JBK06] claims that with DSLs domain concepts can be represented by constructs of the language and that these constructs usually are high level. Thus, if a construct of a DSL is mapped to a construct of a General Purpose Language (GPL) the DSL construct is typically significantly shorter.

Thus, a goal of a DSL should be to express the task in a concise way and it should hide as much detail as possible from the user of the DSL. Based on this design principle we hide the complex AMM construction and the explicit creation of *Annotation*. That led to a concise declarative way to describe model annotations, which is not possible with transformation languages, since - even if the transformation language itself is declarative - multiple instructions are required to create the annotation. Hence, the syntax can be more tailored and contains fewer elements than a model transformation language offers. We call this AMM based language *EQUAL* - Extended QUery and Annotation Language.

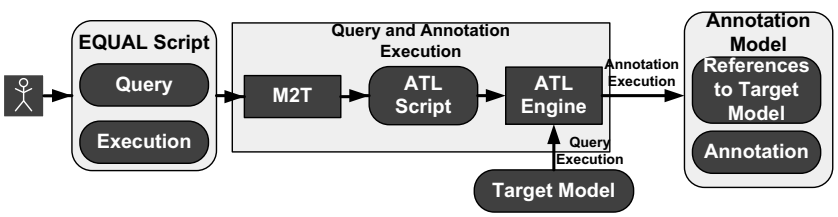


Fig. 4: ATL based architecture for non-intrusive model annotation solution as Block Diagram [KGT06]

The overall architecture of our approach is depicted in Figure 4. The user writes an EQUAL script in a textual editor. This script contains target model queries and *AnnotationProperties*. It is then transformed into ATL scripts that are interpreted. The ATL scripts query the target model and create or alter annotations in the affected annotation model elements.

The EQUAL interpreter is configured with a separate *Configuration* script which is shown in the following listing. This part corresponds to the configuration part of an ATL transformation, i.e. one has to first tell the target meta-models and models. Additionally, it enables to tell the meta-elements which are used for the model annotation. This enables better tool support for the EQUAL editor that we have implemented, especially in terms of auto completion.

```
1  metamodels  {
2    BPMN_NetWeaver : MOIN from 'local:\\';
3    PAMM : EMF from 'uri:http://www.modelplex/amf/2009/MDBPE/PAMM';
4    CORE : EMF from 'uri:http://www.modelplex/amf/2009/MDBPE/CORE';
5  }
6  models  {
7    salesProcessOpportunityManagement : BPMN from 'local:\\
         SalesProcessOpportunityManagement';
8    pam : PARAMETER : from 'ext:C:\\Ann2\\Parameter.mdpe';
9    core : CORE from : from 'ext:C:\\Ann2\\Core.mdpe';
10 }
11 types  {
12   ManualActivity : BPMN_NetWeaver;
13   Step : PAMM;
14   ResourceUsage : PAMM;
15   ResourceUsagePeriod : PAMM;
16   Resource : CORE;
17 }
```

The following listing shows an example of an EQUAL script. The concrete syntax of this script is provided in the appendix. In the following example it can be seen that it consist of a *query* part (lines 1-3 in the listing) and an *annotation* part (lines 4-12 in the listing).

```
1  query  {
2    manualActivities = SELECT ManualActivity FROM
         salesProcessOpportunityManagement WHERE name = "Approval";
3  }
4  annotate  {
5    manualActivities with Step.new  (
6    resourceUsage.add (
7      resource = SELECT Resource FROM core WHERE name = "ManagerYang",
8      periods.add (
9        ResourceUsagePeriod.new (
10       startDate = "1.1.2010",
11       netWorkingTimeConsumption = 0.15))));
12 }
```

The query part corresponds to the point cut expressions in aspect oriented languages [KHH+01, SGSP02]. It can be seen in the listing that we choose a SQL like syntax for these queries. The resulting set of model elements of a query defines the target model ele-

ments that have to be annotated in the annotation part of the script. The example shows the query to get all these *ManualActivitie*s in a BPMN model which are called “Approval”.

The annotate part of the script is similar to the aspect code in AOP [KHH+01, SGSP02]. In the example provided by the listing, attributes of the new annotation are set. All manual activities with the name “Approval” are annotated with a *resourceUsage* of the “ManagerYang” by referencing an existing model element. This reference is enabled, again, via the support of SQL-like queries.

The EQUAL script and the Configuration script of the previous listings are transformed into ATL scripts via a model-to-text transformation. In a first transformation step, the *query* part of the EQUAL script is transformed into an ATL transformation script that queries the target model elements and additionally determines which of these elements were not annotated before. We need to store the *ref* String for those target model elements, since new annotations, created for these elements, should reference them (see Section 4, *ref* attribute). Unfortunately, we had to implement this process in a separate transformation step for reasons described in Section 6.2.

After the execution of this ATL script it is known how many new annotations have to be created and which target model elements they reference. This information is used to generate a subsequent ATL transformation that performs the actual annotation (*execution* phase), since here, new annotations have to be created for every queried model element that was not annotated before.

6 Experiences Gained

Based on our implementation of EQUAL DSL, and the underlying architecture, including the evaluation of the approach for MDPE, we have gained a number of experiences.

6.1 Implementation and usage of the EQUAL DSL

The “openArchitectureWare” framework [ope09] allowed us to easily generate an Eclipse based editor for our DSL that offers user assistance such as syntax highlighting and code completion out of the box or with little customization. It also facilitates checking of conditions. Thus, input errors can easily be detected in the EQUAL editor. This increases further the usability of the EQUAL editor. This is a clear advantage over a pure general purpose transformation language based approach where editors, such as the ATL editor, are not able to provide hints for special use cases, such as MDPE, as they simply allow a wider range of expressions than EQUAL.

Additionally, we compared the efficiency of using EQUAL for model annotations with carrying out the model annotations manually with an annotation editor. Compared to the manual annotation, we particularly experienced benefits using EQUAL when we had to annotate numerous model elements of large target models with the same cross cutting

information, such as the annotation of 26 "Approval" steps in a model containing 772 model elements. In such a case, writing a textual EQUAL script is less repetitive and therefore more timesaving than doing the annotation manually.

6.2 Use of ATL as an Execution Language

We experienced that ATL, which is only one of many possible model transformation languages, is well-suited as an execution language for scalable model annotations due to its powerful language concepts. One example is the extension of the OCL-defined operations for strings, as this includes the use of regular expressions. Hence, it was possible to map regular expressions that appear in an EQUAL query to ATL's OCL expressions.

However, we recognized that the language does not perfectly fit our needs: Due to the possible deep structuring of AMM based annotation meta-models we are forced to create multiple elements and their relations to one another. Therefore, we had to utilize many ATLs rather advanced language features such as the *resolveTemp* method to reference created elements of other rules.

Additionally, refinement transformations have proven to be very useful in our case, since we only need to create new annotations or extend existing ones in an annotation model and do not wish to perform a "real" model transformation with different source and target models. The drawback of this design choice is that the refinement transformations implemented in ATL do not permit the use of imperative code. If we had been able to use such a feature in ATL it would have been possible to iterate over the collection of queried target model elements.

For elements that were not annotated before, we would create new annotations through am imperatively called rule. Since this is not possible with the refinements transformation support of the ATL version used, we had to implement distinct ATL scripts for the *query* phases as well as for the *execution* phase. The first ATL script has to query the target model and to find out which elements need new annotations. This information is needed as an input for the generation of the second ATL script, as we explicitly have to create ATL code for each new annotation. Concluding, the *query* and *execution* phase have to be executed one after another, i.e. in an imperative manner. This concept cannot be mapped to a declarative language which forces construction of an "imperative bridge" between two declarative phases.

This leads to a crucial side effect considering the amount of ATL code to be written for annotations. While the EQUAL script in the given example in the previous section consists of 12 lines of code to create 6 annotations, the generated ATL files contain more than 150 lines of code.

This is verified by another example that we experienced based on MDPE. We ran a query on a model containing 772 elements of which 26 were annotated with the same annotation as in the previous example. The EQUAL script again consisted of 12 lines of code, whereas

the ATL scripts - due to the high number of annotations to be created - contain more than 900 lines of code.

This discrepancy is due mainly to the fact that we have to explicitly create all annotations for each queried target model element and this grows with the number of queried elements. Thus, with the current implementation, ATL in refinement mode cannot be considered to be a scaling model annotation language, but since the ATL code is generated this limitation is not relevant to us. If we would not need to create ATL code for each annotation, we would need less ATL code.

However, even with the possibility of using imperative code in refinement transformations, for instance by using other languages than ATL, the transformation script still would be longer and significantly more complex than the corresponding EQUAL script, as the AMM elements would be visible and the annotation process could not be described in a fully declarative manner.

6.3 DSL Stacking

Although ATL is a DSL, its domain (model transformations) is not specific enough for our task at hand. We showed that making it more domain-specific (model annotations) through the use of another DSL on top of it greatly helps to reduce the coding effort to solve our domain-specific problem. This "DSL stacking" is comparable to the notion of Platform Independent Model (PIM) and Platform Specific Model (PSM) in the MDA architecture of the OMG [Obj03].

7 Related Work

Related to our approach there are a number of aspect oriented languages such as AspectJ [KHH+01], AspectC++ [SGSP02], etc. which offer special constructs and mechanisms like pointcuts and joinpoints to express and weave cross-cutting code back into a target language.

A number of approaches for model annotations, such as UML profiles [PSS07, XWP03, BM05] or model weaving based approaches [VCFM, VGK07] are available. However, scalable model annotations have hardly been considered yet. There is, however, work available to address scalable change evolution for MDE [GLZ06] via general purpose transformation languages.

Compared to that, we propose defining model annotations in their own space with a specialized DSL. This DSL takes advantage of a given annotation structure provided by the generic AMM which enables reducing the number of expressions to be defined by the user. Our approach would also be applicable for other generic annotation approaches, such as provided by UML profiles, but the AMM enables annotating any kind of model conforming to any kind of meta-modelling language. For example, this is needed to apply MDPE

for complex business processes which are orchestrated out of back-end processes, with a long life-cycle and composite processes, with a short life-cycle, where both kinds of processes conform to different meta-models.

Using domain specific (modelling) languages instead of GPLs is not a new idea. A number of examples of applying them for a number of different domain specific problems can be found in the literature since the 1990s [ADR95, GGJ+97, KH97]. Nowadays, graph-based meta-modelling languages, such as MOF [Obj02] are available. Additionally, sufficient tool support for the efficient implementation and management of DSL tools, such as the tools contained in the AMMA open source Eclipse platform [KBJV06], can be utilized.

However, to our knowledge only the approach proposed by Mehr [MF07] addresses the problem of providing a specialized language for systematic model annotations of UML models. Compared to that, our approach can be applied for any target modelling language. Furthermore, we are transforming our DSL to a general purpose transformation language in order to utilize its execution engine. The same concept is applied by state of the art model to code transformations. This enables the utilization of existing code execution engines, such as the Java Virtual Machine (JVM), for modelling languages.

8 Conclusion

Based on the numerous process model annotations required for MDE specialized processes, such as the MDPE process, the need for a systematic model annotation approach is presented. We have presented the combined use of the EQUAL DSL and automatically generated ATL model transformation scripts to automate model annotations which enables utilization of the ATL execution engine. However, we showed that using our DSL for model annotations is much more efficient than manually writing ATL scripts.

Moreover, the generated annotations are represented by a separate AMM conforming model which separates model annotations from the target models and therefore permits the application of the EQUAL approach for arbitrary modelling languages.

We showed that, if an annotations is cross-cutting, EQUAL enables users to perform model annotations in a much more efficient way than doing it manually via state of the art model annotation editors. Thus, modellers are now able to employ our approach as an additional annotation tool when it is necessary to automatically annotate numerous model elements with the same data. Elements can be queried based on their type, name or previously applied annotations. The last query type furthermore enables us to trigger annotations based on already existing annotations which enables stepwise refinement.

As future work we anticipate extending our approach to annotate model elements based on a wider range of algorithms, for instance, to decrease all annotated processing time targets of a business process model by 10%.

References

[ADR95] B. R. T. Arnold, A. Van Deursen, and M. Res. An algebraic specification of a language for describing financial products. In *Proceedings of the IEEE Computer Society Workshop on Formal Methods Application in Software Engineering*, pages 6–13, 1995.

[AHK06] Michael Altenhofen, Thomas Hettel, and Stefan Kusterer. OCL Support in an Industrial Environment. In *Models in Software Engineering: Workshops and Symposia at MODELS 2006, Reports and Revised Selected Papers*, volume 4364 of *LNCS*, pages 169–178, 2006.

[BBM03] Frank Budinsky, Stephen A. Brodsky, and Ed Merks. *Eclipse Modeling Framework*. Pearson Education, 2003.

[BM05] Simonetta Balsamo and Moreno Marzolla. Performance evaluation of UML software architectures with multiclass Queueing Network models. In *Proceedings of the 5th International Workshop on Software and Performance (WOSP'05)*, pages 37–42. ACM, 2005.

[CM98] Charles Consel and Renaud Marlet. Architecturing software using a methodology for language development. In *Proceedings of the 10th International Symposium on Programming Language Implementation and Logic Programming*, volume 1490 of *LNCS*, pages 170–194, 1998.

[CVS+06] Krzysztof Czarnecki, Markus Völter, Thomas Stahl, Jorn Bettin, Arno Haase, Simon Helsen, and Bettina von Stockfleth. *Model-Driven Software Development: Technology, Engineering, Management*. John Wiley & Sons, 2006.

[D'A05] Andrea D'Ambrogio. A model transformation framework for the automated building of performance models from UML models. In *Proceedings of the 5th International Workshop on Software and Performance (WOSP'05)*, pages 75–86. ACM, 2005.

[FBV06] Marcos Didonet Del Fabro, Jean Bézivin, and Patrick Valduriez. Weaving Models with the Eclipse AMW plugin. In *Proceedings of the Eclipse Modeling Symposium, Eclipse Summit Europe*, 2006.

[FBV+09] Mathias Fritzsche, Hugo Bruneliere, Bert Vanhoof, Yolande Berbers, Frédéric Jouault, and Wasif Gilani. Applying Megamodelling to Model Driven Performance Engineering. In *Proceedings of the International Conference and Workshop on the Engineering of Computer Based Systems (ECBS 2009)*, pages 244–253. IEEE Computer Society, 2009.

[FGF+08] Mathias Fritzsche, Wasif Gilani, Christoph Fritzsche, Ivor Spence, Peter Kilpatrick, and Thomas. J. Brown. Towards Utilizing Model-Driven Engineering of Composite Applications for Business Performance Analysis. In *Proceedings of the 4th European conference on Model Driven Architecture Foundations and Applications (ECMDA-FA'08)*, volume 5095 of *LNCS*, pages 369–380. Springer-Verlag, 2008.

[FGS+08] Mathias Fritzsche, Wasif Gilani, Ivor Spence, T. John Brown, Peter Kilpatrick, and Rabih Bashroush. Towards Performance Related Decision Support for Model Driven Engineering of Enterprise SOA Applications. In *Proceedings of the 15th Annual IEEE Computer Society International Conference and Workshop on the Engineering of Computer Based Systems (ECBS'08)*, pages 57–65. IEEE Computer Society, 2008.

[FJ07] Mathias Fritzsche and Jendrik Johannes. Putting Performance Engineering into Model-Driven Engineering: Model-Driven Performance Engineering. In *Models in Software Engineering: Workshops and Symposia at MODELS 2007, Reports and Revised Selected Papers*, volume 5002 of *LNCS*. Springer, 2007.

[FJA+09] Mathias Fritzsche, Jendrik Johannes, Uwe Assmann, Simon Mitschke, Wasif Gilani, Ivor Spence, JohnBrown, and Peter Kilpatrick. Systematic Usage of Embedded Modelling Languages in Automated Model Transformation Chains. In *Proceedings of the 1st International Conference on Software Language Engineering (SLE'08), Revised Selected Papers*, volume 5452 of *LNCS*, pages 134–150. Springer-Verlag, 2009.

[FJZ+08] Mathias Fritzsche, Jendrik Johannes, Steffen Zschaler, Analtoly Zherebtsov, and Alexander Terekhov. Application of Tracing Techniques in Model-Driven Performance Engineering. In *Proceedings of the 4th ECMDA Traceability Workshop (ECMDA-TW)*, pages 111–120, 2008.

[Fle95] Albert Fleischmann. Distributed Systems, Software Design & Implementation. Springer-Verlag, 1995.

[FPG+09] Mathias Fritzsche, Michael Picht, Wasif Gilani, Ivor Spence, John Brown, and Peter Kilpatrick. Extending BPM Environments of your choice with Performance related Decision Support. (Best Paper Award). In *Proceedings of the 7th Business Process Management Conference (BPM'09)*, volume 5701 of *LNCS*, pages 97–112. Springer-Verlag, 2009.

[GGJ+97] Neeraj K. Gupta, Neeraj K. Gupta, Lalita Jategaonkar Jagadeesan, Lalita Jategaonkar Jagadeesan, Eleftherios E. Koutsofios, Eleftherios E. Koutsofios, David M. Weiss, and David M. Weiss. Auditdraw: Generating audits the FAST way. In *In Proceedings of the 3rd IEEE Computer Society International Symposium on Requirements Engineering*, pages 188–197, 1997.

[GLZ06] Jeff Gray, Yuehua Lin, and Jing Zhang. Automating Change Evolution in Model-Driven Engineering. In *Computer*, volume 39, pages 51–58. IEEE Computer Society, 2006.

[GP02] Gordon P. Gu and Dorina C. Petriu. XSLT transformation from UML models to LQN performance models. In *Proceedings of the 2th International Workshop on Software and Performance (WOSP'02)*, pages 227–234. ACM, 2002.

[JABK08] Frédéric Jouault, Freddy Allilaire, Jean Bézivin, and Ivan Kurtev. ATL: A model transformation tool. In *Science of Computer Programming*, volume 72, pages 31–39, 2008.

[JBK06] Frédéric Jouault, Jean Bézivin, and Ivan Kurtev. TCS: a DSL for the Specification of Textual Concrete Syntaxes in Model Engineering. In *Proceedings of the 5th international conference on Generative programming and component engineering (GPCE'06)*, pages 249–254. ACM, 2006.

[jCO09] jCOM1 AG. jPASS! - Subjektorientierte Prozessmodellierung. `http://www.jcom1.com/cms/jpass.html`, 2009.

[KBJV06] Ivan Kurtev, Jean Bézivin, Frédéric Jouault, and Patrick Valduriez. Model-based DSL frameworks. In *Companion to the 21st ACM SIGPLAN symposium on Object-oriented programming systems, languages, and applications (OOPSLA'06)*, pages 602–616. ACM, 2006.

[KGT06] Andreas Knöpfel, Bernhard Gröne, and Peter Tabeling. *Fundamental Modeling Concepts: Effective Communication of IT Systems.* John Wiley & Sons, 2006.

