# Methodenentwicklung

**Dissertation**

zur Erlangung des akademischen Grades

Dr. rer. pol.

vorgelegt an der

Fakultät für Wirtschaftswissenschaften
der
Technischen Universität Dresden

von

Dipl. Wirtsch.-Inf. Steffen Greiffenberg

vorgelegt:
17.01.2003
verteidigt:

Gutachter:
Prof. Dr. Werner Esswein
Prof. Dr. Georg Herzwurz
Prof. Dr. Susanne Strahringer

# Vorwort

Der Lehrstuhl für Wirtschaftsinformatik, insbesondere Systementwicklung, an der Fakultät Wirtschaftswissenschaften der Technischen Universität Dresden ist seit 1998 nicht nur das Zentrum meiner beruflichen Tätigkeit sondern auch der Nährboden für meine persönliche wissenschaftliche Entwicklung. Seinem Inhaber und meinem Doktorvater Prof. Dr. Werner Esswein gilt ein ganz besonderer Dank für die Schaffung der Möglichkeiten, deren Grenzen nie Zwang und deren Inhalt stets Gelegenheit und Hilfe waren. Die hier dargelegten Ideen werden aus meiner tiefen Überzeugung zu einer erfolgreichen gemeinsamen Unternehmung führen.

Mein Freund Andreas Dietzsch war von der Betreuung meiner Diplomarbeit bis zur Durchsicht meiner Dissertation stets ein Partner, dessen Überzeugung in der Diskussion einen hohen Stellenwert für mich hatte. In seiner Rolle als Methoden-Verantwortlicher war er nicht nur ein interessanter „Forschungsgegenstand“ dieser Arbeit, sondern vielmehr die zur Bewertung aller theoretischen Ausführungen notwendige Stimme der Praxis, die mir zudem in diesem Fall auch im Promotionsverfahren ein Jahr voraus war.

Für die Impulse, die ich aus der Betreuung der Diplomarbeiten von Herrn Martin Rose, Herrn Robert Braun und Herrn Andreas Gehlert erhalten habe, bin ich allen Autoren dankbar. Die anregenden Diskussionen mit Herrn Christian Kluge haben diese Arbeit in vielen Teilen zu dem gemacht, was sie heute ist. Frau Lisa Gerstenberger half mir mit unerschütterlicher Ruhe, die Hürden einer neuen Syntax und manchmal auch Semantik des Deutschen zu nehmen. Nur mit Hilfe von Herrn Andreas Gehlert konnte ich dieses Wissen widerspruchsfrei auch an meine Informationstechnik weiterreichen.

Von unschätzbarem Wert für die Fertigstellung dieser Arbeit waren die häufigen Fragestellungen im Freundes- und Bekanntenkreis nach dem inhaltlichen und zeitlichen Horizont meiner Dissertationsschrift. Aufgrund ständig neuer Impulse aus dem wissenschaftlichen Umfeld hätte der Abschluss der Arbeit fast nach Belieben in die Zukunft verlegt werden können. Spätestens nachdem man aufgrund neuer Auflagen die eine oder andere Quelle aktualisiert hat, erkennt man jedoch die Dringlichkeit des eigenen Handelns.

Den größten Wert in meinem Leben bildet die Familie. Meinen Eltern, die diesen Bildungsweg ermöglichten, kann ich nicht genug danken. Meiner Ehefrau gilt ein besonderer Dank für einfach alles und besonders dafür, dass ich etwas von mir an meine Tochter weitergeben darf.

# Inhaltsverzeichnis

# Abkürzungsverzeichnis

| | |
|---|---|
| ACME | Assembling Configuration Management Environments |
| ARIS | Architektur integrierter Informationssysteme |
| bez. | bezüglich |
| BK | Bezugskonfiguration |
| BOC | Business Objectives Consulting |
| BPR | Business Process Redesign |
| bspw. | beispielsweise |
| BWW | Bunge-Wand-Weber Modell |
| bzw. | beziehungsweise |
| CASE | Computer Aided Software Engineering |
| CCB | Change Control Board |
| CR | Change Request |
| CRD | Change Request Dokument |
| CVS | Concurrent Versioning System |
| DBMS | Datenbank Management System |
| DC | Development Culture |
| DFD | Datenflussdiagramm |
| d. h. | das heißt |
| DIN | Deutsches Institut für Normung e. V. |
| EER-Diagramm | Erweiterten Entity-Relationship-Diagramm |
| EN | Europäische Norm |
| engl. | englisch |
| EPK | Ereignisgesteuerte Prozesskette |
| ER | Ergänzende Regel |
| ERM | Entity-Relationship Model |
| etc. | et cetera |
| evtl. | eventuell |
| f | folgende |
| ff | fortfolgende |
| GC | Group Culture |
| ggf. | gegebenenfalls |
| GmbH | Gesellschaft mit beschränkter Haftung |
| GME | Generischer Modelleditor |
| GoM | Grundsätze ordnungsmäßiger Modellierung |

| | |
|---|---|
| GRAL | Graph Specification Language |
| H | Heuristik |
| HC | Hierarchical Culture |
| HoQ | House of Quality |
| i. A. a. | in Anlehnung an |
| i. d. R. | in der Regel |
| i. e. S. | im engeren Sinne |
| IKS | Informations- und Kommunikationssystem |
| IM | Informationsmodelle |
| inkl. | inklusive |
| insb. | insbesondere |
| IS | Informationssystem |
| ISO | International Organization for Standardization |
| i. w. S. | im weiteren Sinne |
| Jh. | Jahrhundert |
| KA | Konfigurationsaudit |
| KB | Konfigurationsbuchführung |
| KE | Konfigurationselement |
| KI | Konfigurationsidentifikation |
| KM | Konfigurationsmanagement |
| KMP | Konfigurationsmanagementplan |
| KMS | Konfigurationsmanagementsystem |
| KÜ | Konfigurationsüberwachung |
| ME | Method Engineering |
| MEMO | Multi Perspective Enterprise Modelling |
| MOF | Meta Object Facility |
| NATO | North Atlantic Treaty Organisation |
| OMD | Object Model Designer |
| OMG | Object Management Group |
| OML | Object Modelling Language |
| OPEN | Object-oriented Process, Environment and Notation |
| OPRR | Object, Property, Relationship and Role |
| OrgML | Organisation Modelling Language |
| PM | Projektmanagement |
| PML | Process Modelling Language |
| QFD | Quality Function Deployment |
| QM | Qualitätsmanagement |

| | |
|---|---|
| QS | Qualitätssicherung |
| R | Regel |
| RC | Rational Culture |
| RIM | Referenz-Informationsmodelle |
| RUP | Rational Unified Process |
| S. | Seite |
| SE | Systementwicklung |
| SEI | Software Engineering Institute |
| SERM | Strukturiertes Entity-Relationship Modell |
| SKM | Software-Konfigurationsmanagement |
| SOM | Semantisches Objektmodell |
| u. a. | unter anderem |
| u. U. | unter Umständen |
| UML | Unified Modeling Language |
| vgl. | vergleiche |
| XML | Extensible Markup Language |
| XP | Extreme Programming |
| z. B. | zum Beispiel |
| zw. | zwischen |

# Abbildungsverzeichnis

# Tabellenverzeichnis

# 1 Einleitung

## 1.1 Zielsetzung und Motivation

Das Konzept der Methode wird bereits seit einigen Dekaden in der Wirtschaftsinformatik und Informatik diskutiert (vgl. [CrÅg01], S. 1). Es wird seit der Vorstellung seiner ersten Vertreter in der Systementwicklung dazu verwendet, die Gestalter von Informationssystemen bei ihrer Arbeit zu unterstützen und zu führen. In dieser Arbeit bildet das zentrale Hilfsmittel der Modellierung und deren methodische Unterstützung den für die Wirtschaftsinformatik typischen Betrachtungsgegenstand. Modelle werden dabei als Ergebnis einer Konstruktionsleistung aufgefasst, die es mit geeigneten Techniken der Repräsentation und einer Beschreibung des Entwicklungsprozesses zu ermöglichen gilt. Die Selbstverständlichkeit, mit der als „state of the art" ein ingenieurmäßiges Herangehen bei der Modellbildung gefordert wird, wird in dieser Arbeit auf eine übergeordnete Ebene übertragen. Somit wird hier ein methodisches Hilfsmittel vorgestellt, mit dem die Spezifikation von Konzepten, Repräsentationstechniken und Prozessen bei der Modellierung möglich wird. Diese Arbeit stellt eine neue Methode der Analyse, Entwicklung und Anwendung von Methoden vor.

Der dabei zugrunde gelegte Begriff des *Informationsmodells* führt zu einer breiten Anwendbarkeit der hier vorgestellten Methode im Unternehmen. Für die Erklärung und Gestaltung von dessen Organisationen und Anwendungssystemen wird sowohl in der Theorie als auch in der Praxis ein effizientes Hilfsmittel gefordert, das schnell, sicher und kostengünstig eine Repräsentation der Strukturen und Abläufe im Unternehmen zulässt. Mit der Arbeit wird ein Werkzeug definiert, das die Qualität dieser Spezifikationen erhöht, indem sowohl direkt zuordenbare Eigenschaften messbar werden, als auch ein Vorgehen so festgelegt werden kann, dass Forderungen nicht nur erhoben, sondern auch eingehalten werden können.

Von *praktischer Relevanz* ist die in dieser Arbeit definierte Methode zur Entwicklung von Methoden für eine Vielzahl von Projekten, in denen die *Informationsmodellierung* angewendet wird. Das damit umspannte Einsatzgebiet ist nicht erst seit den populärwissenschaftlichen Arbeiten von HAMMER und CHAMPY (vgl. [HaCh93]) auf das Gebiet der Systementwicklung beschränkt. Die folgenden Ausführungen werden eine in der Praxis einsetzbare Methode definieren, die es Organisationen erlaubt, ihr Wissen um Verfahren zur Informationsmodellierung zu speichern und ggf. wiederholt anzuwenden. Die Methode wird die Konstruktion von Begriffssystemen und Modellen unterstützen sowie das Vorgehen beschreiben und dokumentieren.

Die *wissenschaftlich-theoretische* Relevanz der Arbeit ergibt sich aus der Notwendigkeit der Schaffung verbesserten Wissens zur Beherrschung von Informations- und Kommunikationssystemen (IKS). HEINZL ET AL. erheben diese Forderung in ihrer Studie zum obersten Erkennt-

nisziel der Wirtschaftsinformatik[1] in den nächsten zehn Jahren (vgl. [Hei+01], S. 229f).[2] Auf Rang 5 wurde in dieser Studie die Schaffung verbesserten Wissens über die Architektur von IKS eingeordnet. Diese Zielstellung motiviert die Arbeit zusätzlich, da unter diesem Punkt auch das Theorie- und Methodenwissen der Wirtschaftsinformatik zusammengefasst wurde.

Der mit der Arbeit entwickelte methodische Ansatz zur Auswahl, Anpassung und Entwicklung von Methoden der Informationsmodellierung wird an einer Reihe von Anforderungen evaluiert, die im Vorfeld zur Bewertung bereits vorhandener Ansätze herangezogen wurde. Mit diesem Hilfsmittel wird am Ende der Arbeit auch über die Erreichung der hier gestellten Ziele geurteilt.[3] Für die praktische Relevanz der Arbeit wird die Methode derart ausgestaltet, dass sie zum einen den im Folgenden vorgestellten Anforderungen entspricht und zum anderen selbst wieder anpassbar an neue Anforderungen gestaltet wird. Die Abbildung 1 fasst die mit der Arbeit verbundenen Zielsetzungen zusammen.

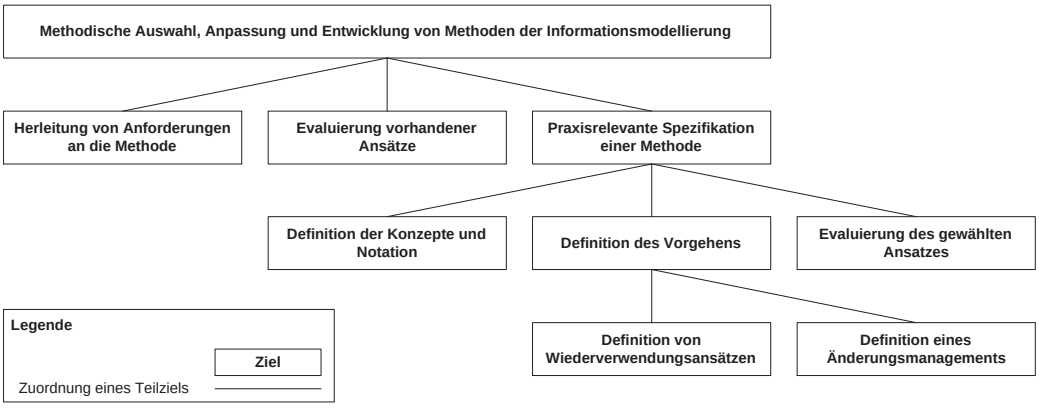



Abbildung 1: Zielstellung der Arbeit

[1] Deren Erhebung und Methode der Rangfolgeermittlung blieb einerseits in Folgebeiträgen nicht unumstritten, insbesondere deren Ermittlung aus wechselnden Anwendungsproblematiken widerspricht universalen Erkenntnisinteressen einer Wissenschaft (vgl. [Ever02], S. 91ff). Andererseits lässt dieser Beitrag jedoch eine breite Diskussion über diese Ziele innerhalb der Wirtschaftsinformatik überhaupt erst möglich werden.