[KH97] Samuel N. Kamin and David Hyatt. A Special-Purpose Language for Picture-Drawing. In *Proeedings of theConference on Domain-Specific Languages*, pages 297–310. Usenix, 1997.

[KHH+01] Gregor Kiczales, Erik Hilsdale, Jim Hugunin, Mik Kersten, Jeffrey Palm, and William G. Griswold. An overview of AspectJ. In *Proceedings of the 15th European Conference on Object-Oriented Programming (ECOOP'01)*, volume 2072 of *LNCS*, pages 327–353. Springer-Verlag, 2001.

[MF07] Farid Mehr and Mathias Fritzsche. Patent application: Annotation of models for model-driven engineering, Attorney Docket No. 2007P00439US/0010-076001, 2007.

[Obj02] Object Management Group. MetaObject Facility (MOF) Specification Version 1.4. `http://www.omg.org/technology/documents/formal/mof.htm`, 2002.

[Obj03] Object Management Group. MDA Guide Version 1.0.1. `www.omg.org/docs/omg/03-06-01.pdf`, 2003.

[Obj06] Object Management Group. Business Process Modeling Notation Specification, Final Adopted Specification, Version 1.0. `http://www.bpmn.org/Documents/OMG%20Final%20Adopted%20BPMN%201-0%20Spec%2006-02-01.pdf`, 2006.

[Obj07] Object Management Group. Unified Modeling Language (OMG UML), Infrastructure, V2.1.2. `http://www.omg.org/spec/UML/2.1.2/`, 2007.

[ope09] openArchitectureWare. `http://www.openarchitectureware.org/`, 2009.

[PSS07] Dorina Petriu, Hui Shen, and Antonino Sabetta. Performance analysis of aspect-oriented UML models. In *Software and Systems Modeling*, volume 6, pages 453–471, 2007.

[Rog02] Paul Rogers. Optimum-seeking Simulation in the Design and Control of Manufacturing Systems: Experiences with OptQuest for Arena. In *Proceedings of the 2002 Winter Simulation Conference (WSC'02)*. Winter Simulation Conference, 2002.

[SGSP02] Olaf Spinczyk, Andreas Gal, and Wolfgang Schröder-Preikschat. AspectC++: An Aspect-Oriented Extension to C++. In *Proceedings of the 40th International Conference on Technology of Object-Oriented Languages and Systems (TOOLS Pacific)*, February 2002.

[SRMS08] Jim Hagemann Snabe, Ann Rosenber, Charles Mølle, and Mark Scavillo. *Business Process Management: The SAP Roadmap.* SAP Press, 2008.

[The09] The Topcased Project Team. Topcased. `http://www.topcased.org`, 2009.

[VCFM] Juan M. Vara1, Valeria De Castro, Marcos Didonet Del Fabro, and Esperanza Marcos. Using Weaving Models to automate Model-Driven Web Engineering proposals. In *Proceedings of Integración de Aplicaciones Web (ZOCO'08)*, pages 86–95.

[VGK07] Markus Völter, Iris Groher, and Bernd Kolb. Mechanisms for Expressing Variability in Models and MDD Tool Chains. In *Proceedings of the SIG-MDSE-Workshop on MDSD in Embedded Systems*, 2007.

[XJ 09] XJ Technologies. AnyLogic — multi-paradigm simulation software. `http://www.xjtek.com/anylogic/`, 2009.

[XWP03] Jing Xu, Murray Woodside, and Dorina Petriu. Performance Analysis of a Software Design using the UML Profile for Schedulability, Performance and Time. In *Proceedings of the 13th International Conference on Modelling Techniques and Tools for Computer Performance Evaluation (TOOLS 03)*, volume 2794 of *LNCS*, pages 291–310. Springer-Verlag, 2003.

Appendix: EQUAL Syntax

In this appendix, the Xtext syntax [ope09] of the EQUAL language is presented.

```
QUAL :
  queryPart=QueryPart
  application=AnnotationPart;

QueryPart :
  "query" "{" name=ID "=" query=Query "}";

Query :
  "SELECT" type=[Type|TypeName] ("and" type=[Type|TypeName] )*
  "FROM" model=[Model] ("and" model=[Model] )*
  ("WHERE" (where=OrExpression))? ";";

String TypeName :
  ID | STRING;

OrExpression :
  andExpressions+=AndExpression
  ("OR" andExpressions+=AndExpression)*;

AndExpression :
  notExpressions+=NotExpression
  ("AND" notExpressions+=NotExpression)*;

NotExpression :
  (not?="NOT")?
  (comparatorExpression=ComparatorExpression |
  parExpression=ParExpression);

ParExpression :
  "(" orExpression=OrExpression ")";

ComparatorExpression :
  left=Feature (comparator="=" | comparator="LIKE" | comparator="<" |
  comparator=">" | comparator="<=" | comparator=">=") right=Value;

Feature :
  name=ID (isMultiple?="[" (index=INT | helper=[Helper] |
  isAll?="all") "]")? ("." nextFeature=Feature)?;

Value :
  IntValue | DoubleValue | StringValue | EnumValue |
  HelperValue | RefValue;

IntValue :
  value=INT;

DoubleValue :
  value=DOUBLE;

StringValue :
  value=STRING;

EnumValue :
```

```
54    value=ENUM;
55
56  HelperValue :
57    value=[Helper];
58
59  RefValue :
60    "ref" value=[ComplexFeatureInstance|RefId];
61
62  Native DOUBLE :
63    "RULE_INT'.'('0'..'9')+";
64
65  Native ENUM :
66    "'#'RULE_ID";
67
68  AnnotationPart :
69    "{" ("annotate" toAnnotate=[Query]
70        "with" annotationProperty=[Type] (
71            isNew?= ".new" | (".adaptAll"
72            ("WHERE" where=OrExpression)?) )
73        "(" ((featureInstances+=FeatureInstance)
74            ("," featureInstances+=FeatureInstance)*)? ")" ";" )* "}";
75
76  FeatureInstance :
77    name = Query | ID (
78        isMultiple?="[" (index=INT | isAll?="all")? "]" ) ?
79        ( isAdd?= ".add"  "("
80            featureInstanceValues+=FeatureInstanceValue
81            ("," featureInstanceValues+=FeatureInstanceValue)* ")"  |
82            "=" value=Value | ":"
83            complexFeatureInstance=ComplexFeatureInstance );
84
85  FeatureInstanceValue :
86    featureInstanceValue=Value |  complexFeatureInstance=
          ComplexFeatureInstance ;
87
88
89  ComplexFeatureInstance :
90    type=[Type] ( "(" refId=RefId ")")?
91    "{" ( (featureInstances+=FeatureInstance)
92        ("," featureInstances+=FeatureInstance)* )? "}";
93    Metamodel : name=ID ":" repository=ID   "from"  id=STRING (
94        "annotates" (annotates="ModelElements" |
95        annotates="Model") )?";";
96
97  Model :
98    name=ID ":" metamodel=[Metamodel] "from" file=STRING ";";
99
100 Type :
101   name=TypeName ":" metamodel=[Metamodel] (
102       isAnnotationProperty?="isAnnotationProperty")? ";";
103
104 String RefId :
105   STRING;
```

”Energie-RMK” - Ein Referenzmodellkatalog für die Energiewirtschaft

José M. González Vázquez[1] und Hans-Jürgen Appelrath[2]

Abstract: Steigende Energiepreise, klimapolitische Zielsetzungen und technologische Weiterentwicklungen insbesondere in der dezentralen regenerativen Energieerzeugung führen zu strukturellen Veränderungen in der Energiewirtschaft. Unternehmen und Softwarehersteller dieser Branche sind gleichermaßen betroffen: Sie stehen vor der Aufgabe, ihre größtenteils jahrzehntelang gewachsenen IT-Landschaften bzw. Software-Produktlinien entsprechend zu modifizieren oder neu zu gestalten. Referenzmodelle haben sich als Hilfsmittel für solche Gestaltungsprozesse in verschiedenen Branchen bewährt. Dieser Beitrag beschreibt einen Ansatz zur Konstruktion eines Referenzmodellkatalogs für Unternehmen und Softwarehersteller in der deutschen Elektrizitäts- und Gaswirtschaft.

1 Ausgangssituation und Problemstellung

Die Energiewirtschaft[3] befindet sich in einem tiefgreifenden Strukturwandel, der sich in den nächsten Jahren noch weiter beschleunigen wird. Die Ursachen sind klimapolitisch und regulativ motiviert, bspw. durch das im Energiewirtschaftsgesetz[4] beschriebene Unbundling[5] [Deu05] und durch Änderungen im Erneuerbare-Energien-Gesetz[6] [Deu08]. Zusätzlich wird der Wandel durch technische Weiterentwicklungen, z.B. in der dezentralen Erzeugung, weiter vorangetrieben. Dies führt zu Veränderungen in der Wertschöpfungskette, von der Gewinnung bis zur Nutzung, und zu einem höheren Anteil an dezentraler Erzeugung (siehe [BBF+08]).[7] Durch steigende Energiepreise werden Energiekosten zu einem wesentlichen Kostenfaktor in Unternehmen, was den Wettbewerbsdruck, insbesondere in der Energiewirtschaft, verstärkt (siehe [Ede08]).

[1] Bereich Energie, OFFIS, Escherweg 2, 26121 Oldenburg, jose.gonzalez@offis.de
[2] Bereich Energie, OFFIS, Escherweg 2, 26121 Oldenburg, appelrath@offis.de

[3] Nach [Gab09] zusammenfassende Bezeichnung für Aufgaben unterschiedlicher Wirtschaftsbereiche zur Bereitstellung von Energiedienstleistungen. Darunter werden verschiedene Aktivitäten von der Gewinnung über den Transport bis zur Umwandlung in Nutzenergie (Wärme, mechanische Arbeit, Licht, Schall u. a.) bei den Verbrauchern verstanden. Im Rahmen dieses Beitrages wird dieser Begriff auf die Elektrizitäts- und Gaswirtschaft eingeschränkt.

[4] Gesetz über die Elektrizitäts- und Gasversorgung (Energiewirtschaftsgesetz-EnWG).

[5] Unbundling (engl. für Entflechtung) beschreibt die eigentumsrechtliche, organisatorische und buchhalterische Trennung der Funktionen Erzeugung, Übertragung und Verteilung, Handel und sonstige Aktivitäten eines Energieversorgers, siehe [Deu05], EnWG Teil 2.

[6] Interessant ist hier die Möglichkeit der Direktvermarktung von Strom aus erneuerbaren Energien.

[7] Gewinnung und Nutzung entsprechen den Wertschöpfungsfunktionen Produktion und Verbrauch aus der Industrie.

Strom und Gas kommt in der deutschen Energiewirtschaft eine hohe Bedeutung zu denn mehr als 45% des Endenergieverbrauchs werden hierdurch gedeckt [Bun08b]. Als leitungsgebundene Energien weisen sie – trotz ihrer physikalischen Unterschiede – eine Reihe von Ähnlichkeiten in ihren Wertschöpfungsketten auf. Ferner gibt es eine Vielzahl von Unternehmen, die beide Sparten (Strom und Gas) bedienen.[8] Daher wird eine spartenübergreifende Betrachtung in diesem Beitrag verfolgt und unter Energiewirtschaft die Elektrizitäts- und Gaswirtschaft verstanden [9].

Folge der organisatorischen und technischen Veränderungen im Energiemarkt sind wachsende Kommunikationsanforderungen. Auf betriebswirtschaftlicher Ebene schafft das Unbundling weitere Marktpartner, ehemals unternehmensinterne Prozesse werden in unternehmensübergreifende Prozesse externalisiert. Parallel dazu führen Smart Metering[10] und die zunehmende dezentrale Erzeugung zu einem erhöhten unternehmensübergreifenden Austausch von Energiedaten auf technischer Ebene. Diese Veränderungen bringen eine Reihe von Anforderungen an Informationssysteme mit sich, wie zusätzlich zu unterstützende Funktionalität und zu gewährleistende Sicherheitsanforderungen (siehe auch [AC07] und [BGPA09]). Sowohl Unternehmen in der Energiewirtschaft, im folgenden auch Anwendungsunternehmen genannt, als auch unterstützende Softwarehersteller und Beratungshäuser sind diesen Veränderungen ausgesetzt. Im Rahmen von Zusammenschlüssen und Zukäufen können weitere IT-Systeme bzw. Softwareprodukte das bisherige Anwendungsportfolio erweitern und Konsolidierungen oder Integrationslösungen erforderlich machen. Anwendungsunternehmen sind an der bestmöglichen Unterstützung der betrieblichen Aufgaben durch IT bei minimalen Kosten (IT/Business Alignment) interessiert. Daneben streben sie die Erschließung neuer Geschäftsfelder mittels IT-Einsatz an.

Softwarehersteller stehen vor der Aufgabe, für die Unterstützung der betrieblichen (teilweise neuen) Aufgaben von Unternehmen in der Energiewirtschaft zeitnah qualitativ hochwertige sowie kostengünstige Softwareprodukte anzubieten und zu entwickeln. Im Gegenzug müssen Anwendungsunternehmen entsprechende Software selbst entwickeln oder erwerben, einsetzen und betreiben.

2 Stand der Wissenschaft und Praxis

Die Anforderungsermittlung ist bei der Softwareauswahl und -entwicklung von entscheidender Bedeutung und gestaltet sich oftmals zeitaufwendig.

[8] Neben den größten fünf Energieversorgungsunternehmen (EVU) Deutschlands (RWE, EON, EnBW, Vattenfall und EWE) sind eine Reihe von EVU, insbesondere Querverbundunternehmen, in beiden Bereichen aktiv, siehe hierzu auch [Bun07].

[9] Zur Komplexitätsreduktion werden Fernwärme und die Mineralölwirtschaft nicht weiter betrachtet, eine entsprechende Erweiterung des hier beschriebenen Ansatzes ist vorstellbar.

[10] Unter Smart Metering wird hier zusammengefasst die automatische Verarbeitung, der Transfer, das Management und die Verwendung von Messdaten verstanden.

Referenzmodelle sind hierbei bewährte Hilfsmittel zur Entwicklung von Informationssystemen.[11] Bekannte Modelle in diesem Zusammenhang sind bspw. für die Industrie das Y-CIM [Sch02] und für Handelsinformationssysteme das Handels-H [BS04]. Sowohl bei der Gestaltung von Anwendungslandschaften als auch bei der Softwareproduktlinienentwicklung wird der Einsatz von Referenzmodellen empfohlen, siehe bspw. [EHH+08] und [PBL05].

Infolge von Recherchen und Befragungen von Domänenexperten wurden verschiedene (Referenz-) Modelle und Standards für den Strom- und Gasmarkt identifiziert.[12] In Bezug auf Standards wurden, den Empfehlungen von Domänenexperten folgend, insbesondere die internationalen Standards der International Electrotechnical Commission (IEC) des Technical Committee 57 (TC 57) ”POWER SYSTEMS management and associated information exchange” untersucht (siehe [URS+09]).[13]

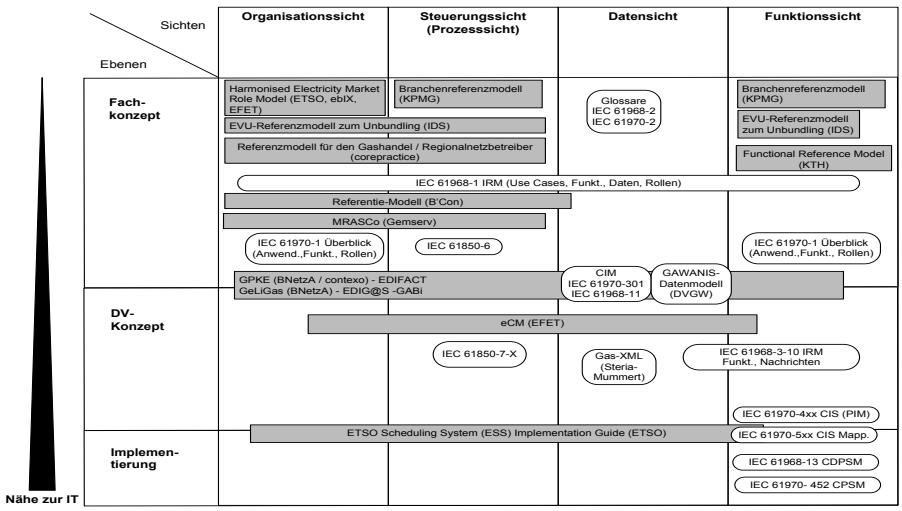


Abb. 1: Strukturierung einer Auswahl von Modellen/Standards der Energiewirtschaft (Gas- und Strombereich) anhand der ARIS-Ebenen und Sichten nach [Gon09]

In Abbildung 1 ist ein Auszug der recherchierten Modelle (Rechtecke) und Standards (Rechtecke mit abgerundeten Ecken) der Energiewirtschaft anhand der ARIS-Ebenen und -Sichten nach Scheer (siehe [Sch02]) dargestellt. Aufgrund der weiten Verbreitung der ARIS-Methode in der Modellierung wurden die hierin beschriebenen Sichten und Ebenen zur Strukturierung verwendet. Abbildung 1 veranschaulicht hierbei die Durchdringung von Normen und Standards im Hinblick auf die ARIS-Ebenen (Nähe zur IT) und -Sichten (von

[11] Sie werden seit Anfang der 90er-Jahre in unterschiedlichen Branchen zur Komplexitätsreduktion und zur Effizienzsteigerung bei der Prozessgestaltung eingesetzt.

[12] Auf eine komplette Liste mit Quellen wird an dieser Stelle aus Platzgründen verzichtet. Insgesamt wurden mehr als 30 Modelle identifiziert, die Suche wird weiter fortgesetzt. Die Recherche erfolgte auf Basis des Desk-Research unterstützt durch Diskussionen mit Domänenexperten.

[13] Die IEC ist die führende internationale Organisation zur Erarbeitung und Veröffentlichung von Standards im elektrotechnischen Umfeld [Int07]. Weitere Informationen zum TC57 finden sich unter http://tc57.iec.ch .

der Organisations- bis zur Funktionssicht). Bei der Auswahl darzustellender Modelle wurden vornehmlich bei Domänenexperten bekannte Modelle und Standards berücksichtigt.