[2] Von HEINZL ET AL. wird hier angemerkt, dass auf der einen Seite von der Wirtschaftsinformatik konkret anwendbare Lösungen gefordert werden, die sich jedoch zumindest in Bezug auf die Erstellung von Informationssystemen kaum reproduzieren lassen. Dem kann nur mit einem hohen Maß an Methoden- und Verfahrenskompetenz entgegengewirkt werden (vgl. [Hei+01], S. 230).

[3] Der Subjektivität dieser Bewertung wird mit einer breiten Diskussion vorhandener Literatur auf diesem Gebiet entgegengewirkt. Sie kann jedoch nicht vollständig verneint werden - die Offenlegung der Kriterien und deren Gewichtung gestattet jedoch die schnelle Ableitung eigener Urteile.

## 1.2 Einordnung der Arbeit

Als wesentliches Hilfsmittel bei der Gestaltung von Geschäftsprozessen oder Entwicklung von IKS gilt in der Wirtschaftsinformatik unbestritten die Modellbildung.[4] Diese Arbeit wird in späteren Abschnitten existierende Verfahren der Modellbildung vorstellen und deren wesentliche Mängel aufzeigen. Ohne bereits an dieser Stelle eine Klassifikation der Ansätze vorzunehmen, sei darauf hingewiesen, dass seit der Entwicklung von Methoden zur Abbildung von Einzelaspekten in Unternehmen (durch die vorgangs-, funktions- und datenorientierten Ansätze aus den 1960er bzw. 1970er Jahren) vor allem zwei wesentliche Tendenzen in der Unternehmensmodellierung erkennbar sind. Zum einen ermöglichen Methoden zur Erstellung integrierter Modelle mehrere zusammenhängende Sichten auf das Unternehmen und zum anderen gewinnt eine Beschreibung der Verfahren zur Modellierung immer mehr an Bedeutung. In Bezug auf die Erkenntnisziele ist dies der notwendigen Komplexitätsbewältigung und der geforderten Praxistauglichkeit der Ansätze bei der Unternehmensgestaltung geschuldet.

Mit der hohen Zahl an existierenden Ansätzen steigt auch die Zahl der Arbeiten, die sich mit der Fragestellung beschäftigen, wie eine Methode zur Modellierung ausgewählt wird bzw. geeignet zu definieren ist. In den letzten Jahren des vergangenen Jahrhunderts wurden diese im Wissensgebiet des **Method Engineering** (ME) zusammengefasst (vgl. [Har+94], S. 170ff). Im Kern werden hier erfolgreich in der Systementwicklung angewendete Verfahren des ingenieurmäßigen Vorgehens auf die Definition dieser Verfahren selbst angewendet. Autoren wie BRINKKEMPER, HARMSEN oder NUSEIBEH beschäftigten sich folgerichtig mit der Schaffung von *Methoden zur Definition von Methoden*. Nicht zuletzt aufgrund der Nähe der Autoren zur Informatik konzentrieren sich ihre Arbeiten auf die Erstellung von *Metamodellen*, die insbesondere die Sprache definieren, welche in den Methoden angewendet wird. Dies hat vor allem den Vorteil, dass u. a. rechnergestützte Werkzeuge zur Methodendefinition entwickelt werden konnten[5] und Methoden vergleichbar wurden.[6] Diese Arbeit wird als einen Schwachpunkt dieser Ansätze jedoch eine fehlende oder mangelhafte Vorgehensbeschreibung und damit eine unzureichende Unterstützung der Informationsmodellierung nachweisen.

Die Aufgabenbereiche des ME umfassen den Entwurf, die Konstruktion und Evaluierung von Methoden (vgl. [Bri+96a], S. vjj). Damit werden für die Entwicklung von IKS Techniken und Werkzeuge zur Verfügung gestellt, worin auch die Ursache für die Adaptierung dort etablierter Techniken für das ME liegt. Mit allen in Abschnitt 1.1 formulierten Zielstellungen ordnet sich

[4] Stellvertretend für zahlreiche Quellen wird hier auf FERSTL und SINZ (vgl. [FeSi01], S. 119) und BALZERT (vgl. [Balz92], S. 4) verwiesen.

[5] MARTIIN ET AL. liefern eine zusammenfassende Betrachtung der Werkzeuge Maestro II, Decamerone und MetaEdit (vgl. [Mar+96], S. 64ff). Daneben sind vor allem die neueren Werkzeuge ADONIS (vgl. [Jun +00]), ParadigmPlus, JKogge, ToolBuilder, ObjectMaker, MetaView und GraphicalDesigner erwähnenswert. (vgl. [Meta00])

[6] siehe dazu z. B. die Arbeiten von STRAHRINGER (vgl. [Stra96]) oder FRANK ET AL. (vgl [Fran98a])

diese Arbeit auf dem Gebiet des ME ein.

## 1.3 Aufbau der Arbeit

Die Arbeit gliedert sich in drei Teile. Der Teil I liefert mit dem Kapitel 2 die begriffliche Grundlage. ORTNER belegt in einer Veröffentlichung, wie wichtig die Auseinandersetzung mit den Grundbegriffen einer Wissenschaftsdisziplin ist. Er unterstreicht die Bedeutung eines differenzierten Gebrauchs dieser gerade für eine interdisziplinäre Wissenschaft wie die der Wirtschaftsinformatik (vgl. [Ortn02], S. 31). Der im einleitenden Teil fundierte Begriff des Informationsmodells wird für die nähere Untersuchung in den späteren Kapiteln klassifiziert. Das anschließende Kapitel 3 beschäftigt sich mit den Grundlagen der Erstellung von Informationsmodellen. Für den dort beschriebenen Lebenszyklus wird eine methodische Unterstützung vorgestellt, die mit den Ausführungen in Kapitel 3.4 besonders dem notwendigen Änderungsmanagement in iterativen Prozessen Rechnung trägt.

Der Teil II der Arbeit konzentriert sich auf das Method Engineering. Die Prozesse auf diesem Gebiet werden aufgrund von Anforderungen an deren Ergebnisse gestaltet, sodass die Bewertung von Methoden im Kapitel 4 eine zentrale Bedeutung besitzt. Hier wird die Zielstellung der Schaffung einer Methode zur Auswahl, Anpassung und Entwicklung von Methoden der Informationsmodellierung operationalisiert. Für das Teilziel der Bewertung vorhandener Ansätze werden konkrete Verfahren vorgestellt und eine Technik integriert, die ein Controlling der Ergebnisqualität ermöglicht. Mit dem anschließenden Abschnitt 5 wird das Teilziel der Bewertung relevanter Ansätze auf dem Gebiet des ME erreicht und gleichzeitig aufgrund der festgestellten Mängel die Motivation für die nachfolgenden Ausführungen dargelegt.