Die identifizierten Referenzmodelle und Standards unterscheiden sich dabei stark hinsichtlich der betrachteten Ebenen und Sichten sowie der fokussierten Anwendungsbereiche. Entweder betrachten sie einzelne Sichten/Ebenen für eine Reihe von Anwendungsbereichen (z.B. Harmonised Electricity Market Role Model der ENTSO-E[14]) oder einzelne Anwendungsbereiche Sichten- und Ebenen-übergreifend (z.B. das GPKE Modell der BNetzA[15]). Die Modelle sind vornehmlich im Umfeld von Behörden, Verbänden, Standardisierungsorganisationen sowie im Auftrag von Softwareherstellern und Beratungshäusern entstanden. Wissenschaftliche Beiträge wurden bis auf das Functional Reference Model der KTH [NGN06] bisher nicht identifiziert. Zukünftig sind hier aber aufgrund verschiedener nationaler und internationaler Förderprojekte im Energiebereich weitere Beiträge zu erwarten.[16] Viele der identifizierten Modelle und Standards sind nicht oder nur teilweise öffentlich zugänglich. In der Energiewirtschaft fehlt ein Modell, das sich einer branchenbezogenen Strukturierung und Konsolidierung widmet, Verweise zu bestehenden Modellen/Standards aufzeigt und somit einen Überblick ermöglicht. Softwarehersteller und Systemdienstleister fordern zur Entwicklung eines offenen und herstellerunabhängigen Branchenreferenzmodells auf [Ban08].

Aufgrund der Vielzahl der unterschiedlichen Modelle gestaltet sich die Identifikation von im spezifischen Kontext geeigneten Modellen zur Unterstützung der Anforderungsanalyse schwierig.

Aus den oben angeführten Gründen erscheint die Entwicklung eines Referenzmodellkatalogs[17] als Übersicht von Referenzmodellen[18] im Sinne von [FL02] für die Energiewirtschaft sinnvoll.

3 Lösungsansatz

Nachfolgend wird der Energie-Referenzmodellkatalog (Energie-RMK[19]), ein Referenzmodellkatalog für die Energiewirtschaft, beschrieben.

[14] ENTSO-E – European Network of Transmission System Operators. Hier wird in dem Rollenmodell nur die Organisationssicht betrachtet.

[15] GPKE – Geschäftsprozesse zur Kundenbelieferung mit Elektrizität. Das zwischen Fach- und DV-Konzept angesiedelte GPKE-Modell betrachtet alle Sichten; BNetzA - Bundesnetzagentur

[16] Siehe beispielsweise nationale Projekte und Initiativen wie E-Energy [Bun08a] und Internet der Energie [BBB+08] sowie die amerikanische Smart Grids Initiative [Ele09].

[17] Unter einem Referenzmodellkatalog wird nach [FL02] eine in Tabellenform vorliegende Übersicht über Referenzmodelle, die nach methodischen Gesichtspunkten erstellt und innerhalb eines gegebenen Rahmens weitestgehend vollständig und systematisch gegliedert ist, verstanden.

[18] In der Literatur herrscht kein einheitliches Verständnis für den Begriff "Referenzmodell", siehe Untersuchungen in [Tho06] und [FL04]. Im Rahmen dieser Arbeit wird die nutzungsorientierte Referenzmodelldefinition nach ([Tho06], S. 17) verfeinert. In diesem Sinne soll hier jedes Modell bzw. Teilmodell, das auf fachlich-inhaltlicher Ebene bei der Unterstützung der Konstruktion eines anderen spezifischen Modells der Energiewirtschaft genutzt werden kann, als Referenzmodell angesehen werden (Metamodelle sind hierbei explizit ausgeschlossen).

[19] Im Folgenden auch als RMK oder Katalog referenziert.

3.1 Ziele und Bestandteile des Energie-RMK

Ziel des Energie-RMKs ist es, die Gestaltung von Anwendungen bzw. Softwareprodukten, insbesondere im Rahmen der Anforderungsanalyse, zu unterstützen. In diesem Zusammenhang sollen die Identifikation von geeigneten Modellen und Standards erleichtert sowie die Entwicklung unternehmensspezifischer Modelle unterstützt werden. Hierzu strukturiert der Energie-RMK die in Abschnitt 2 angedeutete Vielfalt an Modellen und Standards und liefert einen funktionalen Rahmen.

Auf Basis des RMKs sollen Anwendungsunternehmen und Softwarehersteller in die Lage gebracht werden, ihre aktuelle (IST) und zukünftige (SOLL) IT- bzw. Softwareproduktlandschaft in den Energie-RMK einzuordnen bzw. auf dieser Basis zu erarbeiten. Der RMK dient hierbei als Wissensbasis zur Identifikation von weiteren Informationsquellen, insbesondere Referenzmodellen, Gesetzen, Standards und Normen. Durch die Einordnung von Informationssystemen in den RMK soll eine Abdeckungsanalyse bezüglich zu unterstützender Funktionen sowie Gesetzen gegenüber den dort aufgeführten logischen Anwendungen durchführbar sein. Weiterhin sollen diese Funktionen näher beschreibende Informationsquellen, wie Standards oder Gesetze, angezeigt werden.[20]

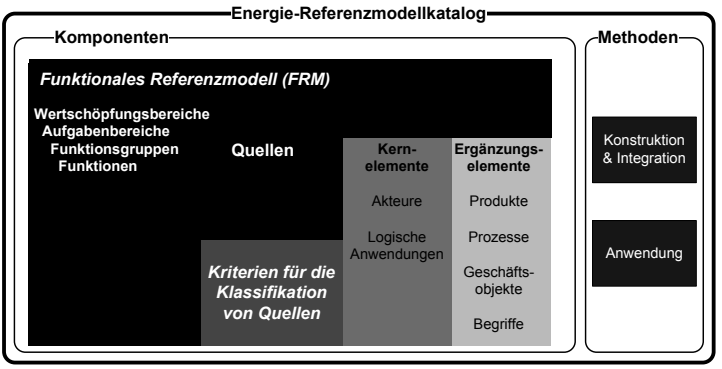


Abb. 2: Bestandteile des Energie-Referenzmodellkatalogs

Abbildung 2 stellt die Komponenten und Methoden des Energie-RMKs dar. Der Energie-RMK besteht aus vier Komponenten: ein Struktur bildendes funktionales Referenzmodell (FRM), Quellen und Kriterien zu deren Klassifikation sowie Kern- und Ergänzungselemente. Kernelemente stehen im Fokus des hier zu erarbeitenden RMKs und werden daher im Detail möglichst vollständig beschrieben. Im Gegensatz dazu werden Ergänzungselemente zur Anreicherung des RMKs eingesetzt und nur grob beschrieben. Hierbei wird angenommen, dass Funktionen bzw. Aufgaben nicht für sich alleine stehen, sondern Operationen an fachlichen Informationsobjekten ausführen. Dies erfolgt durch Akteure mit dem Ziel Produkte/Dienstleistungen bereitzustellen und zwar innerhalb von unternehmensinternen oder unternehmensübergreifenden Prozessen. Diese Annahme entspricht im We-

[20] Bei neu erlassenen Gesetzen oder Richtlinien und Integration in den Energie-RMK können zu untersuchende Funktionsbereiche leicht identifiziert werden.

sentlichen der ARIS-Methodik und den dort verankerten eEPK[21], die als Bindeglied zwischen den verschiedenen ARIS-Sichten (Daten-, Prozess-, Funktions- und Organisationssicht) fungieren. Zusätzlich werden zur Verbesserung des Verständnisses zentrale Begriffe erläutert. Die lediglich grob ausgeprägten Ergänzungselemente dienen der Qualitätssicherung, um die Erfassung aller relevanten Funktionen sicherzustellen.

Zusätzlich werden zwei Methoden bereitgestellt, einerseits zur Konstruktion und zur Integration weiterer Quellen in den Katalog sowie andererseits zur Nutzung des RMKs. Im Rahmen dieses Beitrages werden die Methoden nicht weiter in der Tiefe betrachtet.

Der Energie-RMK soll folgende Fragen beantworten bzw. zur besseren Beantwortbarkeit beitragen:

- Welche Funktionen werden durch welche Wertschöpfungs- und Aufgabenbereiche sowie Funktionsgruppen gruppiert? (FRM)
- In welchen Quellen finden sich welche weitere Beschreibungen zu den verschiedenen Funktionen? (Quellen und Kriterien)
- Welche Funktionen werden von welchen Akteuren wahrgenommen bzw. durch welche typischen Anwendungen unterstützt? (Kernelemente)
- Welche Produkte, Prozesse, Geschäftsobjekte und Begriffe stehen im Zusammenhang mit den groben Wertschöpfungsbereichen und Aufgabenbereichen in der Energiewirtschaft? (Ergänzungselemente)

Das FRM, bestehend aus Wertschöpfungsbereichen, Aufgabenbereichen, Funktionsgruppen und Funktionen, bildet den Kern (siehe Abbildungen 3 und 5). Das FRM beschreibt aus Sicht von Unternehmen der Energiewirtschaft spezifische Wertschöpfungs- und Aufgabenbereiche. In Abbildung 3 ist die fachliche Matrix des FRM dargestellt, auf der horizontalen Ebene sind die Wertschöpfungsbereiche von links nach rechts von der Produktion/Gewinnung bis zum Einsatz als Spalten aufgeführt, diese werden von typischen Aufgabenbereichen (wie Beschaffung und Absatz) als Zeilen durchzogen. Mit Hilfe der Matrix erfolgt die Strukturierung auf oberster Ebene, in weiteren Detaillierungsstufen werden Funktionsgruppen und Funktionen beschrieben. Auf Basis dieser Strukturierung sollen logische Anwendungen[22], Akteure und Quellen (Gesetze, Verordnungen, etc.) zugeordnet und somit "einsortiert" werden. Die Prozesskette des FRM enthält vier zentrale Wertschöpfungsfunktionen der Elektrizitäts- und Gaswirtschaft: Produktion[23], Handel, Transport und Einsatz. Je nach Teilbereich sind entsprechende Funktionsmuster auf betriebswirtschaftlicher und/oder technischer Ebene ausgeprägt.

Die Kategorisierung von Quellen stellt einen wesentlichen Aspekt der Arbeit dar, hier wird auf bewährte Klassifikationskriterien für Referenzmodelle und Standards aus [Sch98],

[21] eEPK - erweiterte Ereignisgesteuerte Prozesskette.

[22] In Anlehnung an den Begriff der Anwendungslandschaftskomponente in [EHH+08] werden hierunter logische Anwendungssysteme wie bspw. ein Energiedatenmanagementsystem verstanden. Diese bündeln eine bestimmte Menge fachlicher Funktionalität unabhängig von einer konkreten Implementierung.

[23] Produktion wird in diesem Beitrag als Synonym für die Umwandlung in nutzbare Energie verwendet.

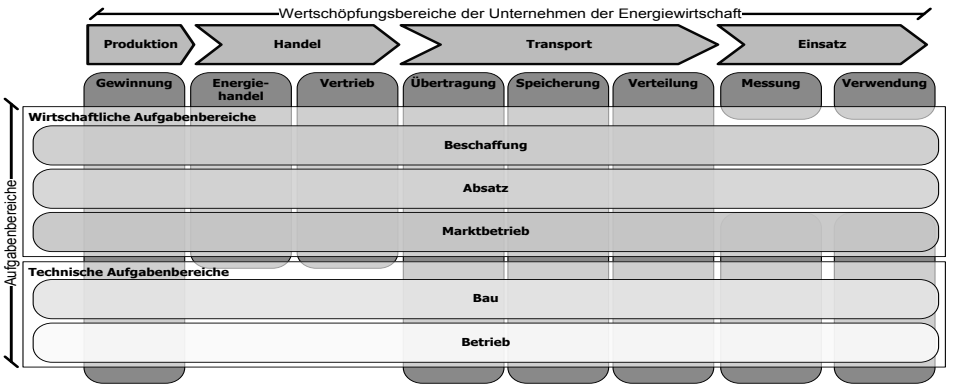


Abb. 3: Matrix (Wertschöpfungs- und Aufgabenbereiche) des funktionalen Referenzmodells des Energie-RMKs (in Anlehnung an [Gon09])

[vB03] und [dV06] zurückgegriffen. Abbildung 4 zeigt die drei wesentlichen Merkmale für die Klassifikation und mögliche Ausprägungen[24] auf: Abdeckung - abgedeckte Ebenen und Sichten, Typ - Herkunft und Art der Quelle, sowie Status - angenommener Einsatz und Status der Weiterentwicklung.

Merkmale		Ausprägungen								
Abdeckung	Ebenenbezug	Fachkonzept			DV-Konzept			Implementierung		
	Sichten	Funktionen			Logische Anwendungen			Produkte		Prozesse
		Daten			Akteure			Begriffe		
	Granularität	Grob			Detailliert					Anzahl
	Funktionales Referenzmodell		Gewinnung	Energiehandel	Vertrieb	Übertragung	Speicherung	Verteilung	Messung	Verwendung
		Beschaffung								
		Absatz								
		Abwicklung								
		Bau								
		Betrieb								
Typ	Politische Region	Deutsch		International	Europäische Union					
	Dokumententyp	Spezifikation			Empfehlung		Verordnung		Glossar	
Status	Anwendung	in der Industrie			in der Forschung			k.a.		
	Weiterentwicklung	in Arbeit			abgeschlossen			letzte Änderung:		

Abb. 4: Kriterien für die Klassifikation von Quellen (Auszug, in Anlehnung an [PGS09])

In Abbildung 5 ist ein erster Auszug des Energie-RMKs aufgeführt, die Struktur orientiert sich dabei an [FL02]. Der Gliederungsteil wird aus der funktionalen Struktur gebildet, in die entsprechende Quellen und andere Kernelemente (Hauptteil) eingeordnet werden. Die Kategorien für die Klassifikation von Quellen aus Abbildung 4 bilden den Zugriffsteil. Abbildung 5 veranschaulicht hier beispielhaft die Einordnung des Paragraphen 50 des Energiewirtschaftsgesetzes (EnWG) zu einer Funktion des FRM.

Neben der Strukturierung von Quellen, die einen wesentlichen Teil des RMKs ausmachen, sollen alle Elemente anhand des Gliederungsteils, allerdings auf unterschiedlichen Detallierungsstufen, kategorisiert werden. Ergänzungselemente (wie beispielsweise Prozesse)

[24] Hierbei sind Mehrfachauswahlen möglich.

Gliederungsteil				Hauptteil		Zugriffsteil									
Ebenen				Quellen		Modellkriterien									
						Abdeckung							Typ		Status
Funktionales Referenzmodell															
1 Wertschöpfung	2 Aufgabenbereich	3 Funktionsgruppen	4 Funktionen	Name	Details	Ebenenbezug	Sichten	Granularität	Funktionales Referenzmodell	Aufgabenbereiche	Politische Region	Dokumententyp	Anwendung	Weiterentwicklung
Gewinnung	Beschaffung	Erschließung												
		Operative Beschaffung	Beschaffen von Rohstoffen für die Sicherung der Energieversorgung	EnWG	§ 50									
				…	…									
				…	…									
		Lieferantenmanagement												
	Absatz													
	Abwicklung													
	Bau													
	Betrieb													
Energiehandel	Beschaffung													
	Absatz													
	Abwicklung													
Vertrieb	…													

Abb. 5: Struktur und exemplarische Inhalte des Referenzmodellkatalogs (Auszug)

werden nur grob und daher nur auf Ebene zwei eingeordnet (siehe Abbildung 2). Der Zugriffsteil besteht in diesem Fall nur aus einer Beschreibung des einzelnen Elements.

3.2 Methodisches Vorgehen

Die Entwicklung des RMKs erfolgt auf Grundlage der Design-Science Methode von Hevner et al. [HMPR04] und orientiert sich an den dort beschriebenen sieben Forschungsrichtlinien, siehe Abbildung 6. Charakteristisch für die Methode ist ein iteratives Vorgehen mit abwechselnden Phasen der Konstruktion und Evaluation, das sogenannte ”build and evaluate”. Weitere methodische Grundsteine bilden die Arbeiten von [FL02] zu Referenzmodellkatalogen bezüglich Vorgehensmodell und Begriffsverständnis sowie die Grundsätze Ordnungsmäßiger Modellierung (GOM) von [Sch98]. Aufgrund der Komplexität der Energiewirtschaft und eigene in Projekten gesammelte Erfahrungen der Autoren erscheint ein iteratives Vorgehen mit kontinuierlichen Evaluationszyklen nach Hevner erfolgversprechend. Im Folgenden wird das Vorgehen zu den Nummerierungen der Design Richtlinien in Abbildung 6 in Bezug gesetzt.

Im Zentrum steht hierbei die Konstruktion und Evaluation des Katalogs, welche entsprechend der Richtline ”Design als Suchprozess” (R6) in mehreren Versionen schrittweise erarbeitet wird. Auf die Anforderungen aus den Forschungsrichtlinien ”Design als zielgerichtetes Artefakt” (R1), ”Problemrelevanz” (R2) und ”Beitrag der Forschung” (R4) wurde in den Abschnitten 1 und 3 eingegangen. Als Artefakte werden ein Energie-RMK und Methoden zur Konstruktion und Nutzung des Katalogs erarbeitet (R1). Die Unterstützung der Gestaltung von Softwareprodukten in der Energiewirtschaft ist gerade vor dem Hintergrund des Strukturwandels von Bedeutung (R2). Der RMK trägt zur Erweiterung und Strukturierung der bisherigen Wissensbasis bei und leistet damit einen Beitrag zur For-

Design-Science Forschungsrichtlinien		Beschreibung
R1	**Design als zielgerichtetes Artefakt** ***(Design as an Artifact)***	Das Ergebnis von Design-Science-Forschung stellt ein innovatives, zielgerichtetes Artefakt (wie ein Konstrukt, ein Model, eine Methode oder eine Instanz) dar, um ein Problem zu lösen.
R2	**Problemrelevanz** ***(Problem Relevance)***	Das Ziel von Designwissenschaft ist die Entwicklung technischer Lösungen für wichtige aktuelle und zukünftige Probleme in Unternehmungen. Relevanz wird über den Nutzen definiert, den eine Lösung stiftet.
R3	**Evaluierung** ***(Design Evaluation)***	Nutzen, Qualität und Effizienz eines Artefaktes müssen durch adäquate wissenschaftliche Methoden evaluiert werden.
R4	**Beitrag der Forschung** ***(Research Contributions)***	Designwissenschaft muss einen klar definierbaren Beitrag zur Problemlösung und/oder den Forschungsmethoden der Disziplin liefern. Die Innovationshöhe eines Beitrages kann in Bezug auf die Neuigkeit, die Allgemeingültigkeit und die Bedeutung eines Artefaktes für einen Anwendungsbereich gezeigt werden.
R5	**Methodische Stringenz in den Forschungsmethoden** ***(Research Rigor)***	Designwissenschaft erfordert die stringente Anwendung wissenschaftlicher Methoden in der Schaffung und in der Evaluierung von Artefakten.
R6	**Design als Suchprozess** ***(Design as a Search Process)***	Design stellt einen Suchprozess dar, in dem Lösungen vorgeschlagen, verfeinert und evaluiert werden, um schrittweise eine Wissensbasis aufzubauen. Hier gilt es, geeignete Methoden anzuwenden, um auf Basis der verfügbaren Mittel in einer endlichen Zeit zu einem akzeptablen Ergebnis zu gelangen.
R7	**Weitergabe von Forschungsergebnissen** ***(Communication of Research)***	Die Ergebnisse von Designwissenschaft müssen sowohl technologieorientierten als auch managementorientierten Interessenten effektiv vermittelt werden können.