Mit dem Teil III wird in Kapitel 6 das $E^3$-Modell vorgestellt, in Kapitel 7 durch geeignete Wiederverwendungsansätze und in Kapitel 8 durch ein Vorgehensmodell ergänzt. Das mit dem Kapitel 9 spezifizierte Konfigurationsmanagement ermöglicht in Kombination mit den Wiederverwendungsansätzen der Kapitel 7.1 und 7.2 eine hohe Praxisrelevanz der Methode. Mit der entstehenden $E^3$-Methode ist ein Method Engineering möglich, das den Anforderungen aus dem Kapitel 4 dieser Arbeit genügt. Diese subjektive Einschätzung des Autors wird in Abschnitt 10 derart dokumentiert, dass zum einen ein Vergleich mit den in Teil 5 beschriebenen Methoden möglich wird und zum anderen die Meinungsbildung des Lesers unterstützt wird.

Der Anhang im Teil IV enthält neben einigen Ausführungen zur Qualität von Methoden die vollständige Spezifikation des $E^3$-Modells, dessen Referenzmetamodells und Konfigurationsmanagementsystems. Die Abbildung 2 fasst den Aufbau der ersten drei Teile zusammen.

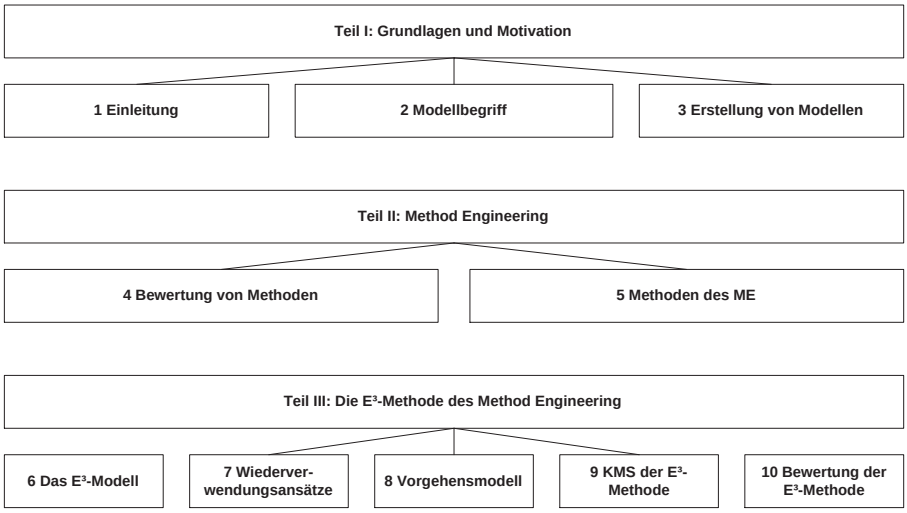



Abbildung 2: Aufbau der Arbeit

# Teil I
# Grundlagen und Motivation

Die folgenden Kapitel im Teil I der Arbeit werden auf Basis des Zeichen- und Systembegriffes Modelle definieren und in Umkehrungen deren Rolle bei der Repräsentation von Systemen beschreiben. Für die praktische Anwendung des Modellbegriffes in der Wirtschaftsinformatik ist dessen Diversifikation zur Beschreibung der Anwendungsfälle von hoher Bedeutung. So bilden sie zum einen Produkte der Systementwicklung und des Method Engineering und dienen zum anderen als ein Hilfsmittel bei deren Gestaltung.

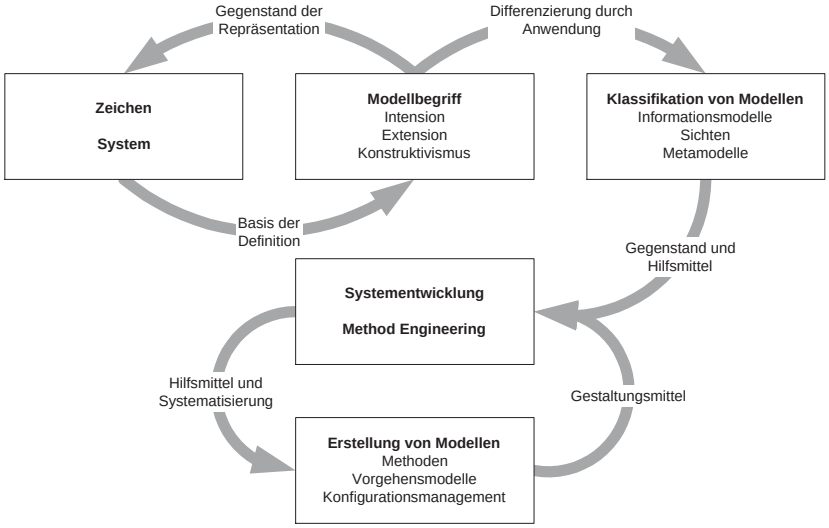



Abbildung 3: Gedankengang in Teil I

# 2 Modellbegriff

Für diese Arbeit wird die Definition des Modellbegriffes für zwei Ziele verfolgt. Auf der einen Seite liegt nahe, dass Modelle Produkte des Prozesses der Modellerstellung sind, auf der anderen Seite lässt sich mit Modellen dieser Prozess auch beschreiben.[7] Hierzu wird in einem ersten Schritt der grundlegende Ansatz in der Semiotik als Lehre vom Zeichen erläutert. Aus diesem und einer anschließenden kurzen Einführung in den Systembegriff lässt sich die Intension[8] des Modellbegriffes und dessen Extension[9] beschreiben, welcher zu einer klaren Definition von Informationsmodellen führt, mit deren Klassifikation dieser Abschnitt endet.[10]

## 2.1 Sprache und Zeichen

Der Austausch von Informationen wird nicht nur in der Informatik allgemein durch die Kette Quelle – Sender – Kanal – Botschaft – Empfänger beschrieben (vgl. [ShWe49]). In diesem Kommunikationsprozess werden Zeichen zum Austausch von Informationen verwendet. Die Botschaft ist dementsprechend gleichbedeutend mit dem Zeichen. Sender und Empfänger müssen dabei über eine möglichst gleiche Menge von Regeln verfügen, die den Zeichen einen Sinn zuordnen. Unterschieden wird seit den Stoikern[11] (vgl. [Eco77], S. 28):

- **semaion**: das eigentliche Zeichen als Entität,
- **semainomenon**: das, was vom Zeichen ausgesagt wird und keine physische Entität darstellt und
- **pragma**: der Gegenstand, auf den sich das Zeichen bezieht.

Diese Unterteilung wurde vielfach in der Semiotik aufgegriffen und unter der Bezeichnung **semiotisches Dreieck** mit zahlreichen unterschiedlichen Begriffen an den Eckpunkten belegt. Die triadische Zeichenrelation kann im Deutschen so beschrieben werden: „Ein Zeichenmittel (M) bezieht sich auf ein Objekt (O) und wird von einem dritten Moment, dem Interpretanten (I), als in dieser Beziehung stehend interpretiert.“ ([Schö99], S. 9) Die Existenz eines *direkten* Bezuges des Zeichenmittels zum Objekt wird in der Semiotik aufgrund erkenntnistheoretischer Überlegungen allgemein verneint.

[7] Die Vorteilhaftigkeit dieses Vorgehens wird der Abschnitt 3 nachweisen.
[8] Der Sinn und Inhalt eines Begriffes
[9] Gesamtheit der Gegenstände, die unter einen Begriff fallen
[10] Damit wird auch der entsprechenden DIN Rechnung getragen, die zur Bildung von Definitionen eine Festlegung von Begriffsinhalt und -umfang verlangt (vgl. [Deut93], S. 4).
[11] Anhänger der Stoa, eine philosophische Schule im antiken Griechenland, die ein von Vernunft und Übereinstimmung mit sich selbst und der Natur bestimmtes Leben forderte

Als intrinsisch wird die Beziehung zwischen Mittel und Interpretant betrachtet: „Das Bild einer Tomate zeigt eine Tomate, es hat eine Tomate zum Inhalt.“ ([Schö99], S. 46) Im Gegensatz dazu ist die Beziehung zwischen Mittel und Objekt extrinsisch. Die Diskussionen um die Beziehungen bezeichnet SCHÖNRICH als die „... semiotischen Kriege, die um die Repräsentationsfunktion M-O ...“ geführt werden ([Schö99], S. 47). Ableiten lässt sich aus diesen trotz zahlreicher unterschiedlicher Meinungen, dass auch eine direkte Beziehung zwischen Mittel und Objekt nicht bestehen kann. Vielmehr bedarf sie eines *Interpretanten zweiter Stufe* (zu weiteren Ausführungen siehe [Schö99], S. 45ff).

Die Wirkung des Repräsentamens (Zeichenmittels) auf den Interpreten, also den Akt der Bedeutungssuche, nannte PEIRCE **Semiose**. Der Interpret kann nur eine Beziehung zum Interpretanten aufbauen, dieser selbst wird in einer weiteren Stufe zum Objekt der Semiose. ECO beschreibt die Semiose als offen, es genügt die Erkennung als Zeichen und anschließende Auslegung, die eindeutige Sinndeutung (und damit Erkennung des Objekts) ist nicht gefordert bzw. überhaupt in diesem Sinne möglich.

**Sprache** kann in diesem Zusammenhang als ein System von Zeichen und Regeln für deren Anwendung definiert werden, man spricht daher bei Sprachen auch von *Zeichensystemen* (vgl. [Stra96], S. 17). Auf Basis der Sprachwissenschaft wird präzisiert: Die **Zeichen (Sprachzeichen)** sind stets *materiell* gegeben (akustisch bzw. graphisch) und sie haben immer *Form* und *Inhalt* (vgl. [Bra$^+$99], S. 2).[12] Die Charakteristik einer Sprache bestimmt sich durch die Ebenen der *Syntax*, der *Semantik* und der *Pragmatik* (vgl. [Stac73], S. 197, Fußnote 108; [Bra$^+$99], S. 2ff und [Zema92], S. 72ff):[13]

- Die **Syntax** bezieht sich auf die Relation der Zeichen zueinander (*1-stellige Relation*). Sie legt Regeln zur Verknüpfung der Zeichen fest.
- Die **Semantik** bezieht sich auf die Relation der Zeichen zum Bezeichneten (*2-stellige Relation*). Sie legt die *Bedeutung* des Zeichens fest, d. h. was es aus der geistigen oder physikalischen Welt beschreiben soll. Da dieses weltliche Element mit einem Zeichen beschrieben (belegt) wurde, nennt man es auch bezeichnet.
- Die **Pragmatik** bezieht sich auf die Relation Zeichen - Bezeichneter - Bezeichnender (*3-stellige Relation*). Sie legt fest, was derjenige, der das Zeichen benutzt, damit meinen kann. Die Pragmatik determiniert so das Ziel einer sprachlichen Handlung. Der Akteur der sprachlichen Handlung ist der *Bezeichner (Sprecher)*.

[12] Die hier zugrunde gelegte Definition von *Zeichen (Sprachzeichen)* entspricht der von Zeichen aus *taxemischen Zeichensystemen* (vgl. [Stac73], S. 198). Da das Sprachzeichen *Form* und *Inhalt* haben muss, stellen bspw. die Buchstaben des deutschen Alphabets keine Sprachzeichen dar, ihnen fehlt der Inhalt (vgl. [Bra$^+$99], S. 2).

[13] HEINATZ fügt diesen Ebenen die der Hermeneutik zur Interpretation der Kommunikationssituation hinzu (vgl. [Hein98], S. 45f). Deren Bindung an den Anwendungsfall verhindert jedoch eine wiederverwendbare Beschreibung.

Semantik und Pragmatik hängen wechselseitig voneinander ab. Was eine sprachliche Äußerung bedeutet, wird durch die Meinung des Bezeichnenden festgelegt (Sprachkonvention). Was dieser mit einer sprachlichen Äußerung meinen kann, lernt der Bezeichnende über die Erfahrung, was die Äußerung in dieser Sprache bedeutet. (vgl. [Bra+99], S. 2).

## 2.2 Systembegriff

Der Systembegriff ist Gegenstand zahlreicher Wissenschaftsbereiche[14] und mit dementsprechend vielgestalten Inhalten belegt. Im allgemeinen Sprachgebrauch wird mit ihm das *einheitlich geordnete Ganze* beschrieben (vgl. [Wiss96], S. 726). In seiner ursprünglichen Bedeutung als Zusammenhang wurde er bereits von den griechischen Philosophen PLATON und ARISTOTELES in ihrer ganzheitlichen Denkweise diskutiert. BEER definiert ein System als eine Ansammlung miteinander in Beziehung stehender Teile (vgl. [Beer59], S. 24).[15] Es bildet den Gegenstandsbereich insbesondere der System-Kybernetik[16] und Systemtheorie[17].

Bei der Bildung von Modellen wird nach KNUTH das System als Ganzes betrachtet, das System strukturiert und das koordinierte Zusammenwirken im System beschrieben (vgl. [Knut95], S. 49). Gegenstand der Abbildung können die Struktur als Menge von Elementen und deren Beziehungen, das Verhalten als Zusammenwirkung dieser Elemente und das dabei verfolgte Ziel sein (vgl. [Knut95], S. 51). KRALLMANN ergänzt diese Definition durch die Abgrenzbarkeit des Systems gegen seine Umwelt, mit der es über Input- und Outputbeziehungen interagiert (vgl. [Kral94], S. 6). Zudem werden Komponenten, die nicht weiter in Subsysteme zerlegt werden können oder sollen, **Systemelemente** genannt. Schließlich wird auch der Gedanke des **Systemzwecks** berücksichtigt. Ein grundlegender Systemzweck liegt oft schon in der Erhaltung des Systems, weitere differenziertere Zwecke sind jedoch denkbar. Systemelemente können selbst wiederum Systeme sein, die in diesem Fall auch als **Sub-** oder **Teilsysteme** (vgl. [FeSi01], S. 12 und [Gese01]) bezeichnet werden.