Abb. 6: Forschungsrichtlinien des Design Science nach Hevner et al.; in Anlehnung an [HMPR04] und [Bic05]

schung (R4). Die Richtlinien ”Evaluierung” (R3), ”Methodische Stringenz in den Forschungsmethoden” (R5) und ”Weitergabe von Forschungsergebnissen” (R7) betreffen das Vorgehen und werden im Folgenden behandelt.

Zentraler Bestandteil im Rahmen der Arbeit ist die Konstruktion eines Referenzmodellkatalogs für Unternehmen der Energiewirtschaft. Hierbei werden für die Energiewirtschaft spezifische und grundlegende Funktionen identifiziert, die zur Gliederung des RMKs verwendet werden. Aufbauend auf [FL02] wird eine Gliederung des Energie-RMKs in die drei Teile Gliederungsteil, Hauptteil und Zugriffsteil vorgenommen. Für den Gliederungsteil wird eine entsprechende funktionale Untergliederung der Aufgaben von Energieversorgungsunternehmen vorgesehen. In Anlehnung an den Begriff der Facharchitektur in [Kel07] werden nur branchenspezifische Funktionen aufgeführt; branchenneutrale Themenbereiche wie Personal oder Buchhaltung werden nicht betrachtet. Diese Struktur stellt selbst ein funktionales Referenzmodell dar. Der Fokus des Katalogs liegt auf der Strukturierung von Quellen zu Aktivitäten von Energieversorgungsunternehmen entlang der Wertschöpfungskette, daher erfolgt hier die Einordnung von bestehenden (Referenz-) Modellen, Standards und Normen sowie logischen Anwendungen.

Auf Basis vorhandener Modelle, Standards und Beschreibungen in der Literatur sowie Diskussionen mit Experten und Durchführung von Workshops, in denen das Modell diskutiert und angewendet wird, gilt es, iterativ verfeinerte Katalogversionen zu erstellen (R6). Neben der fachlichen Fokussierung auf Modelle und Standards der Energiebranche sollen, insbesondere bezüglich Strukturierung und Gestaltung des Modells, Erfahrungen aus verwandten Branchen wie der Telekommunikation, bspw. Prozess- und Datenmodelle wie

eTOM und SID[25], der Industrie [Sch02] und dem Handel [BS04] genutzt werden. Trotz der Fokussierung auf den deutschen Energiemarkt sollen neben nationalen auch internationale Modelle (bspw. von der ENTSO-E) und Standards (bspw. von der IEC) berücksichtigt werden, um das dort beschriebene Wissen zu nutzen. Ausgehend von den bisherigen Ergebnissen und Studien (siehe [URS+09]) wird im Umfeld von Standardisierungsorganisationen (wie IEC), Branchenverbänden (wie ETSO) und Regulierungsbehörden sowie Beratungsunternehmen und Softwarehäusern nach weiteren Modellen und Standards recherchiert. Aufgrund der Komplexität der Energiebranche ist eine umfassende detaillierte Betrachtung nur mit erheblichem Aufwand möglich. Daher erfolgt eine grobe Betrachtung "in die Breite" die für ausgewählte Bereiche (wie "Smart Grids" , siehe [Ele09]) punktuell detailliert wird.

Neben den GOM und dem Vorgehensmodell zur Konstruktion von Referenzmodellen nach Schütte [Sch98] sollen weitere bewährte Methoden, beispielsweise zur Konstruktion von Referenzmodellkatalogen [FL02] und Ordnungsrahmen[26] [Mei01], angewendet werden (R5).

Durch das bereits als Ontologie[27] vorliegende und bewährte Common Information Modell (CIM)[28] der IEC erscheint die Repräsentation des RMKs als Ontologie zur semantischen Integration mit dem CIM sowie weiterer Modelle lohnenswert (siehe hierzu auch [UG07]). Aufgrund des designorientierten Forschungsansatzes kommt der Evaluation der Ergebnisse eine besondere Bedeutung zu. Hierbei werden insbesondere analytische Methoden betrachtet.[29] Eine ontologiebasierte Repräsentation des RMKs wird zur Evaluation der Struktur als sinnvoll erachtet, hierzu ist die Arbeit von Fettke zur Referenzmodellevaluation auf Basis ontologischer Gütekriterien [Fet06] relevant (R3). Weiterhin sind die Ansätze zur Unternehmensmodellierung mittels Ontologien von Green und Rosemann in [GR05] betrachtenswert. Auf Basis regelmäßiger Workshops wird der Katalog schrittweise entwickelt, angewendet und mit Experten aus Fach- und IT-Abteilungen sowie mit Softwareproduktverantwortlichen diskutiert. Weiterhin sind Veröffentlichungen des Modells und die Diskussionen mit weiteren Domänenexperten aus wirtschaftlichen (bspw. EDNA, www.edna-initiative.de) und wissenschaftlichen (bspw. GI-AK Energieinformationssysteme) Fachgruppen sowie Standardisierungsgremien (bspw. DKE[30]) geplant (R7).

Hieraus ergibt sich für die Entwicklung des Energie-RMKs das in Abbildung 7 dargestellte Vorgehen.

[25] Siehe hierzu Informationen des TeleManagement Forum unter www.tmforum.org.

[26] Ein weiterer zentraler Begriff in der Referenzmodellierungsforschung ist der Ordnungsrahmen, wobei ein Ordnungsrahmen selbst ein Referenzmodell sein kann. Im Rahmen dieses Beitrags wird unter einem Ordnungsrahmen nach [Mei01] ein Modell auf sehr hoher Abstraktionsebene verstanden, das die Navigation durch Modelle auf niedrigerer Abstraktionsebene erlaubt.

[27] Unter einer Ontologie wird nach [Gru93] eine explizite formale Spezifikation einer gemeinsamen Konzeptualisierung verstanden.

[28] Das CIM stellt ein Datenmodell dar, welches Objekte für den Bereich der Energiewirtschaft sowie deren Relationen Untereinander darstellt [UG07]. Das CIM ist bisher im nordamerikanischen Raum, insbesondere zum Austausch von Stromnetzmodellen, weit verbreitet.

[29] Eine Übersicht zu Evaluationsmethoden liefert Hevner et al. [HMPR04], in dem die fünf Kategorien *observational*, *analytical*, *expiremental*, *testing* und *descriptive* unterschieden werden.

[30] Deutsche Kommission Elektrotechnik Elektronik Informationstechnik, www.dke.de.

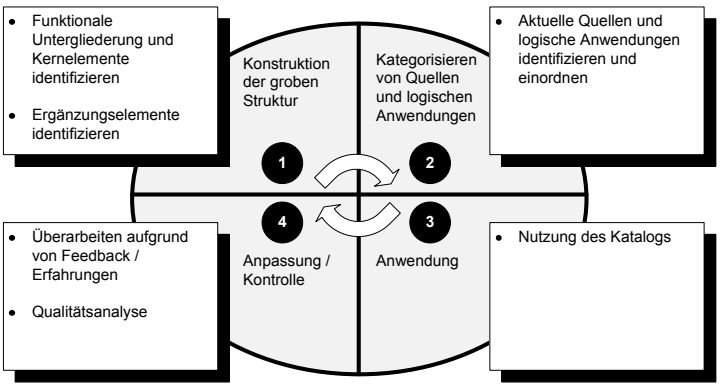


Abb. 7: Vorgehen bei Konstruktion und Anwendung des Energie-RMKs

Das Vorgehensmodell orientiert sich an den vier Phasen[31] zur Referenzmodellkatalogkonstruktion nach [FL02] und besteht selbst aus vier Phasen, ”Konstruktion der groben Struktur” – hier gilt es auf Basis von Kern- und Ergänzungselementen, eine initiale Struktur zu erarbeiten – , ”Kategorisieren von Quellen und logischen Anwendungen” – Einordnen von Quellen –, ”Anwendung” – Einsatz des Katalogs in Projekten – und ”Anpassung / Kontrolle” – Überarbeiten der Struktur und Qualitätsanalyse. Die Phasen sollen jeweils iterativ durchlaufen werden, um auf diese Weise überarbeitete Versionen des RMKs zu erstellen und die Wissensbasis kontinuierlich zu erweitern.

3.3 Energie-RMK Schema und Ontologie

Wie im vorigen Abschnitt beschrieben erscheint die Modellierung des Energie-RMK als Ontologie im Hinblick auf die Integration weiterer Quellen und Durchführung von Qualitätsanalysen vielversprechend. Weitere Vorteile ergeben sich aus der technologieneutralen Repräsentation und der einfachen Bereitstellung und Abfrage der Wissensbasis mittels Open Source Software. Die Fähigkeiten zur Ableitung von neuem Wissen aus der Wissensbasis (Reasoning) werden als besonders nützlich angesehen, um bei minimaler expliziter Modellierung weitere Beziehungen abzuleiten. Nachfolgend werden zunächst die grundlegenden Beziehungen zwischen den Elementen des RMKs dargestellt und die Modellierung als Ontologie in Form von Klassen (*Class*) und Eigenschaften (*Data* und *Object Properties*) skizziert.

In Abbildung 8 ist eine schematische Darstellung des Energie-RMKs zur Verdeutlichung der Beziehungen zwischen den einzelnen Bestandteilen aufgeführt: FRM (oben links), Sichten (rechts) und Kriterien (unten rechts). Im Zentrum stehen die Funktionen des RMKs (RM[32]-Funktion) die den zentralen Verknüpfungspunkt zu den Kernelementen bilden.

[31] Vorstudie, Erstellung, Anwendung sowie Evaluation und Pflege.

[32] RM - Referenzmodell; wird im Folgenden zur Bezeichnung der Bestandteile des FRMs des Energie-RMKs verwendet.

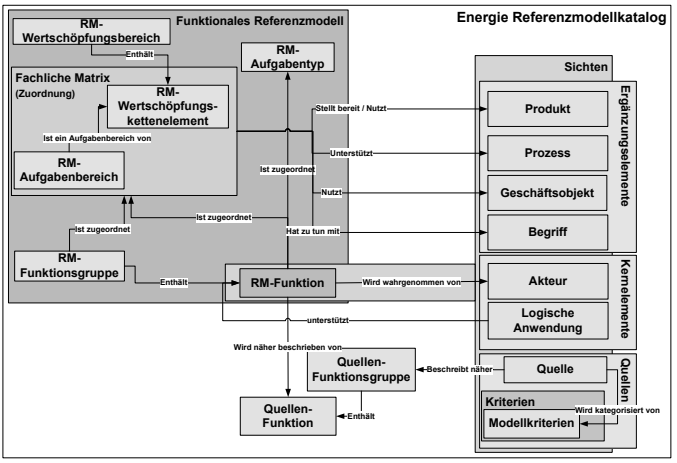


Abb. 8: Energie-RMK Schema

Funktionen werden dabei zusätzlich einem RM-Aufgabentyp zugewiesen. Aufgabentypen entsprechen dabei typischen Aufgabenfeldern aus den Bereichen des Managements[33] , des Produkt[34]- / Dienstleistungs- und Anlagenlebenszyklus[35] in Unternehmen der Energiewirtschaft.[36] Die fachliche Matrix, bestehend aus der Zuordnung von Funktionsmustern (RM-Aufgabenbereichen) zu Wertschöpfungsbereichen, bildet die Verknüpfung zwischen Ergänzungselementen und Funktionsgruppen. Einen weiteren wichtigen Aspekt stellt die Modellierung von Quellen und die entsprechende Zuordnung zu den Funktionen des Katalogs dar. Hier werden aufgrund des damit verbundenen Aufwands und des Grundsatzes der Wirtschaftlichkeit aus den GOM nach Schütte [Sch98] nur als wichtig erachtete Quellen zweistufig als Funktionshierarchie modelliert und auf Funktionsebene mit den Funktionen des RMKs verknüpft.[37] In diesem Fall wird sich die zweistufige Modellierung im Wesentlichen auf deutsche Gesetze und Vorschriften beschränken, da diese für Unternehmen in Deutschland zwingend zu beachten sind. Weitere Quellen werden hingegen nur grob eingeordnet.

Abbildung 9 stellt einen Ausschnitt der Repräsentation des Energie-RMKs als Ontologie dar. Hier sind in Anlehnung an Abbildung 8 Sichten (oben links), Modellkriterien (oben rechts) und Verknüpfungen von Quellen zu Funktionen (unten) angedeutet. Im Fokus steht auch hier die RM-Funktion die über die *ObjectProperty wirdBeschriebenIn* mit der *Quellen-Funktion* verbunden ist. Ferner ist die Abbildung von Funktionen des funktionalen Referenzmodells und der Quellen mit der zweistufigen Hierarchie über die Klassen Funktionsgruppen und Funktionen mit ihren Unterklassen und entsprechenden *Object-*

[33] Bspw. Planen, Realisieren, Kommunizieren, Kontrollieren nach dem Management-Kreis siehe [Sch72].

[34] Bspw. Beschaffen, Produzieren, Handeln, Vermarkten und Bereitstellen.

[35] Bspw. Planen, Bauen, Inbetriebnehmen, Überwachen, Betreiben, Instandhalten und Rückbauen.

[36] Durch die Anwendung von typischen Mustern soll eine möglichst vollständige Abdeckung der betrieblichen Funktionen erreicht werden.

[37] Es werden nur Quellen abgebildet, die auch die Funktionssicht adressieren, Organisationsmodelle werden bspw. nicht als funktionale Hierarchie modelliert.

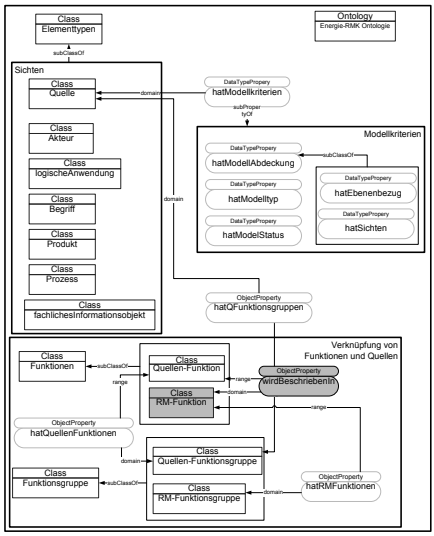


Abb. 9: Auszug aus der Energie-RMK Ontologie

Properties angedeutet. Die Abbildung der Sichten als Klassen und der Modellkriterien als *DataTypeProperty*-Hierarchie ist ebenfalls skizziert.

3.4 Einordnung des gewählten Ansatzes

Der vorliegende Beitrag greift die in [Gon09] skizzierten Ansätze auf und fokussiert diese im Hinblick auf die Entwicklung eines Referenzmodellkatalogs für die Energiewirtschaft. Ferner wurden die in den Arbeiten von [PGS09] und [BGPA09] gewonnenen Erfahrungen bei der Nutzung von Bestandteilen des Energie-RMKs berücksichtigt.
In Bezug auf die Konstruktion eines Referenzmodellkataloges lehnt sich der hier beschriebene Beitrag stark an [FL02] sowie den GOM nach [Sch98] an und folgt dem methodischen Vorgehen nach [HMPR04]. Bezogen auf die Klassifikation von Quellen für den RMK werden bewährte Klassifikations-Schemata nach [Sch98], [vB03] und [dV06] herangezogen.

4 Zusammenfassung und Ausblick

Im Rahmen dieses Beitrags wurden ein Ansatz und erste Ergebnisse zur Konstruktion eines Referenzmodellkatalogs für die Energiewirtschaft vorgestellt und die Modellierung des Energie-RMKs als Ontologie skizziert. Der vorgestellte Ansatz befindet sich noch in der Weiterentwicklung, erste Anwendungen des RMKs sind schon erfolgt, siehe [PGS09] und [BGPA09]. Ein wichtiger Aspekt der weiteren Arbeit stellt die Anwendung und Evaluierung des Katalogs dar. Die hieraus resultierenden Rückkopplungen und Diskussionen mit Domänenexperten werden als wichtig angesehen, um einen qualitativ hochwertigen

und nutzbaren Referenzmodellkatalog zu erarbeiten und eine Akzeptanz für den Energie-RMK innerhalb der Domäne zu erreichen.

Danksagung: Dieser Beitrag ist im Rahmen eines durch die EWE AG geförderten Projektes entstanden.

Literaturverzeichnis

[AC07] H.-J Appelrath und P. Chamoni. Veränderungen in der Energiewirtschaft-Herausforderungen für die IT. *WIRTSCHAFTSINFORMATIK*, 49(5), 2007.

[Ban08] Ralf Banning. Modellgetriebene Integration in der Energiewirtschaft: Softwaredesign zwischen Handwerk und Industrie? 11. EDNA Fachtagung in Frankfurt, 20.03.2008.

[BBB+08] C. Block, F. Bomarius, P. Bretschneider, F. Briegel, N. Burger, B. Fey, H. Frey, J. Hartmann, C. Kern, B. Plail, G. Praehauser, L. Schetters, F. Schöpf, D. Schumann, F. Schwammberger, O. Terzidis, R. Thiemann, C. van Dinther, K. von Sengbusch, A. Weidlich und C. Weinhardt. Internet der Energie - IKT für die Energiemärkte der Zukunft, 2008.

[BBF+08] Bernd Buchholz, Volker Bühner, Hellmuth Frey, Wolfgang Glaunsinger, Martin Kleimaier, Magnus Pielke, Hans Roman, Johannes Schmiesing, Johannes Stein, Zbigniew Styczynski und Hartmut Baden. Smart Distribution 2020: Virtuelle Kraftwerke in Verteilungsnetzen: Technische, regulatorische und kommerzielle Rahmenbedingungen, 26.06.2008.

[BGPA09] Petra Beenken, José M. González, Matthias Postina und H. J Appelrath. Sicherheitsorientierte Gestaltung von Anwendungslandschaften in der Energiewirtschaft. In *Internationaler ETG-Kongress 2009*. 2009.

[Bic05] Martin Bichler. Design Science in Information Systems Research: von Alan R. Hevner, Salvatore T. March, Jinsoo Park, Sudha Ram in MIS Quarterly, Vol. 28, No. 1, S. 75-105, März 2004: referiert von Prof. Dr. Martin Bichler, 15.11.2005.

[BS04] Jörg Becker und Reinhard Schütte. *Handelsinformationssysteme: Domänenorientierte Einführung in die Wirtschaftsinformatik*. Redline Wirtschaft, Frankfurt am Main, 2., vollst. überarb., erw. und aktualis. Auflage, 2004.