Der Begriff des **Informationssystems (IS)** leitet sich aus der modellhaften Unterteilung von Unternehmen in ein Leistungssystem zur Leistungserstellung und ein Lenkungssystem zur Planung, Steuerung und Kontrolle des Leistungssystems ab (vgl. [FeSi01], S. 4ff). Im Kontext betrieblicher Systeme in Unternehmen und öffentlichen Verwaltungen wird der informationsverarbeitende Teil des Leistungssystems und das gesamte Lenkungssystem unter dem Begriff des Informationssystems zusammengefasst. Aus Sicht der Aufgabenträger besteht dieses aus

[14] z. B. in der Biologie als Beschreibung ökologischer Zusammenhänge; in der Astronomie zur Gruppierung von Körpern im Raum; in der Volkswirtschaftslehre z. B. als Grundlage der Spieltheorie

[15] Allgemein wird unter einem System eine funktionale Einheit aus mehreren Einzelteilen verstanden, die zur Ausführung einer oder mehrerer Aufgaben dient. Daneben wird unter dem Begriff des Systems auch die gesellschaftliche Ordnung bzw. Organisationsform eines Staates verstanden (vgl. [Micr98]).

[16] Theorie dynamischer Systeme; das Wort *kybernetes* ist griechischen Ursprungs und bedeutet Steuermann

[17] Theorie der statischen Systeme

automatisierten und nicht automatisierten Teilen (vgl. [Schu01], S. 34).

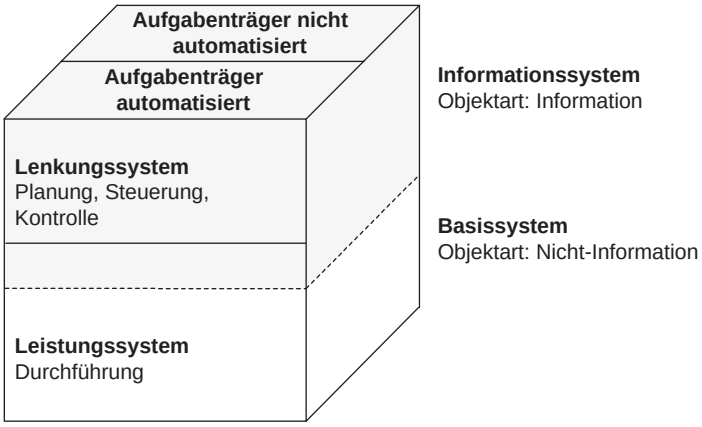



Abbildung 4: Abgrenzung von Informationssystemen (i. A. a. [FeSi01], S. 4)

## 2.3 Intension des Modellbegriffes

Der Begriff Modell wird in verschiedenen Sinnzusammenhängen verwendet. Der Ursprung im Spätlateinischen *modellus* steht insbesondere für ein Muster oder Vorbild. Die Verwendung erstreckt sich in der Umgangssprache jedoch von Nachbildungen im kleineren Maßstab, Gießformen oder Ausführungsarten bis hin zu Personen oder Sachen als Vorbild für ein Werk der bildenden Kunst, Mannequins und einer eher zweideutigen Umschreibung. Die Abschnitte 2.1 und 2.2 bieten eine erste Annäherung an den Modellbegriff dieser Arbeit, indem Modelle als System von Zeichen interpretiert werden. So versteht KRALLMANN unter einem Modell „... ein abstraktes System, das ein anderes (meist reales) System in vereinfachter Weise abbildet." ([Kral94], S. 12) Ergänzend bezeichnet CHROUST ein Modell als ein Objekt, das „... Analogieschlüsse in Bezug auf ein anderes Objekt (das 'Original') gestattet." ([Chro92], S. 37) Beide Definitionen stehen exemplarisch für die Fülle von Begriffsfestlegungen, die in der Literatur zu finden sind.

Gemäß den Ausführungen von STACHOWIAK wird an dieser Stelle der mögliche Begriffsumfang beschrieben. Dieser ist „... identisch mit der Gesamtklasse der unter den Begriff fallenden Individuen ..." ([Stac73], S. 129) und häufig ausdrucksstärker als eine definitorische Aussage. In den folgenden Abschnitten dieser Arbeit wird der Begriff jeweils unterschiedlich konkretisiert.

Auf die Angabe des zugrunde liegenden modellistischen Erkenntniskonzepts wird an dieser Stelle verzichtet, es sei jedoch darauf hingewiesen, dass die Beschreibungen auf Tatsachenfeststellungen von STACHOWIAK basieren, die er im vorwissenschaftlichen und wissenschaftlichen Kontext durchführte. Identifiziert wurde das Abbildungsmerkmal, das Verkürzungsmerk-

mal und das pragmatische Merkmal. STACHOWIAK beschreibt den so entstehenden allgemeinen Begriff des Modells als am inhaltsärmsten, da „... je größer und weiter der Umfang des Begriffs, desto kleiner und ärmer sein Inhalt, und umgekehrt“ ([Stac73], S. 130).

### Abbildungsmerkmal

„Modelle sind stets Modelle von etwas, nämlich Abbildungen, Repräsentationen natürlicher oder künstlicher Originale ...“ ([Stac73], S. 131). Rekursiv können diese selbst zum Original weiterer Abbildungen werden und bilden so die Grundlage des im Abschnitt 2.5.4 gebildeten Metamodellbegriffes. Mengentheoretisch beschreibt dieser Begriff die Zuordnung von Attributen einer als Original bezeichneten Menge auf eine Menge von Modell-Attributen.

### Verkürzungsmerkmal

Modelle erfassen im Allgemeinen nicht alle Attribute des durch sie repräsentierten Originals (vgl. [Stac73], S. 132). Da die Auswahl durch den Modellersteller bzw. Modellnutzer getroffen wird und somit unterschiedlichen Graden subjektiver und intersubjektiver Eindeutigkeit unterliegt, sieht STACHOWIAK bereits das pragmatische Merkmal im weitesten Sinne vertreten.

### Pragmatisches Merkmal

Die Subjektivität bei der Modellerstellung und -nutzung begründet die Auswahl von Attributen, die nur durch bestimmte Subjekte, zu festgelegten Zeitintervallen und unter Nutzung ausgewählter gedanklicher oder tatsächlicher Operationen erfolgt. Es muss bei der Betrachtung stets berücksichtigt werden, wovon etwas, für wen, wann und wozu bezüglich seiner Funktion Modell ist (vgl. [Stac73], S. 133).