[Bun07] Bundesverband der Energie- und Wasserwirtschaft (BDEW). Energiemarkt Deutschland: Zahlen und Fakten zur Gas- und Stromversorgung, 18.12.2007.

[Bun08a] Bundesministerium für Wirtschaft und Technologie. E-Energy - Informations- und kommunikationstechnologiebasiertes Energiesystem der Zukunft: Ein Förderwettbewerb des Bundesministeriums für Wirtschaft und Technologie, 2008.

[Bun08b] Bundesministerium für Wirtschaft und Technologie. Endenergieverbrauch nach Energieträgern: Deutschland: Quelle: Arbeitsgemeinschaft Energiebilanzen, Stand: Aug. 2008; letzte Änderung: 08.10.2008, August 2008.

[Deu05] Deutscher Bundestag. Gesetz über die Elektrizitäts- und Gasversorgung (Energiewirtschaftsgesetz - EnWG): EnWG, 2005.

[Deu08] Deutscher Bundestag. Erneuerbare-Energien-Gesetz (EEG, 2009), 25.08.2008.

[dV06] Henk J. de Vries. IT Standards Typology. In Kai Jakobs, Hrsg., *Advanced topics in information technology standards and standardization research*, Seiten 1–26. Idea Group Pub., Hershey PA, 2006.

[Ede08] Helmut Edelmann. Wettbewerb in den Energiemärkten: Stadtwerkestudie 2008: Management Summary, 05.06.2008.

[EHH+08] Gregor Engels, Andreas Hess, Bernhard Humm, Oliver Juwig, Marc Lohmann, Jan-Peter Richter, Markus Voß und Johannes Willkomm. *Quasar enterprise: Anwendungslandschaften serviceorientiert gestalten*. dpunkt.Verl., Heidelberg, 1. Auflage, 2008.

[Ele09] Report to NIST on the Smart Grid Interoperability Standards Roadmap: Contract No. SB1341-09-CN-0031—Deliverable 10: Post Comment Period Version Document, August 2009.

[Fet06] Peter Fettke. *Referenzmodellevaluation: Konzeption der strukturalistischen Referenzmodellierung und Entfaltung ontologischer Gütekriterien*, Jgg. 5. Logos-Verl., Berlin, 2006.

[FL02] Peter Fettke und Peter Loos. Der Referenzmodellkatalog als Instrument des Wissensmanagements - Methodik und Anwendung. In Jörg Becker und Ralf Knackstedt, Hrsg., *Wissensmanagement mit Referenzmodellen. Konzepte für die Anwendungssystem- und Organisationsgestaltung*, Seiten 3–24. Springer, Berlin et al., 2002.

[FL04] Peter Fettke und Peter Loos. Referenzmodellierungsforschung: Langfassung eines Aufsatzes. In Peter Loos, Hrsg., *Working Papers of the Research Group Information Systems & Management*, number 16. Juli 2004.

[Gab09] Gabler Wirtschaftslexikon. Energiewirtschaft, 2009. Online verfügbar unter http://wirtschaftslexikon.gabler.de/Archiv/128894/energiewirtschaft-v1.html, zuletzt geprüft am 15.10.2009.

[Gon09] José M. González. Gestaltung nachhaltiger IT-Landschaften in der Energiewirtschaft mit Hilfe von Referenzmodellen. In Torsten Eymann, Hrsg., *Bayreuther Arbeitspapiere zur Wirtschaftsinformatik*, Jgg. 40, Seiten 35–44. Februar 2009.

[GR05] Peter F. Green und Michael Rosemann. *Business systems analysis with ontologies*. Idea Group Publ., Hershey Pa., 2005.

[Gru93] Thomas R. Gruber. A Translation Approach to Portable Ontology Specifications. In *Knowledge Acquisition*, Jgg. 2, Seiten 199–220. 1993.

[HMPR04] Alan R. Hevner, Salvatore T. March, Jinsoo Park und Sudha Ram. Design Science in Information Systems Research. *MIS Quaterly*, 28(1):75–105, 2004.

[Int07] International Electrotechnical Commission (IEC). IEC 61968-1: Application integration at electric utilities – System interfaces for distribution management – Part 1: Interface architecture and general requirements, 11.04.2007.

[Kel07] Wolfgang Keller. *IT-Unternehmensarchitektur: Von der Geschäftsstrategie zur optimalen IT-Unterstützung*. dpunkt.Verl., Heidelberg, 1. Auflage, 2007.

[Mei01] Volker Meise. *Ordnungsrahmen zur prozessorientierten Organisationsgestaltung: Modelle für das Management komplexer Reorganisationsprojekte*, Jgg. 10. Kovac, Hamburg, 2001.

[NGN06] Per Närman, Magnus Gammelgard und Lars Nordström. A Functional Reference Model For Asset Management Applications Based on IEC 61968–1. In *Nordic Distribution and Asset Management Conference*. 2006.

[PBL05] Klaus Pohl, Günter Böckle und Frank Linden. *Software Product Line Engineering: Foundations, Principles, and Techniques /// Software product line engineering: Foundations, principles, and techniques ; with 10 tables*. Springer, Berlin, 2005.

[PGS09] Matthias Postina, José M. González und Igor Sechyn. On the Architecture Development of Utility Enterprises with Special Respect to the Gap Analysis of Application Landscapes. In Ulrike Steffens, Jan S. Addicks, Matthias Postina und Niels Streekmann, Hrsg., *MDD, SOA und IT-Management (MSI 2009)*, Seiten 17–31. Gito, 2009.

[Sch72] U. Schubert. Der Management-Kreis. In *Management für alle Führungskräfte in Wirtschaft und Verwaltung*, Jgg. 1, Seiten 36–49. 1972.

[Sch98] Reinhard Schütte. *Grundsätze ordnungsmäßiger Referenzmodellierung: Konstruktion konfigurations- und anpassungsorientierter Modelle*, Jgg. 233. Gabler, Wiesbaden, 1998.

[Sch02] August-Wilhelm Scheer. *ARIS – vom Geschäftsprozess zum Anwendungssystem*. Springer, Berlin, 4., durchges.. Auflage, 2002.

[Tho06] Oliver Thomas. Das Referenzmodellverständnis in der Wirtschaftsinformatik: Historie, Literaturanalyse und Begriffsexplikation: Heft 187, Januar 2006.

[UG07] Mathias Uslar und Fabian Grüning. Zur semantischen Interoperabilität in der Energiebranche: CIM IEC 61970. *WIRTSCHAFTSINFORMATIK*, 4(49):295–303, 2007.

[URS+09] Mathias Uslar, Sebastian Rohjans, Tanja Schmedes, José M. González, Petra Beenken, Tobias Weidelt, Michael Specht, Christoph Mayer, Astrid Niesse, Jens Kamenik, Claas Busemann, Karlheinz Schwarz und Franz Hein. Untersuchung des Normungsumfeldes zum BMWi-Förderschwerpunkt "e-Energy-IKT-basiertes Energiesystem der Zukunft": Studie für das Bundesministerium für Wirtschaft und Technologie (BMWi), 03.2009.

[vB03] Jan vom Brocke. *Referenzmodellierung: Gestaltung und Verteilung von Konstruktionsprozessen*, Jgg. 4. Logos, Berlin, 2003.

A Domain Specific Language for Multi User Interface Development

Alexander Behring[1], Andreas Petter[2] and Max Mühlhäuser[3]

Abstract: User Interface Development is increasingly facing the demand that an application must provide different User Interfaces (UIs) for different contexts of use, e.g., interaction device and primary task. This leads to two key challenges: how to create these multiple UIs for one application (creation challenge), and how to consistently modify them (modification challenge). The creation challenge has been addressed in various works utilizing automatic UI generation. We present a domain specific language (DSL) suitable to address the modification challenge. The DSL makes use of explicit relations between different UI versions, along which modications of the UIs can be propagated. With the presented approach, modifications can be applied more easily, which is important for iterative (UI) design.

1 Introduction

Technological advances allow us to interact with computing platforms in a great variety of situations. Form factors of devices are getting smaller and the devices are more and more affordable for a bigger audience. We are currently seeing an increase in the number of user interfaces an application has to provide. Commercial services already are providing different interface versions for one application – not only for desktop computers but also for mobile devices. Examples beyond simple train, airplane or bus schedule services are electronic boarding-passes and the possibility to buy electronic public transportation tickets for and with mobile devices.

These user interfaces (*UI*s) differ depending on the *context of use*. Characteristics of the user, the used computing platform (incl. software) and the environment make up the context of use. We take over this notion from Calvary et al. [CCD+04]. On an abstract level, we divide the development of such UIs for multiple contexts of use into two challenges: the creation challenge, and the modification challenge.

The **creation challenge** focuses on the initial creation of a UI. The user interface to be created can be build on the green grass. On the other hand, for the **modification challenge** it is assumed that an existing UI is to be updated. To cope with the updating and allow the developer to control how the existing UI is updated, is the issue for the modification challenge.

[1] FG Telecooperation, TU Darmstadt, 64283 Darmstadt, Germany, behring@tk.informatik.tu-darmstadt.de
[2] FG Telecooperation, TU Darmstadt, 64283 Darmstadt, Germany. a_petter@tk.informatik.tu-darmstadt.de
[3] FG Telecooperation, TU Darmstadt, 64283 Darmstadt, Germany, max@tk.informatik.tu-darmstadt.de

Our approach to address the modification challenge is based on the key idea to allow one modification to change multiple UIs. In order to control the propagation of a modification, UIs are ordered in a tree. The modification is "passed" down the tree along associations between the UIs. Changing these associations allows the UI engineer to adapt the propagation of modifications to suite the problem at hand. The unique feature of our domain specific language (*DSL*) is that its application constructs this tree and thus allows one modification to be applied to multiple UIs.

In this paper, we present a DSL to model user interfaces and address the modification challenge. The DSL is based on a set of basic requirements, presented in the following. In section 2, the concepts of the DSL (its semantic) is introduced. In the subsequent section, the metamodel (abstract syntax), expressed in EMF (Eclipse Modeling Framework) [4] is described. Constraints (well-formedness rules) presented thereafter also belong to the DSL and ensure the validity of the metamodel instances. Finally, the use of the DSL, with its concrete syntax, and its integration in tools is laid out. We conclude with a discussion of the presented work and related approaches.

1.1 Requirements

The DSL was designed based on a set of four basic requirements, presented in this section. Further discussion of the topic can be found in [BPFM08, BPM09].

Requirement 1: *UI engineers provide information at the level of abstraction suitable for the problem at hand.*

The Model Driven Architecture (*MDA*) also pursues this goal [KUW02]. Especially, changes to the UI can be situated at different levels (e.g., for all UIs or just for the iPhone version, cf. figure 1). Hereby, the *1.1) number of abstraction levels* (cf. section 2) suitable for a UI may vary depending on domain and application. Furthermore, the *1.2) nature of abstraction* shall not be fixed: the developer should be free to choose whether to abstract from, e.g., user characteristics, the computing platform or the UI toolkit.

Requirement 2: *The approach must easily be extendable to new UI toolkits.*

In order to not limit the scope of the approach, easy extension to new UI toolkits is an important property, as Myers et al. note in [MHP00].

Requirement 3: *Support full control over the UI look and feel for UI engineers.*

[4] http://www.eclipse.org/emf

Automatic approaches hold great potential to increase efficiency, but their interfaces are prone to usability problems and lack aesthetic quality[5] [MHP00, MVLC08, DMLC08]. Thus, *3.1) the UI engineer must be able to manually modify all UIs* (also generated ones), and *3.2) be in control of how modifications are applied*. Furthermore, according to Myers et al [MHP00], the control over the "low-level pragmatics of the interactions look and feel" is important for UI engineers. Therefore, *3.3) the approach should rather focus this detailed control* than on the commonalities of different UI descriptions.

Requirement 4: *Conceptualize for ease of use for UI engineers.*

Ease of use is crucial for the adoption of an approach. Especially a *4.1) closest-as-possible resemblance between edited artifact and resulting UI* guarantees that the UI engineer is not working on abstract artifacts that isolate her from the concrete interface. It lowers the threshold of use and reduces unpredictability problems. Myers et al note these problems in [MHP00] and conclude that this was one of the reasons, why User Interface Management Systems (UIMS) did not catch on.

2 Modeling Concepts (DSL Semantic)

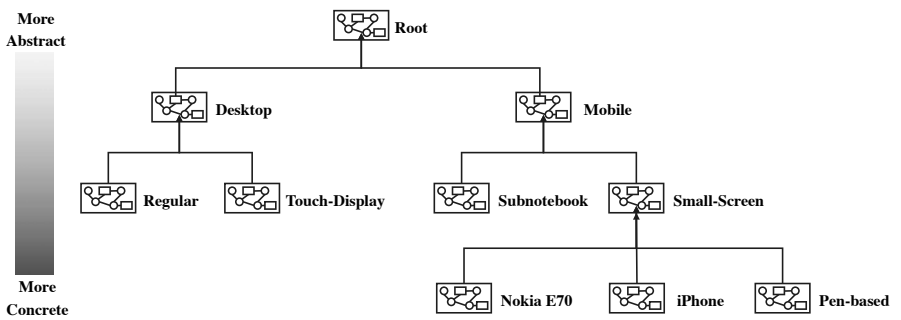


Fig. 1: An exemplary *refinement tree* showing multiple levels of refinement. More abstract UI models (at the top) are refined to more concrete versions.

This section introduces, after a brief overview, the concepts used in the DSL (i.e. its semantic), before the corresponding metamodel (abstract syntax) is presented in the next section.

When creating a UI for a new context of use, the UI engineer bases the new UI on an already existing one (she refines it). This new **UI refinement** can be modified freely (e.g., elements added, removed, and properties changed) to suite the new context of use at hand. Along with the new refinement, associations between the two UIs and their constituents are created to document the refinement relationship of the new refinement and the UI it is based on. Applying this refinement step for multiple contexts of use creates a tree of

[5] Of course depending on the definition of aesthetic quality and the degree of automatization.

UIs ordered according to the refinements made (cf. figure 1). Inside each UI, Interaction Objects (introduced below) describe the UIs look and feel. To apply a modification to multiple UIs, it is passed along the associations between the different UIs. The UI engineer can influence this propagation by changing the refinement associations between the UIs.

The next sections discuss the elements to describe a UI in detail (Interaction Objects), how they are classified (Interaction Object Class) and how refinement is accomplished in our approach.

2.1 Interaction Objects

An *Interaction Object* is an entity representing a form of interaction between the user and the application. They can appear in a concrete form (Concrete Interaction Objects), which Users can perceive and/or manipulate, e.g., a combo box, a GUI button, a sound or a button on an interaction device. They also appear in an abstract form (Abstract Interaction Objects) abstracting away from one or many concrete representation, e.g., a select-one-of-n or an action invocation. Interaction Objects are the basic building block used in our approach for UI modeling, they are instantiated to model the user interface.

In our opinion, a clear division into Abstract and Concrete Interaction Objects is infeasible. A concrete element like an HTML Combo Box for example has different representations on different platforms, e.g., Mac OS Safari versus Windows Internet Explorer. Elements of the XForms toolkit[6] (e.g., select, group and submit) can be mapped to different representations. The distinction between abstract and concrete thus becomes blurry: given the set of an AUI element (CTT approach, [MPS04]), an HTML element, an XForms element and a Swing element, this set cannot clearly be divided into abstract and concrete elements – where do you draw the line? As further an arbitrary number of refinements steps is required (requirement 1), we do not use the distinction into concrete and abstract, but allow Interaction Objects at arbitrary abstraction levels to be used.

2.2 Interaction Object Class

Similar to UML Classes [Obj05b], Interaction Objects must be assigned a type. The type can, e.g., be HTML Combo Box, XForms Select or Swing JPanel. In order to type an Interaction Object, it is assigned an Interaction Object Class. The class implies the attributes an Interaction Object can have.

Interaction Object Classes are organized in libraries, allowing an easy extension to new toolkits (requirement 2). A library provides all classes relevant to support a given UI toolkit (e.g., Swing, SWT or XForms). Hereby, inheritance relationships between the classes can be modeled, as well. The specializing class inherits all attributes of the generalized[7] class.

[6] http://www.w3.org/MarkUp/Forms/

[7] Please note that we use the term *generalized* in the context of class inheritance, in order to distinguish it from *abstracted*, as used in the context of refinement.

2.3 Refinement and User Interface Boxes (UI Box)

The user interface adapted to a given context of use is defined by arranging Interaction Objects in a UI Box. The UI Box is a container, which has a context of use assigned to it.

To specialize a UI for a new context of use, a more abstract UI is refined. For example, the iPhone UI in figure 1 refines the more abstract Small-Screen UI. Hereby, the UI engineer modifies the UI to make it more specific for the new context of use. Repeated refinement results in a *Refinement Tree* of UIs, as shown in figure 1. The nodes in the figure represent user interface models (UI Boxes) for different contexts of use. UIs in the tree are ordered from abstract to more concrete (refined) UIs, with a single topmost (*root*) UI. The more concrete (*refining*) version refine the more *abstract* versions.

This refinement allows the UIs to be at any level of abstraction (cf. our remark on Abstract and Concrete Interaction Objects). The UI engineer can build refinement trees as deep as needed – no number of refinement levels is prescribed – and thereby provide the information at the level of abstraction suitable (requirement 1). This includes choosing the nature of abstraction. For example, whether to abstract from a specific platform, or a user characteristic. Our approach thus allows the refinement as needed by the concrete problem at hand.

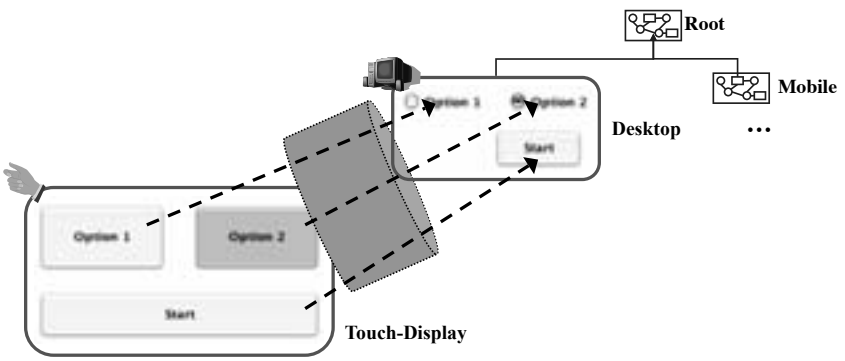


Fig. 2: Refinement associations on the UI Box level (big dotted tube) guard (“tube”) the refinement associations on Interaction Objects level (dashed arrows), as formulated in the Nesting Consistency Constraint (cf. section 3.1.5).

The key idea to address the modification challenge, is to allow one modification to change multiple UIs (connected in the Refinement Tree). Edges connecting the UIs for the different contexts of use are called *Refinement Associations*. Their semantic is ”concretization for a given context of use”. Besides the association on UI level, more detailed information on Interaction Object level is needed to unambiguously identify which elements are refining each other. The Interaction Objects themselves are connected via refinement associations, too, as illustrated in figure 2. Using this concept, one modification can change multiple UIs at once by being propagated along the refinement associations.

When applying a modification in a UI, it can either affect *i)* the properties of Interaction Objects, or *ii)* add or remove Interaction Objects. Property modifications are propagated,

as retained in the model by the UI engineer through the use of refinement associations. These associations can be interpreted as consistency conditions: if refinement is in place, modifications occur in all refining UI versions – the refinement association describe what features of the UIs are automatically kept consistent. The latter modification to add and remove Interaction Objects is of a different nature and thus supported differently. In our opinion, it can only be addressed through also providing adequate tool support.

3 Metamodel for the Domain Specific Language (Abstract Syntax)

Based on the concepts of the previous section, we developed a metamodel (abstract syntax) for describing user interfaces. It is formulated using EMF (Eclipse Modeling Framework), a framework for model-driven development in the Eclipse [8] project. EMF was used to produce model code – to be able to instantiate metamodel instances at runtime – and build editors to create and modify user interface models using the presented approach.

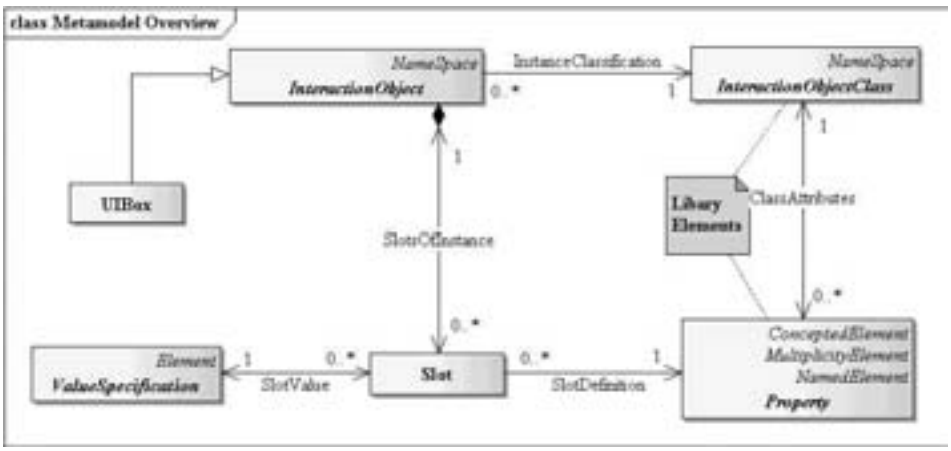


Fig. 3: Overview of the metamodel. An Interaction Object can have properties, must be typed, and can be nested. Interaction Object Classes are used to type Interaction Object and define which properties an Interaction Object can have.

In figure 3, an overview of the metamodel is given. Interaction Object Classes are represented by an element of the same name. Classes can have properties by attaching them via the Class Attributes association. A Property can only belong to one class. Both, Interaction Object Classes and Properties, are held in libraries and are referenced by Interaction Object and Slots. These references are not bidirectional, libraries are independent of UI descriptions (Interaction Objects in UI Boxes).

The actual UI model consists of instances of UI Boxes, Interaction Objects, ValueSpecifications and Slots (depicted in the middle and the left side of figure 3). An Interaction Object references its *classification* via the association Instance Classification. Assigning *property values* is similar to UML [Obj05b]: Slots mediate Values Specifications with

[8] http://www.eclipse.org

Properties. A Slot references the Property it sets a value for, as well as the Value Specification (i.e. the value) to be set. Value Specifications and Properties can have Slots associated with them, but do not have to (e.g., no value is set).

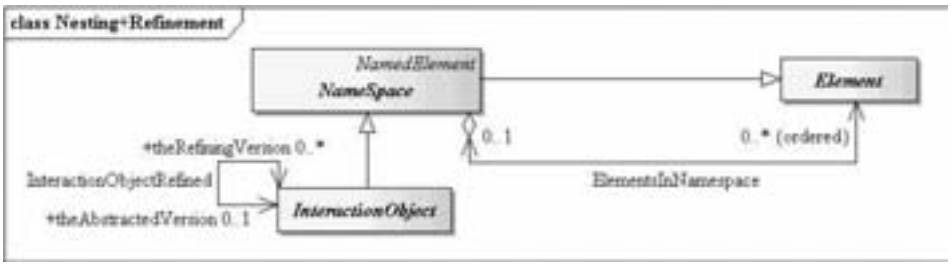


Fig. 4: Implementation of the nesting and refinement features in the metamodel.

The *nesting* of Interaction Objects is illustrated on the right side of figure 4: an Interaction Object inherits from Namespace, which can contain Elements. In turn, Interaction Objects are Elements. In the same figure (left side), our implementation of *refinement* is depicted: an Interaction Object refines another Interaction Object via the *Interaction Object Refined* association. Hereby not only Interaction Objects can refine each other, but also UI Boxes, as they inherit from Interaction Object. Furthermore, the *refinement association is transitive*: if a property value is looked up in the more abstract version and no value is found, the next more abstract version will be consulted and so on (cf. figure 5).

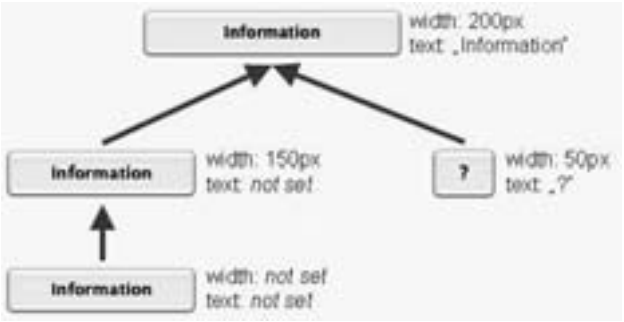


Fig. 5: Example illustrating the refinement of properties. The refinement on the lower left inherits its width from the more abstract version and its text from the topmost version.

Refinement of Properties is implemented without a special association. We assume that if an interactor does not set a property value, the value of its abstract version should be used. So the critical question is, whether the abstract version's property can be unambiguously identified from the refining version's property. It can, because the abstract version has the same property and thus, its value can be accessed and used. Using this approach, a property can either be:

set locally,	i.e. the Interaction Object has a value set for the property, or
refined,	when it has no value set.

Multiple refinement, similar to multiple inheritance, is not supported in order to allow consistency by construction: an Interaction Object can only refine 0..1 abstract versions.

3.1 Constraints and Formalism (Well-Formedness Rules)

To further clarify the use and well-formedness of an instance of the presented metamodel, constraints are used. In this section, constraints relevant for our approach are presented. The first two constraints are independent of the refinement itself, whereas the latter constraints are all targeted at providing a consistent Refinement Tree.

3.1.1 Library Consistency

When interpreting a UI, and rendering it, toolkits cannot be mixed (e.g., an HTML Combo Box inside a Swing JPanel). We call the corresponding constraint the Library Consistency Constraint. It consists of two sub-constraints: *i)* the library itself may only contain elements of one UI toolkit and *ii)* all elements in a UI Box must belong to one library. Only when both sub-constraints are satisfied, the Library Consistency Constraint holds.

Using our metamodel, we collect all elements belonging to one library inside a library-element using an ownership-association (not in the scope of this paper). The first sub-constraint is fulfilled, if all elements owned by a library element indeed belong to the same UI toolkit. To fulfill this, the library itself has to be specified correctly, which cannot be checked on model-level, but must be done by externally (by a human).

The second sub-constraint is fulfilled, if all elements nested inside a UI Box are typed by classes that all belong to the same library, i.e., all these Interaction Object Classes are owned by the same library element. In contrast to the previous sub-constraint, this can be checked on model-level.

3.1.2 Property Resolvability

As modeling properties in the presented metamodel is based on UML, a similar constraint must be enforced. We adapt the UML constraint [Obj05b], pg. 127, to ”a Slot specifies the value of its defining Property, which must be a Property of an Interaction Object Class of the Interaction Object owning the Slot.” This can be illustrated in figure 3: when following associations from a Slot element, on the one hand via the Property element, on the other hand via an Interaction Object, the same Interaction Object Class must be met.

3.1.3 Circular Refinements

In order for the presented concepts to work, the tree structure of refinement associations on UI Box and Interaction Object level must be preserved. This means that no circular

refinements may exist. For all Interaction Objects, it must thus hold that for every Interaction Object they are refined by, they may not be the refinement of. Hereby, refinement has to be considered *transitively*: if Interaction Object A refines Interaction Object B and Interaction Object B refines Interaction Object C, Interaction Object A refines (transitively) Interaction Object C.

When regarding transitivity of refinement between UI Boxes, three possible relations between two UI Boxes in a Refinement Tree exist. UI Box *A* relates to UI Box *B*:

refined,	*A* (transitively) is the refinement of *B*,
abstracted,	*B* (transitively) is the refinement of *A*, and
related,	*B* and *A* (transitively) refine the same UI Box *C* (not identical to *A* or *B*).

When regarding Interaction Objects that are not UI Boxes, a fourth type is possible: *unrelated*. That is, two Interaction Objects are not connected via refinement associations. Taking a different view point, we can also note that for a given Interaction Object that is not a UI Box, there is not necessarily a related Interaction Object in the other UI Boxes.

3.1.4 Refinement Condition

Naturally, not all combinations of refining Interaction Objects are possible when refining. Trying to refine an HTML Combo Box to a Swing JPanel must lead into inconsistencies; *i)* their purpose is different, the combo box is used to select an item, the panel to group elements, and *ii)* their properties are incompatible. Checking whether both Interaction Objects serve the same purpose is currently ongoing research in our group, but the property compatibility can be formulated: an Interaction Object B can only refine an Interaction Object A, if matching properties of both Interaction Objects (as they are defined by their classes and the classes generalizations) are compatible. Compatible means that the types of the properties can be converted into each other. Matching means that the properties have the same name.

If we assume that upward modification propagation is not possible, i.e., the UI engineer cannot modify a UI and expect more abstract UIs to reflect this modification, compatibility can be specialized. Compatible then means that all possible values of the property of *A* can be converted to values of the property of *B*.

3.1.5 Nesting Consistency

When refining a UI Box, the nesting order of the affected Interaction Objects should not be changed. For example, a panel contained inside a tab should not be changed to the tab being inside the panel in the refined version. This would produce inconsistent semantics of the groupings (”why elements are grouped in the panel, what is their commonality”). Note that, on the other hand, insertion and deletion of Interaction Object must be allowed.

Figure 6 illustrates this: the lowest element in UI Box A is removed, whereas the topmost element in UI Box B is added (neither has a refinement partner in the other UI).

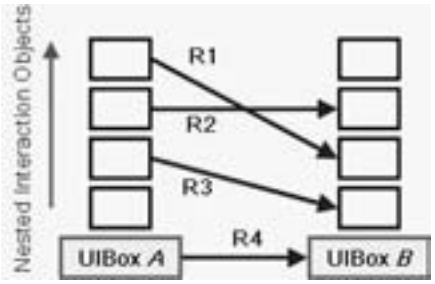


Fig. 6: Illustration of a violation of the Nesting Consistency Constraint. Arrows depict refinement associations (R1 – R4), nested Interaction Objects (small boxes) sit on top of their parents. The crossing of R1 and R2 violates the constraint.

Thus, the constraint is formulated: for a refinement of UI Box *A* to UI Box *B*, for every nested element X in *A* it must hold that the refinements of X's nested element are not parents of X's refinement. A violation of the *Nesting Consistency* constraint is illustrated in figure 6.

3.2 Integration of the Domain Specific Language and Concrete Syntax

Besides a clear definition of the metamodel (abstract syntax) and its constraints (well-formedness), it is important to consider the use of the DSL, especially it's appearance to the UI engineer (concrete syntax). Especially in the area of UI development, adequate tool support is crucial, otherwise, the approach will not survive (for some examples, see [MHP00]).

The UIs created and modified using the presented metamodel can either be transformed into code or an intermediary artifact (*generator approach*) or be directly interpreted (*runtime interpretation*), as noted in [Pin00]. We chose the interpreter approach, because using the generator approach, code and model have to be kept synchronized (known as the *round-trip problem* [HT06]). Using the interpreter approach, model executability is achieved automatically. This facilitates an experimental approach to learning the use of the language and more direct feedback [Sel03]. The UI engineer can more quickly evaluate changes to the UI, which is beneficial for them [DRO07], especially in iterative processes. [9]

For editing models, we put forward the use of WYSIWYG editors (cf. requirement 4). Myers et al. identified in [MHP00] user interface builders as a successful tool approach and noted that their advantage is the conceptual match of manipulating graphical interfaces by graphical means. Furthermore, direct feedback on changes and experimentation is possible. Thus, the concrete syntax in our case is determined by the implementation of the Swing UI toolkit, or – more general – by the editor used.

[9] Regarding performance, our current implementation of a Swing interpreter shows only very small delay when loading and initially interpreting the UI, while the interaction with the UI has no perceivable delays.

The refinement concept presented in this paper has been implemented in Eclipse-based tools. A WYSIWYG-editor and a special Eclipse view to investigate refinement associations were created. Part of the tools can be seen in figure 7. The interested reader can refer to [BPM09] for more details.

Fig. 7: The interpreter and editor (right side) together with the synchronized Eclipse properties page. The property page shows the currently selected item (button "Find an exhibit").

4 Discussion

After implementing some first case studies, we conducted a conceptual user study to collect feedback on the presented approach. Seven participants with a great variance in GUI building skills (ranging from no experience to very skilled interface builder users) had to complete two tasks. They were given a set of ready-made UIs they had to modify. Task one consisted of replacing text in all UIs, whereas task two was to add tooltips to existing elements. After a brief introduction, the participants were asked to complete both tasks without the refinement concept in an interface builder [10], and with utilizing the refinement concept in the Eclipse-based tools.

All participants successfully completed both tasks using the Refinement Associations and had no delays in applying the concept with regard to the use of the interface builder. We can conclude that the concept is easily comprehensible (wrt. industrial-grade UI builders) and thus has a low threshold of use. Furthermore, we discovered through measurements that using the tools with the refinement concept, the error rate was reduced for the tasks: $4\% \pm 6\%$ for our approach versus $21\% \pm 20\%$ for Netbeans. We attributed this to the repetitive and tedious nature when performing the tasks with traditional tools, which was confirmed by the participants feedback. In [BP09], the interested reader can find a detailed description and discussion of the study.

A larger case study was conducted in the context of the SoKNOS project [11]. We integrated the adaptation framework into the SoKNOS portal (UI) and created a SoKNOS plugin to support the users in messaging tasks. Using the approach presented in this paper, we

[10] We used Netbeans for this purpose, because Eclipse does not provide an industrial-grade interface builder, cf. http://www.netbeans.org.

[11] http://www.soknos.de

created various refinements for the plugin's UI (e.g., for special roles and different screen sizes and custom hardware). The presented approach allowed us to solely modify the parts of the UI model that changed, inheriting other parts of the UI layout and behavior from more abstract UI versions.

The work presented uses the **library metaphor**, which has the advantages of a stable and simple core language, as well as maximum flexibility with respect to extensibility [AK05] (requirement 2). New Interaction Object Classifiers do not have to be encoded directly into the metamodel, but can be integrated by adding them via a library. This is an important property, as Myers et al. note in [MHP00] in order to not limit the scope of the approach.

Our approach transfers concepts of MDA [KUW02] to the domain of user interfaces. In MDA, a platform independent model is transformed into a platform specific model. When interpreting the different contexts of use as "platforms" in the MDA sense, the presented refinement approach can be seen as multiple applications of MDA-like transformations. The result is a Refinement Tree (cf. figure 1). This tree has no fixed levels of abstraction, does not constrain the nature of abstraction and thus allows the UI engineer to provide the information at the level of abstraction suitable for the problem at hand (requirement 1).

Using the approach presented, the UI engineer has full control over the UI look and feel (requirement 3). The refinement can be controlled by the developer through modifying the Refinement Associations between UI elements. Furthermore, all aspects of the UI elements (properties, type) can be modified, so that the UI engineer has full control over the UI look and feel. Ease of use for the developer (requirement 4) is addressed through tool support, primarily through WYSIWYG editing, avoiding isolation of the UI engineer from the resulting interface by the use of abstract artifacts. This structured and comprehensible approach to the modification challenge allows UI engineers to easily predict the outcome of their modifications and quickly modify multiple user interfaces. Especially iterative processes will thus benefit.

The modification challenge is specifically addressed by our DSL: using the DSL generates the Refinement Tree, in which modifications can be propagated. This allows to apply one modification to multiple UIs simultaneously.

5 Related Approaches

Other model-based approaches for UI development discuss the concepts used for building metamodels such as the one presented here, but do not provide concrete metamodels [SCF+06, CCT+02]. Another important difference to other works is the number of supported refinement levels (requirement 1). Most other works on model-based UI development allow two levels of abstraction – an abstract plus a concrete level. In contrast, the presented approach supports an arbitrary number of refinement levels. This allows the developer to choose levels of abstraction suitable for the problem at hand. Also, the UI engineer using our approach is free to choose the nature of abstraction, e.g., whether to abstract from a specific platform or user characteristics.

A number of works focusses on the **creation challenge** - the initial creation of a UI. Questions related to this challenge are about what elements are suitable for the UI, how UIs are structured and how the developer is involved into the creation process. A strong motivation for most works in this area is the adaptation to a newly encountered context of use. Thus, the aim is to support as many as possible contexts of use, often without necessarily knowing and specifying them in advance. This is frequently accomplished by automating the UI creation and adaptation, e.g., as in [CCD+04, NMH+02, MPS04, LVM+04, ZZHM07, MVLC08]. Hereby, modification of existing UIs is not explicitly investigated in these works.

On the other hand, the **modification challenge** focuses on updating an already existing UI. Questions about how the UI developer can apply modifications to UIs, in what way the update is executed, how she can evaluate her modifications and what the impact of a single modification is are prevailing. Damask [LL08] addresses this challenge and provides a pattern and layer concept. Sukaviriya et al. address the modification challenge in [SSRM07] by reflecting changes to business models in UI models.

Many approaches that rather focus on the creation challenge make use of model-to-model transformations (e.g., [MPS04, LVM+04, CCT+02]). These transformations could, in principle, be used to synchronize different artifacts after modifications have been made. Transformation languages like QVT relations [Obj05a], Solverational [PBM09] and triple graph grammars (TGG) [Sch95] can be employed. For example, Sottet et al. [SCF+06] use ATL[12], and Limbourg applies a generic graph-transformation language [Lim04]. Approaches that provide such generic solutions, as transformations do, can be applied to a great range of problem domains. But since they are not focused on the UI, their syntax, and more important their semantic, does not address UI specific issues. Consequently, they are very abstract for the UI engineer to use. But when using abstract descriptions, the connection to the concrete interfaces is often not clear to the engineer [MVLC08, MHP00]. The approach suffers from unpredictability. Furthermore, a new (often complex) language has to be learned, thus raising the threshold of use [MHP00] (requirement 4).