Zur Bestimmung der Modellintension aus erkenntnistheoretischer Sicht zieht die Literatur zumeist den abbildungsorientierten oder konstruktivistischen Modellgedanken heran.[18] Bei Annahme eines abbildungsorientierten Verständnisses ist ein Modell als „... eine vereinfachende und abstrahierende Darstellung eines Realitätsausschnitts ...“ ([Schü98], S. 52) zu interpretieren, „... anhand dessen die wichtigsten Eigenschaften eines Originals erkannt, verstanden und analysiert werden können.“ ([Sti+97a], S. 449) Ähnliche Auffassungen vertreten beispielsweise auch FERSTL und SINZ (vgl. [FeSi01], S. 18), ROSEMANN (vgl. [Rose96], S. 17f) und DINKELBACH (vgl. [Dink73], S. 161).

[18] Daneben werden aber auch der strukturorientierte und prädikatenlogische (vgl. [Schü98], S. 46, Fußnote 45), der kritisch-rationalistische (vgl. [Jäge82], S. 131) und der axiomatische Modellbegriff (vgl. [Schu01], S. 16ff) diskutiert. Aufgrund der geringen Verbreitung letztgenannter Ansätze beschränken sich die Betrachtungen hier jedoch auf die Abbildungsorientierung und den Konstruktivismus.

Im Gegensatz dazu definiert sich bei Zugrundelegung des Konstruktivismus ein Modell nicht mehr über die Abbildungsrelation zwischen Original und Modell, sondern als „... das Ergebnis einer Konstruktion eines Modellierers, der für Modellnutzer eine Repräsentation eines Originals zu einer Zeit als relevant mit Hilfe einer Sprache deklariert ...“ ([Schü98], S. 59). Vergleichbare Ansichten finden sich auch bei BRETZKE (zitiert bei Herrmann [Herr91], S. 116ff), MOLIÈRE ([Moli84], S. 125) und HERRMANN ([Herr91], S. 128).

Gegen die Verwendung eines abbildungsorientierten Modellverständnisses sind zwei Hauptkritikpunkte vorzubringen. Zunächst stellt sich die Frage, ob eine Abbildung der Strukturen der Realität (also des Originals bei Istmodellen) überhaupt möglich ist. Voraussetzung hierfür wäre ihre objektive Wahrnehmung durch den Modellersteller, die jedoch von WATZLAWIK angezweifelt wird: „Jede Wirklichkeit ist die Konstruktion derer, die die Wirklichkeit zu entdecken glauben.“ ([Watz85], S. 9) In diesem Fall „... müssen Probleme als strukturierte, empirisch vorfindbare und damit einer bloßen Abbildung zugängliche Sachverhalte verstanden werden.“ ([Herr91], S. 117) Somit enthielten wahrnehmbare Probleme bereits die zugehörigen Lösungen, die durch Anwendung bekannter Umformoperationen herausgearbeitet werden könnten. Demgegenüber ist ein Problem jedoch im Allgemeinen „... durch einen Spannungszustand oder eine Diskrepanz zwischen einem mehr oder weniger gut definierten Istzustand und einem Sollzustand gekennzeichnet.“ ([Moli84], S. 78) Auslöser desselben sind Ziele, auf deren Grundlage die derzeitige Situation als unbefriedigend wahrgenommen wird. Sie sind stets als subjektgebunden zu betrachten, womit im Umkehrschluss auch Probleme subjektspezifisch und somit nicht objektiv erkennbar sind (vgl. [Moli84], S. 79f). Wird ergänzend die Problemdefinition von DRESBACH zugrunde gelegt, so zeichnen sie sich u. a. auch durch eine gewisse Schwierigkeit beim Erreichen des Sollzustands aus (vgl. [Dres99], S. 78). Vertreter der Abbildungsorientierung berücksichtigen diesen Sachverhalt, indem bei der Modellierung realer Systeme zunächst (subjektive) Interpretationen der Diskurswelt gefordert werden (vgl. auch [FeSi01], S. 18 und [Rose96], S. 19). Dies führt jedoch den Gedanken einer, im Idealfall sogar isomorphen, Abbildungsrelation zwischen Original und Modell ad absurdum.

Den zweiten Kritikpunkt der Abbildungsorientierung bildet der passiv-rezeptive Modellierungsprozess, nach dessen Verständnis ein Modell durch Beobachtung entsteht. Dies ist mit den oben dargelegten Einschränkungen bez. der Original-Modell-Abbildungsrelation zwar für Istmodelle, allerdings nicht für die Erstellung von Sollmodellen möglich (vgl. [Schü98], S. 58f). Vielmehr konstruiert der Modellierer bei Bildung letzterer eine innere Vorstellung (internes Modell), die durch geeignete Kommunikationssysteme externalisiert wird. Aufgrund der diskutierten Unzulänglichkeiten der Abbildungsorientierung wird im Weiteren das konstruktivistische Modellverständnis zugrunde gelegt.

Aus der Betonung der Konstruktionsleistung des Modellerstellers leitet sich ein großer Einfluss desselben auf das zu entwickelnde Modell ab. Er ist insb. durch das *Wertesystem* des Ent-

wicklers, seine *Psyche* und die *Gesamtheit seiner Kenntnisse* geprägt ([Müll80], S. 473). Aus dieser Prämisse sind für die Unterstützung des Modellerstellungsprozesses zwei Konsequenzen abzuleiten. Es muss zum einen im Rahmen der Konstruktion ein ständiger Abgleich zwischen den Anforderungen des späteren Nutzers und dem Verständnis des Erstellers stattfinden, um die Erfüllung des pragmatischen Modellmerkmals sicherzustellen. Besondere Relevanz erhält die Problematik, wenn die Person des Modellierers und Anwenders auseinander fallen. Die zweite Konsequenz betrifft die kollektive Modellerstellung. Da die oben erwähnten Einflussdeterminanten für jeden Modellierer unterschiedlich ausgeprägt sein können, muss zwischen den Entwicklern ebenfalls ein ständiger Abstimmungsprozess stattfinden, um die Erfüllung der Zielsetzungen sicherzustellen.