Mori et al. [MPS04] on the other hand encode their transformations into their tool – which implies that they are very specific, but the UI engineer cannot fully influence the transformations anymore. By using the presented mechanisms, our approach tries to connect the benefits of both sides: the UI engineer can fully influence the way modifications are applied and at the same time, the concepts to do so are easily comprehensible and UI specific.

UsiXML [LVM+04] by Quentin Limbourg is a well-known metamodel for UI modeling. The metamodel includes task, abstract and concrete UI modeling. Its Interaction Objects Classes are hardcoded into the metamodel. The focus of usiXML is on the creation challenge, hereby a generic graph-transformation language is used. In contrast, our metamodel allows arbitrary refinement levels so that the UI engineer can choose the nature of abstraction suitable to the problem at hand (requirement 1). Furthermore, our approach applies

[12] ATL – Atlas Transformation Language, cf. http://www.eclipse.org/m2m/atl/

the library metaphor and does not hardcode Interaction Object Classes into the metamodel, which allows for more easy extensibility (requirement 2).

Damask [LL08] is a tool mainly targeted towards UI prototyping. The modification of UIs that already are used in running applications is not Damask's focus. However, the layer concept presented in Damask is similar to the Refinement Tree of our approach. By introducing a DSL able to support arbitrary toolkits and allowing more than two layers of refinement, our work goes beyond Damask and allows a wide range of different contexts of use.

Gummy [MVLC08] is a UIML-based [HSL+08] tool for creating different UIs for different contexts of use. The tool allows creation of the UI in a WYSIWYG-fashion and maintains a UIML description, i.e., the abstract user interface, together with a platform mapping (e.g., to Java Swing), i.e. the concrete user interface. The abstract description can be used to generate new UI versions, but after creation the connection to the source UI is lost: modifications can only be applied to a single UI. In contrast, our approach keeps these connections and thus allows to apply one modification to multiple UIs. The authors of Gummy look into integration of a layer concept (like Damask or our work) into their approach [MVLC08].

Our previous work in [BPFM08] discusses the motivation and requirements behind refinement in more detail, but does not elaborate on the approach, and [BPM09] focusses on tool support. In contrast, the paper at hand presents the refinement approach including metamodel and constraints.

6 Conclusion and Outlook

We presented a DSL to describe user interfaces explicitly targeting the modification challenge. The DSL consists of the metamodel presented (abstract syntax), constraints that constitute its well-formedness rules. The semantic of its modeling concepts was described. The usage of the metamodel was described, its concept and implementation discussed, as well as its concrete syntax. The DSL adheres to the introduced requirements.

We currently research the integration of transformation approaches and the concepts presented in this paper. Because hand-crafting user interfaces for multiple target platforms is a costly task, transformations can be used for automatic generation of UIs. But the aesthetic quality of automatically generated UIs often stands behind that of manually crafted interfaces [MHP00, MVLC08, DMLC08]. Thus, easy manual modifications must be made possible.

References

[AK05] Colin Atkinson and Thomas Kühne. Concepts for Comparing Modeling Tool Architectures. In *MoDELS*, pages 398–413, 2005.

[BP09] Alexander Behring and Andreas Petter. Mapache Dialogue Refinement Tools: a Preliminary User Study. Technical Report TR-10, Telecooperation Research Division, TU Darmstadt, Darmstadt, May 2009. ISSN 1864-0516.

[BPFM08] Alexander Behring, Andreas Petter, Felix Flentge, and Max Mühlhäuser. Towards Multi-Level Dialogue Refinement for User Interfaces. In *Workshop on User Interface Description Languages*, 2008. April 5-10, 2008, Florence, Italy.

[BPM09] Alexander Behring, Andreas Petter, and Max Mühlhäuser. Rapidly Modifying Multiple User Interfaces of one Application. In *ICSOFT 2009*, pages 344–347. INSTICC Press, Jul 2009.

[CCD+04] Gaëlle Calvary, Joëlle Coutaz, Olfa Dâassi, Lionel Balme, and Alexandre Demeure. Towards a New Generation of Widgets for Supporting Software Plasticity: The "Comet". In Rémi Bastide, Philippe A. Palanque, and Jörg Roth, editors, *EHCI/DS-VIS*, volume 3425 of *Lecture Notes in Computer Science*, pages 306–324. Springer, 2004.

[CCT+02] Gaelle Calvary, Joëlle Coutaz, David Thevenin, Quentin Limbourg, Nathalie Souchon, Laurent Bouillon, Murielle Florins, and Jean Vanderdonckt. Plasticity of User Interfaces: A Revised Reference Framework. In *TAMODIA '02: Proceedings of the First International Workshop on Task Models and Diagrams for User Interface Design*, pages 127–134. INFOREC Publishing House Bucharest, 2002.

[DMLC08] Alexandre Demeure, Jan Meskens, Kris Luyten, and Karin Coninx. Design by Example of Plastic User Interfaces. In *Proceedings of CADUI2008, the 7th International Conference on Computer-Aided Design of User Interfaces*, 2008.

[DRO07] Jr.Ev Dan R. Olsen. Evaluating user interface systems research. In *UIST '07: Proceedings of the 20th annual ACM symposium on User interface software and technology*, pages 251–258, New York, NY, USA, 2007. ACM.

[HSL+08] James Helms, Robbie Schaefer, Kris Luyten, Jean Vanderdonckt, Jo Vermeulen, and Marc Abrams (Editors). User Interface Markup Language (UIML) Version 4.0 Committee Draft. Available at http://www.oasis-open.org/committees/download.php/28457/uiml-4.0-cd01.pdf, 2008. Last Access 19.02.2009.

[HT06] B. Hailpern and P. Tarr. Model-driven development: the good, the bad, and the ugly. *IBM Syst. J.*, 45(3):451–461, 2006.

[KUW02] Thomas Koch, Axel Uhl, and Dirk Weise. Model Driven Architecture, 2002.

[Lim04] Quentin Limbourg. *Multi-Path Development of User Interfaces*. PhD thesis, Universit catholique de Louvain, 2004.

[LL08] James Lin and James A. Landay. Employing patterns and layers for early-stage design and prototyping of cross-device user interfaces. In *CHI*, pages 1313–1322, 2008.

[LVM+04] Quentin Limbourg, Jean Vanderdonckt, Benjamin Michotte, Laurent Bouillon, and Víctor López-Jaquero. USIXML: A Language Supporting Multi-path Development of User Interfaces. In *EHCI/DS-VIS*, pages 200–220, 2004.

[MHP00] Brad Myers, Scott E. Hudson, and Randy Pausch. Past, Present, and Future of User Interface Software Tools. *ACM Transactions on Computer-Human Interaction (TOCHI)*, 7(1):3–28, 2000.

[MPS04] Giulio Mori, Fabio Paternò, and Carmen Santoro. Design and Development of Multidevice User Interfaces through Multiple Logical Descriptions. *IEEE Trans. Softw. Eng.*, 30(8):507–520, 2004.

[MVLC08] Jan Meskens, Jo Vermeulen, Kris Luyten, and Karin Coninx. Gummy for multiplatform user interface designs: shape me, multiply me, fix me, use me. In *AVI '08: Proceedings of the working conference on Advanced visual interfaces*, pages 233–240, New York, NY, USA, 2008. ACM.

[NMH+02] Jeffrey Nichols, Brad A. Myers, Michael Higgins, Joseph Hughes, Thomas K. Harris, Roni Rosenfeld, and Mathilde Pignol. Generating remote control interfaces for complex appliances. In *UIST '02: Proceedings of the 15th annual ACM symposium on User interface software and technology*, pages 161–170, New York, NY, USA, 2002. ACM.

[Obj05a] Object Management Group. MOF QVT Final Adopted Specification 2.0. Available at: `http://www.omg.org/docs/ptc/05-11-01.pdf`, 2005.

[Obj05b] Object Management Group. Unified Modeling Language: Superstructure 2.0. Available at: `http://www.omg.org/docs/formal/05-07-04.pdf`, 2005.

[PBM09] Andreas Petter, Alexander Behring, and Max Mühlhäuser. Constraint Solving in Model Transformations. In Richard F. Paige, editor, *International Conference on Model Transformation, ICMT 2009*. Springer, 2009. to appear.

[Pin00] Paulo Pinheiro da Silva. User Interface Declarative Models and Development Environments: A Survey. *Lecture Notes in Computer Science*, 1946:207–226, 2000.

[SCF+06] Jean-Sbastien Sottet, Gaelle Calvary, Jean-Marie Favre, Joelle Coutaz, and Alexandre Demeure. Towards Mapping and Model Transformation for Consistency of Plastic User Interfaces. In Kai Richter, Jeffrey Nichols, Krzysztof Gajos, and Ahmed Seffah, editors, *Workshop on The Many Faces of Consistency in Cross-platform Design, ACM conf. on Computer Human Interaction, CHI 2006*. ACM Press, 2006.

[Sch95] Andy Schürr. Specification of Graph Translators with Triple Graph Grammars. In *WG '94: Proceedings of the 20th International Workshop on Graph-Theoretic Concepts in Computer Science*, pages 151–163, London, UK, 1995. Springer-Verlag.

[Sel03] B. Selic. The pragmatics of model-driven development. *Software, IEEE*, 20(5):19–25, 2003.

[SSRM07] Noi Sukaviriya, Vibha Sinha, Thejaswini Ramachandra, and Senthil Mani. Model-Driven Approach for Managing Human Interface Design Life Cycle. In *MoDELS*, pages 226–240, 2007.

[ZZHM07] Xulin Zhao, Ying Zou, Jen Hawkins, and Bhadri Madapusi. A Business-Process-Driven Approach for Generating E-Commerce User Interfaces. In *MoDELS*, pages 256–270, 2007.

GI-Edition Lecture Notes in Informatics

P-1 Gregor Engels, Andreas Oberweis, Albert Zündorf (Hrsg.): Modellierung 2001.

P-2 Mikhail Godlevsky, Heinrich C. Mayr (Hrsg.): Information Systems Technology and its Applications, ISTA'2001.

P-3 Ana M. Moreno, Reind P. van de Riet (Hrsg.): Applications of Natural Language to Information Systems, NLDB'2001.

P-4 H. Wörn, J. Mühling, C. Vahl, H.-P. Meinzer (Hrsg.): Rechner- und sensorgestützte Chirurgie; Workshop des SFB 414.

P-5 Andy Schürr (Hg.): OMER – Object-Oriented Modeling of Embedded Real-Time Systems.

P-6 Hans-Jürgen Appelrath, Rolf Beyer, Uwe Marquardt, Heinrich C. Mayr, Claudia Steinberger (Hrsg.): Unternehmen Hochschule, UH'2001.

P-7 Andy Evans, Robert France, Ana Moreira, Bernhard Rumpe (Hrsg.): Practical UML-Based Rigorous Development Methods – Countering or Integrating the extremists, pUML'2001.

P-8 Reinhard Keil-Slawik, Johannes Magenheim (Hrsg.): Informatikunterricht und Medienbildung, INFOS'2001.

P-9 Jan von Knop, Wilhelm Haverkamp (Hrsg.): Innovative Anwendungen in Kommunikationsnetzen, 15. DFN Arbeitstagung.

P-10 Mirjam Minor, Steffen Staab (Hrsg.): 1st German Workshop on Experience Management: Sharing Experiences about the Sharing Experience.

P-11 Michael Weber, Frank Kargl (Hrsg.): Mobile Ad-Hoc Netzwerke, WMAN 2002.

P-12 Martin Glinz, Günther Müller-Luschnat (Hrsg.): Modellierung 2002.

P-13 Jan von Knop, Peter Schirmbacher and Viljan Mahni_ (Hrsg.): The Changing Universities – The Role of Technology.

P-14 Robert Tolksdorf, Rainer Eckstein (Hrsg.): XML-Technologien für das Semantic Web – XSW 2002.

P-15 Hans-Bernd Bludau, Andreas Koop (Hrsg.): Mobile Computing in Medicine.

P-16 J. Felix Hampe, Gerhard Schwabe (Hrsg.): Mobile and Collaborative Busi-ness 2002.

P-17 Jan von Knop, Wilhelm Haverkamp (Hrsg.): Zukunft der Netze –Die Verletzbarkeit meistern, 16. DFN Arbeitstagung.

P-18 Elmar J. Sinz, Markus Plaha (Hrsg.): Modellierung betrieblicher Informationssysteme – MobIS 2002.

P-19 Sigrid Schubert, Bernd Reusch, Norbert Jesse (Hrsg.): Informatik bewegt – Informatik 2002 – 32. Jahrestagung der Gesellschaft für Informatik e.V. (GI) 30.Sept.-3.Okt. 2002 in Dortmund.

P-20 Sigrid Schubert, Bernd Reusch, Norbert Jesse (Hrsg.): Informatik bewegt – Informatik 2002 – 32. Jahrestagung der Gesellschaft für Informatik e.V. (GI) 30.Sept.-3.Okt. 2002 in Dortmund (Ergänzungsband).

P-21 Jörg Desel, Mathias Weske (Hrsg.): Promise 2002: Prozessorientierte Methoden und Werkzeuge für die Entwicklung von Informationssystemen.

P-22 Sigrid Schubert, Johannes Magenheim, Peter Hubwieser, Torsten Brinda (Hrsg.): Forschungsbeiträge zur "Didaktik der Informatik" – Theorie, Praxis, Evaluation.

P-23 Thorsten Spitta, Jens Borchers, Harry M. Sneed (Hrsg.): Software Management 2002 – Fortschritt durch Beständigkeit

P-24 Rainer Eckstein, Robert Tolksdorf (Hrsg.): XMIDX 2003 – XML-Technologien für Middleware – Middleware für XML-Anwendungen

P-25 Key Pousttchi, Klaus Turowski (Hrsg.): Mobile Commerce – Anwendungen und Perspektiven – 3. Workshop Mobile Commerce, Universität Augsburg, 04.02.2003

P-26 Gerhard Weikum, Harald Schöning, Erhard Rahm (Hrsg.): BTW 2003: Datenbanksysteme für Business, Technologie und Web

P-27 Michael Kroll, Hans-Gerd Lipinski, Kay Melzer (Hrsg.): Mobiles Computing in der Medizin

P-28 Ulrich Reimer, Andreas Abecker, Steffen Staab, Gerd Stumme (Hrsg.): WM 2003: Professionelles Wissensmanagement – Erfahrungen und Visionen

P-29 Antje Düsterhöft, Bernhard Thalheim (Eds.): NLDB'2003: Natural Language Processing and Information Systems

P-30 Mikhail Godlevsky, Stephen Liddle, Heinrich C. Mayr (Eds.): Information Systems Technology and its Applications

P-31 Arslan Brömme, Christoph Busch (Eds.): BIOSIG 2003: Biometrics and Electronic Signatures

P-32 Peter Hubwieser (Hrsg.): Informatische Fachkonzepte im Unterricht – INFOS 2003

P-33 Andreas Geyer-Schulz, Alfred Taudes (Hrsg.): Informationswirtschaft: Ein Sektor mit Zukunft

P-34 Klaus Dittrich, Wolfgang König, Andreas Oberweis, Kai Rannenberg, Wolfgang Wahlster (Hrsg.): Informatik 2003 – Innovative Informatikanwendungen (Band 1)

P-35 Klaus Dittrich, Wolfgang König, Andreas Oberweis, Kai Rannenberg, Wolfgang Wahlster (Hrsg.): Informatik 2003 – Innovative Informatikanwendungen (Band 2)

P-36 Rüdiger Grimm, Hubert B. Keller, Kai Rannenberg (Hrsg.): Informatik 2003 – Mit Sicherheit Informatik

P-37 Arndt Bode, Jörg Desel, Sabine Rathmayer, Martin Wessner (Hrsg.): DeLFI 2003: e-Learning Fachtagung Informatik

P-38 E.J. Sinz, M. Plaha, P. Neckel (Hrsg.): Modellierung betrieblicher Informationssysteme – MobIS 2003

P-39 Jens Nedon, Sandra Frings, Oliver Göbel (Hrsg.): IT-Incident Management & IT-Forensics – IMF 2003

P-40 Michael Rebstock (Hrsg.): Modellierung betrieblicher Informationssysteme – MobIS 2004

P-41 Uwe Brinkschulte, Jürgen Becker, Dietmar Fey, Karl-Erwin Großpietsch, Christian Hochberger, Erik Maehle, Thomas Runkler (Edts.): ARCS 2004 – Organic and Pervasive Computing

P-42 Key Pousttchi, Klaus Turowski (Hrsg.): Mobile Economy – Transaktionen und Prozesse, Anwendungen und Dienste

P-43 Birgitta König-Ries, Michael Klein, Philipp Obreiter (Hrsg.): Persistance, Scalability, Transactions – Database Mechanisms for Mobile Applications

P-44 Jan von Knop, Wilhelm Haverkamp, Eike Jessen (Hrsg.): Security, E-Learning. E-Services

P-45 Bernhard Rumpe, Wofgang Hesse (Hrsg.): Modellierung 2004

P-46 Ulrich Flegel, Michael Meier (Hrsg.): Detection of Intrusions of Malware & Vulnerability Assessment

P-47 Alexander Prosser, Robert Krimmer (Hrsg.): Electronic Voting in Europe – Technology, Law, Politics and Society

P-48 Anatoly Doroshenko, Terry Halpin, Stephen W. Liddle, Heinrich C. Mayr (Hrsg.): Information Systems Technology and its Applications

P-49 G. Schiefer, P. Wagner, M. Morgenstern, U. Rickert (Hrsg.): Integration und Datensicherheit – Anforderungen, Konflikte und Perspektiven

P-50 Peter Dadam, Manfred Reichert (Hrsg.): INFORMATIK 2004 – Informatik verbindet (Band 1) Beiträge der 34. Jahrestagung der Gesellschaft für Informatik e.V. (GI), 20.-24. September 2004 in Ulm

P-51 Peter Dadam, Manfred Reichert (Hrsg.): INFORMATIK 2004 – Informatik verbindet (Band 2) Beiträge der 34. Jahrestagung der Gesellschaft für Informatik e.V. (GI), 20.-24. September 2004 in Ulm

P-52 Gregor Engels, Silke Seehusen (Hrsg.): DELFI 2004 – Tagungsband der 2. e-Learning Fachtagung Informatik

P-53 Robert Giegerich, Jens Stoye (Hrsg.): German Conference on Bioinformatics – GCB 2004

P-54 Jens Borchers, Ralf Kneuper (Hrsg.): Softwaremanagement 2004 – Outsourcing und Integration

P-55 Jan von Knop, Wilhelm Haverkamp, Eike Jessen (Hrsg.): E-Science und Grid Adhoc-Netze Medienintegration

P-56 Fernand Feltz, Andreas Oberweis, Benoit Otjacques (Hrsg.): EMISA 2004 – Informationssysteme im E-Business und E-Government

P-57 Klaus Turowski (Hrsg.): Architekturen, Komponenten, Anwendungen

P-58 Sami Beydeda, Volker Gruhn, Johannes Mayer, Ralf Reussner, Franz Schweiggert (Hrsg.): Testing of Component-Based Systems and Software Quality

P-59 J. Felix Hampe, Franz Lehner, Key Pousttchi, Kai Ranneberg, Klaus Turowski (Hrsg.): Mobile Business – Processes, Platforms, Payments

P-60 Steffen Friedrich (Hrsg.): Unterrichtskonzepte für inforrmatische Bildung

P-61 Paul Müller, Reinhard Gotzhein, Jens B. Schmitt (Hrsg.): Kommunikation in verteilten Systemen

P-62 Federrath, Hannes (Hrsg.): „Sicherheit 2005“ – Sicherheit – Schutz und Zuverlässigkeit

P-63 Roland Kaschek, Heinrich C. Mayr, Stephen Liddle (Hrsg.): Information Systems – Technology and ist Applications

P-64 Peter Liggesmeyer, Klaus Pohl, Michael Goedicke (Hrsg.): Software Engineering 2005

P-65 Gottfried Vossen, Frank Leymann, Peter Lockemann, Wolffried Stucky (Hrsg.): Datenbanksysteme in Business, Technologie und Web

P-66 Jörg M. Haake, Ulrike Lucke, Djamshid Tavangarian (Hrsg.): DeLFI 2005: 3. deutsche e-Learning Fachtagung Informatik

P-67 Armin B. Cremers, Rainer Manthey, Peter Martini, Volker Steinhage (Hrsg.): INFORMATIK 2005 – Informatik LIVE (Band 1)

P-68 Armin B. Cremers, Rainer Manthey, Peter Martini, Volker Steinhage (Hrsg.): INFORMATIK 2005 – Informatik LIVE (Band 2)

P-69 Robert Hirschfeld, Ryszard Kowalcyk, Andreas Polze, Matthias Weske (Hrsg.): NODe 2005, GSEM 2005

P-70 Klaus Turowski, Johannes-Maria Zaha (Hrsg.): Component-oriented Enterprise Application (COAE 2005)

P-71 Andrew Torda, Stefan Kurz, Matthias Rarey (Hrsg.): German Conference on Bioinformatics 2005

P-72 Klaus P. Jantke, Klaus-Peter Fähnrich, Wolfgang S. Wittig (Hrsg.): Marktplatz Internet: Von e-Learning bis e-Payment

P-73 Jan von Knop, Wilhelm Haverkamp, Eike Jessen (Hrsg.): “Heute schon das Morgen sehen“

P-74 Christopher Wolf, Stefan Lucks, Po-Wah Yau (Hrsg.): WEWoRC 2005 – Western European Workshop on Research in Cryptology

P-75 Jörg Desel, Ulrich Frank (Hrsg.): Enterprise Modelling and Information Systems Architecture

P-76 Thomas Kirste, Birgitta König-Riess, Key Pousttchi, Klaus Turowski (Hrsg.): Mobile Informationssysteme – Potentiale, Hindernisse, Einsatz

P-77 Jana Dittmann (Hrsg.): SICHERHEIT 2006

P-78 K.-O. Wenkel, P. Wagner, M. Morgenstern, K. Luzi, P. Eisermann (Hrsg.): Land- und Ernährungswirtschaft im Wandel

P-79 Bettina Biel, Matthias Book, Volker Gruhn (Hrsg.): Softwareengineering 2006

P-80 Mareike Schoop, Christian Huemer, Michael Rebstock, Martin Bichler (Hrsg.): Service-Oriented Electronic Commerce

P-81 Wolfgang Karl, Jürgen Becker, Karl-Erwin Großpietsch, Christian Hochberger, Erik Maehle (Hrsg.): ARCS´06

P-82 Heinrich C. Mayr, Ruth Breu (Hrsg.): Modellierung 2006

P-83 Daniel Huson, Oliver Kohlbacher, Andrei Lupas, Kay Nieselt and Andreas Zell (eds.): German Conference on Bioinformatics

P-84 Dimitris Karagiannis, Heinrich C. Mayr, (Hrsg.): Information Systems Technology and its Applications

P-85 Witold Abramowicz, Heinrich C. Mayr, (Hrsg.): Business Information Systems

P-86 Robert Krimmer (Ed.): Electronic Voting 2006

P-87 Max Mühlhäuser, Guido Rößling, Ralf Steinmetz (Hrsg.): DELFI 2006: 4. e-Learning Fachtagung Informatik

P-88 Robert Hirschfeld, Andreas Polze, Ryszard Kowalczyk (Hrsg.): NODe 2006, GSEM 2006

P-90 Joachim Schelp, Robert Winter, Ulrich Frank, Bodo Rieger, Klaus Turowski (Hrsg.): Integration, Informationslogistik und Architektur

P-91 Henrik Stormer, Andreas Meier, Michael Schumacher (Eds.): European Conference on eHealth 2006

P-92 Fernand Feltz, Benoît Otjacques, Andreas Oberweis, Nicolas Poussing (Eds.): AIM 2006

P-93 Christian Hochberger, Rüdiger Liskowsky (Eds.): INFORMATIK 2006 – Informatik für Menschen, Band 1

P-94 Christian Hochberger, Rüdiger Liskowsky (Eds.): INFORMATIK 2006 – Informatik für Menschen, Band 2

P-95 Matthias Weske, Markus Nüttgens (Eds.): EMISA 2005: Methoden, Konzepte und Technologien für die Entwicklung von dienstbasierten Informationssystemen

P-96 Saartje Brockmans, Jürgen Jung, York Sure (Eds.): Meta-Modelling and Ontologies

P-97 Oliver Göbel, Dirk Schadt, Sandra Frings, Hardo Hase, Detlef Günther, Jens Nedon (Eds.): IT-Incident Mangament & IT-Forensics – IMF 2006

P-98 Hans Brandt-Pook, Werner Simonsmeier und Thorsten Spitta (Hrsg.): Beratung in der Softwareentwicklung – Modelle, Methoden, Best Practices

P-99 Andreas Schwill, Carsten Schulte, Marco Thomas (Hrsg.): Didaktik der Informatik

P-100 Peter Forbrig, Günter Siegel, Markus Schneider (Hrsg.): HDI 2006: Hochschuldidaktik der Informatik

P-101 Stefan Böttinger, Ludwig Theuvsen, Susanne Rank, Marlies Morgenstern (Hrsg.): Agrarinformatik im Spannungsfeld zwischen Regionalisierung und globalen Wertschöpfungsketten

P-102 Otto Spaniol (Eds.): Mobile Services and Personalized Environments

P-103 Alfons Kemper, Harald Schöning, Thomas Rose, Matthias Jarke, Thomas Seidl, Christoph Quix, Christoph Brochhaus (Hrsg.): Datenbanksysteme in Business, Technologie und Web (BTW 2007)

P-104 Birgitta König-Ries, Franz Lehner, Rainer Malaka, Can Türker (Hrsg.) MMS 2007: Mobilität und mobile Informationssysteme

P-105 Wolf-Gideon Bleek, Jörg Raasch, Heinz Züllighoven (Hrsg.) Software Engineering 2007

P-106 Wolf-Gideon Bleek, Henning Schwentner, Heinz Züllighoven (Hrsg.) Software Engineering 2007 – Beiträge zu den Workshops

P-107 Heinrich C. Mayr, Dimitris Karagiannis (eds.) Information Systems Technology and its Applications

P-108 Arslan Brömme, Christoph Busch, Detlef Hühnlein (eds.) BIOSIG 2007: Biometrics and Electronic Signatures

P-109 Rainer Koschke, Otthein Herzog, Karl-Heinz Rödiger, Marc Ronthaler (Hrsg.) INFORMATIK 2007 Informatik trifft Logistik Band 1

P-110 Rainer Koschke, Otthein Herzog, Karl-Heinz Rödiger, Marc Ronthaler (Hrsg.) INFORMATIK 2007 Informatik trifft Logistik Band 2

P-111 Christian Eibl, Johannes Magenheim, Sigrid Schubert, Martin Wessner (Hrsg.) DeLFI 2007: 5. e-Learning Fachtagung Informatik

P-112 Sigrid Schubert (Hrsg.) Didaktik der Informatik in Theorie und Praxis

P-113 Sören Auer, Christian Bizer, Claudia Müller, Anna V. Zhdanova (Eds.) The Social Semantic Web 2007 Proceedings of the 1st Conference on Social Semantic Web (CSSW)

P-114 Sandra Frings, Oliver Göbel, Detlef Günther, Hardo G. Hase, Jens Nedon, Dirk Schadt, Arslan Brömme (Eds.) IMF2007 IT-incident management & IT-forensics Proceedings of the 3rd International Conference on IT-Incident Management & IT-Forensics

P-115 Claudia Falter, Alexander Schliep, Joachim Selbig, Martin Vingron and Dirk Walther (Eds.) German conference on bioinformatics GCB 2007

P-116 Witold Abramowicz, Leszek Maciszek (Eds.) Business Process and Services Computing 1st International Working Conference on Business Process and Services Computing BPSC 2007

P-117 Ryszard Kowalczyk (Ed.) Grid service engineering and manegement The 4th International Conference on Grid Service Engineering and Management GSEM 2007

P-118 Andreas Hein, Wilfried Thoben, Hans-Jürgen Appelrath, Peter Jensch (Eds.) European Conference on ehealth 2007

P-119 Manfred Reichert, Stefan Strecker, Klaus Turowski (Eds.) Enterprise Modelling and Information Systems Architectures Concepts and Applications

P-120 Adam Pawlak, Kurt Sandkuhl, Wojciech Cholewa, Leandro Soares Indrusiak (Eds.) Coordination of Collaborative Engineering - State of the Art and Future Challenges

P-121 Korbinian Herrmann, Bernd Bruegge (Hrsg.) Software Engineering 2008 Fachtagung des GI-Fachbereichs Softwaretechnik

P-122 Walid Maalej, Bernd Bruegge (Hrsg.) Software Engineering 2008 - Workshopband Fachtagung des GI-Fachbereichs Softwaretechnik

P-123 Michael H. Breitner, Martin Breunig, Elgar Fleisch, Ley Pousttchi, Klaus Turowski (Hrsg.)
Mobile und Ubiquitäre Informationssysteme – Technologien, Prozesse, Marktfähigkeit
Proceedings zur 3. Konferenz Mobile und Ubiquitäre Informationssysteme (MMS 2008)

P-124 Wolfgang E. Nagel, Rolf Hoffmann, Andreas Koch (Eds.)
9th Workshop on Parallel Systems and Algorithms (PASA)
Workshop of the GI/ITG Speciel Interest Groups PARS and PARVA

P-125 Rolf A.E. Müller, Hans-H. Sundermeier, Ludwig Theuvsen, Stephanie Schütze, Marlies Morgenstern (Hrsg.)
Unternehmens-IT: Führungsinstrument oder Verwaltungsbürde
Referate der 28. GIL Jahrestagung

P-126 Rainer Gimnich, Uwe Kaiser, Jochen Quante, Andreas Winter (Hrsg.)
10th Workshop Software Reengineering (WSR 2008)

P-127 Thomas Kühne, Wolfgang Reisig, Friedrich Steimann (Hrsg.)
Modellierung 2008

P-128 Ammar Alkassar, Jörg Siekmann (Hrsg.)
Sicherheit 2008
Sicherheit, Schutz und Zuverlässigkeit
Beiträge der 4. Jahrestagung des Fachbereichs Sicherheit der Gesellschaft für Informatik e.V. (GI)
2.-4. April 2008
Saarbrücken, Germany

P-129 Wolfgang Hesse, Andreas Oberweis (Eds.)
Sigsand-Europe 2008
Proceedings of the Third AIS SIGSAND European Symposium on Analysis, Design, Use and Societal Impact of Information Systems

P-130 Paul Müller, Bernhard Neumair, Gabi Dreo Rodosek (Hrsg.)
1. DFN-Forum Kommunikations-technologien Beiträge der Fachtagung

P-131 Robert Krimmer, Rüdiger Grimm (Eds.)
3rd International Conference on Electronic Voting 2008
Co-organized by Council of Europe, Gesellschaft für Informatik and E-Voting.CC

P-132 Silke Seehusen, Ulrike Lucke, Stefan Fischer (Hrsg.)
DeLFI 2008:
Die 6. e-Learning Fachtagung Informatik

P-133 Heinz-Gerd Hegering, Axel Lehmann, Hans Jürgen Ohlbach, Christian Scheideler (Hrsg.)
INFORMATIK 2008
Beherrschbare Systeme – dank Informatik
Band 1

P-134 Heinz-Gerd Hegering, Axel Lehmann, Hans Jürgen Ohlbach, Christian Scheideler (Hrsg.)
INFORMATIK 2008
Beherrschbare Systeme – dank Informatik
Band 2

P-135 Torsten Brinda, Michael Fothe, Peter Hubwieser, Kirsten Schlüter (Hrsg.)
Didaktik der Informatik – Aktuelle Forschungsergebnisse

P-136 Andreas Beyer, Michael Schroeder (Eds.)
German Conference on Bioinformatics GCB 2008

P-137 Arslan Brömme, Christoph Busch, Detlef Hühnlein (Eds.)
BIOSIG 2008: Biometrics and Electronic Signatures

P-138 Barbara Dinter, Robert Winter, Peter Chamoni, Norbert Gronau, Klaus Turowski (Hrsg.)
Synergien durch Integration und Informationslogistik
Proceedings zur DW2008

P-139 Georg Herzwurm, Martin Mikusz (Hrsg.)
Industrialisierung des Software-Managements
Fachtagung des GI-Fachausschusses Management der Anwendungsentwick-lung und -wartung im Fachbereich Wirtschaftsinformatik

P-140 Oliver Göbel, Sandra Frings, Detlef Günther, Jens Nedon, Dirk Schadt (Eds.)
IMF 2008 - IT Incident Management & IT Forensics

P-141 Peter Loos, Markus Nüttgens, Klaus Turowski, Dirk Werth (Hrsg.)
Modellierung betrieblicher Informations-systeme (MobIS 2008)
Modellierung zwischen SOA und Compliance Management

P-142 R. Bill, P. Korduan, L. Theuvsen, M. Morgenstern (Hrsg.)
Anforderungen an die Agrarinformatik durch Globalisierung und Klimaveränderung

P-143 Peter Liggesmeyer, Gregor Engels, Jürgen Münch, Jörg Dörr, Norman Riegel (Hrsg.)
Software Engineering 2009
Fachtagung des GI-Fachbereichs Softwaretechnik

P-144 Johann-Christoph Freytag, Thomas Ruf, Wolfgang Lehner, Gottfried Vossen (Hrsg.)
Datenbanksysteme in Business, Technologie und Web (BTW)

P-145 Knut Hinkelmann, Holger Wache (Eds.)
WM2009: 5th Conference on Professional Knowledge Management

P-146 Markus Bick, Martin Breunig, Hagen Höpfner (Hrsg.)
Mobile und Ubiquitäre Informationssysteme – Entwicklung, Implementierung und Anwendung
4. Konferenz Mobile und Ubiquitäre Informationssysteme (MMS 2009)

P-147 Witold Abramowicz, Leszek Maciaszek, Ryszard Kowalczyk, Andreas Speck (Eds.)
Business Process, Services Computing and Intelligent Service Management
BPSC 2009 · ISM 2009 · YRW-MBP 2009

P-148 Christian Erfurth, Gerald Eichler, Volkmar Schau (Eds.)
9th International Conference on Innovative Internet Community Systems
I²CS 2009

P-149 Paul Müller, Bernhard Neumair, Gabi Dreo Rodosek (Hrsg.)
2. DFN-Forum Kommunikationstechnologien
Beiträge der Fachtagung

P-150 Jürgen Münch, Peter Liggesmeyer (Hrsg.)
Software Engineering
2009 - Workshopband

P-151 Armin Heinzl, Peter Dadam, Stefan Kirn, Peter Lockemann (Eds.)
PRIMIUM
Process Innovation for Enterprise Software

P-152 Jan Mendling, Stefanie Rinderle-Ma, Werner Esswein (Eds.)
Enterprise Modelling and Information Systems Architectures
Proceedings of the 3rd Int'l Workshop EMISA 2009

P-153 Andreas Schwill, Nicolas Apostolopoulos (Hrsg.)
Lernen im Digitalen Zeitalter
DeLFI 2009 – Die 7. E-Learning Fachtagung Informatik

P-154 Stefan Fischer, Erik Maehle Rüdiger Reischuk (Hrsg.)
INFORMATIK 2009
Im Focus das Leben

P-155 Arslan Brömme, Christoph Busch, Detlef Hühnlein (Eds.)
BIOSIG 2009:
Biometrics and Electronic Signatures
Proceedings of the Special Interest Group on Biometrics and Electronic Signatures

P-156 Bernhard Koerber (Hrsg.)
Zukunft braucht Herkunft
25 Jahre »INFOS – Informatik und Schule«

P-157 Ivo Grosse, Steffen Neumann, Stefan Posch, Falk Schreiber, Peter Stadler (Eds.)
German Conference on Bioinformatics 2009

P-158 W. Claupein, L. Theuvsen, A. Kämpf, M. Morgenstern (Hrsg.)
Precision Agriculture Reloaded – Informationsgestützte Landwirtschaft

P-159 Gregor Engels, Markus Luckey, Wilhelm Schäfer (Hrsg.)
Software Engineering 2010

P-161 Gregor Engels, Dimitris Karagiannis Heinrich C. Mayr (Hrsg.)
Modellierung 2010

P-162 Maria A. Wimmer, Uwe Brinkhoff, Siegfried Kaiser, Dagmar Lück-Schneider, Erich Schweighofer, Andreas Wiebe (Hrsg.)
Vernetzte IT für einen effektiven Staat
Gemeinsame Fachtagung Verwaltungsinformatik (FTVI) und Fachtagung Rechtsinformatik (FTRI) 2010

P-163 Markus Bick, Stefan Eulgem, Elgar Fleisch, J. Felix Hampe, Birgitta König-Ries, Franz Lehner, Key Pousttchi, Kai Rannenberg (Hrsg.)
Mobile und Ubiquitäre Informationssysteme
Technologien, Anwendungen und Dienste zur Unterstützung von mobiler Kollaboration

The titles can be purchased at:

Köllen Druck + Verlag GmbH
Ernst-Robert-Curtius-Str. 14 · D-53117 Bonn
Fax: +49 (0)228/9898222
E-Mail: druckverlag@koellen.de