Zusammenfassend lässt sich feststellen, dass sich Modelle im Sinne dieser Arbeit durch die Erfüllung der Modellmerkmale nach STACHOWIAK auszeichnen und deren Erstellung maßgeblich durch die Konstruktionsleistung des Modellierers geprägt ist. Der verwendete Modellbegriff setzt sich im Wesentlichen aus dem Modellverständnis von MOLIÈRE (vgl. [Moli84], S. 100) und der Betrachtung des Prozesses der Modellbildung nach SCHÜTTE (vgl. [Schü98], S. 60ff) zusammen, wie er in Abbildung 5 dargestellt ist.[19]

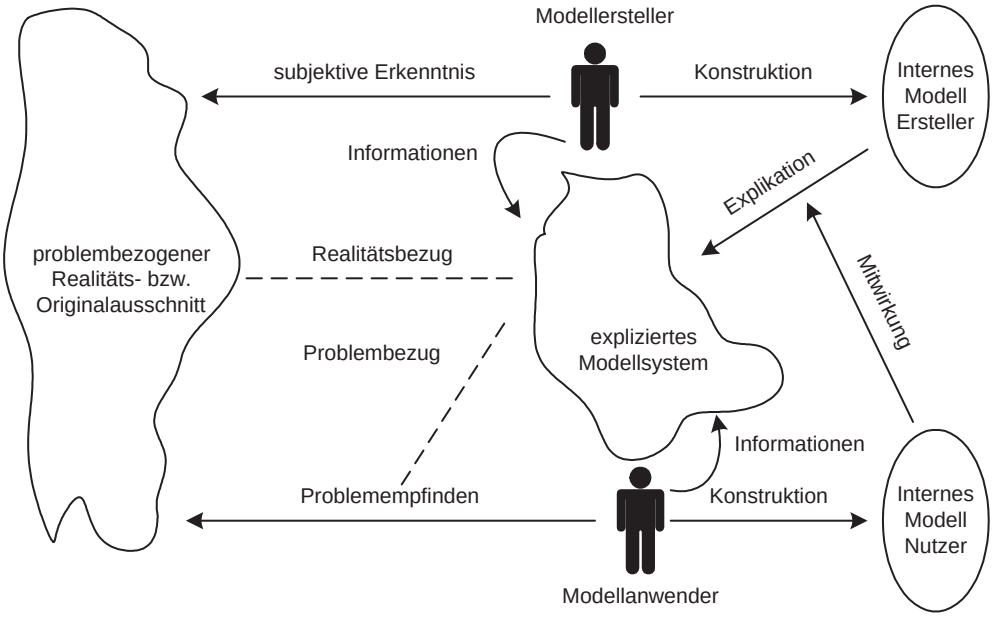



Abbildung 5: Graphische Darstellung des verwendeten Modellbegriffes in Anlehnung an MOLIÈRE (vgl. [Moli84], S. 100) und SCHÜTTE (vgl. [Schü98], S. 61)

[19] Dieses Modellverständnis der Informatik deckt sich weitgehend mit dem mathematischen Verständnis und wird durch die Anwendungsfelder der Informatik ergänzt (vgl. [Ortn02], S. 46).

## 2.4 Extension des Modellbegriffes

Mit dem Bezug zur allgemeinen Modelltheorie von STACHOWIAK wurde durch Abschnitt 2.3 ein sehr weiter Begriffsraum aufgespannt, wie in Abbildung 6 auszugsweise dargestellt ist. Ähnliche umfangreiche Begriffsfelder werden auch von JÄGER (vgl. [Jäge82], S. 144) und MOLIÈRE (vgl. [Moli84], S. 51) angeführt. Der damit aufgespannte Begriffsraum ist für die Zwecke dieser Arbeit, auch im Hinblick auf ihre Positionierung im Bereich der Wirtschaftsinformatik, geeignet einzuschränken. In diesem Sinne werden im Folgenden Modelle als Beschreibungsmittel für Organisationen und Softwaresysteme näher beschrieben, da der Erkenntnisgegenstand der Wirtschaftsinformatik „… Informations- und Kommunikationssysteme (IKS) in Wirtschaft und Verwaltung‘ …“ sind ([Wiss94], S. 80). Zudem besitzen sie in beiden Bereichen die Eigenschaft, strukturelle und dynamische Zusammenhänge zu beschreiben (vgl. [Wint00], S. 13).

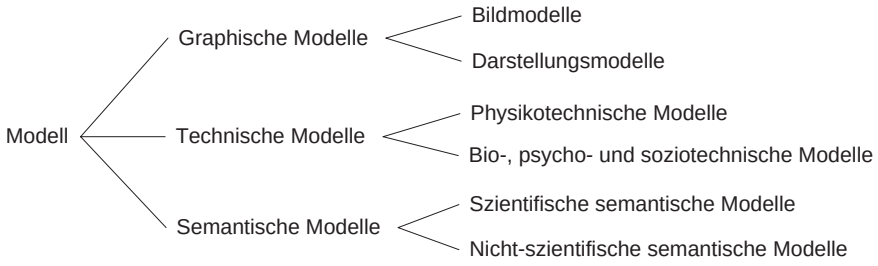



Abbildung 6: Auszugsweise Darstellung des Begriffsraums nach den allgemeinen Modellmerkmalen von STACHOWIAK (vgl. [Stac73], S. 159ff und 196ff)

Informationsmodelle „… bilden wesentliche Grundlagen für die Spezifikation und den Entwurf von Informationssystemen“ ([PiMa94], S. 107), sodass sie als zentraler Modelltyp der Wirtschaftsinformatik verstanden werden können (vgl. [Schü98], S. 63). Deshalb wird der Modellbegriffsraum in dieser Arbeit auf die Klasse der **Informationsmodelle (IM)** beschränkt, die das „… immaterielle Abbild des betrieblichen Objektsystems aus Sicht der in diesem verarbeiteten Informationen für Zwecke des Informationssystem- und des Organisationsgestalters.“ darstellen ([Bec+95], S. 435). Sie ordnen sich in die Gruppe der szientifisch semantischen Modelle ein, deren Grundlage stets erfahrungswissenschaftliche Erkenntnisse bilden (vgl. [Stac73], S. 242ff).[20]

[20] Dadurch grenzen sich diese von den nicht-szientifischen Modellen ab, die zwar ebenfalls eine Semantik besitzen, sich jedoch durch einen künstlerischen oder allgemein narrativen, also erzählenden und feststellenden, Charakter auszeichnen.

## 2.5 Klassifikation von Informationsmodellen

Eine weitere Klassifikation des Modellbegriffes erweist sich nach einer Erörterung der Verwendungsweisen und Fixierung des Begriffes als erforderlich, da dessen Attributierung in der Wirtschaftsinformatik zu unterschiedlichen Verwendungsweisen führt. Mit dem Bezug auf Informationsmodelle in der Wirtschaftsinformatik wurde bereits auf eine mögliche Klassifikation (nämlich die nach dem Gegenstandsbereich) hingewiesen. Aufgrund der Tatsache, dass weitere Dimensionen existieren und deren Verwendung einer vorangestellten Explikation bedarf, wird der begriffliche Umfang klassifiziert.

### 2.5.1 Formale Modelle und Modellierungssprachen

Den Zusammenhang zwischen Sprachen und Zeichen beschreibt ECO in der Aussage „Sobald eine beobachtbare und interpersonale Form sichtbaren Zeichenverhaltens zustande kommt, ist eine Sprache da.“ ([Eco77], S. 109) In diesem Sinne werden Systeme zusammenhängender Zeichen zu Sprachen, welche sich wiederum klassifizieren lassen durch die enthaltenen Zeichen. MORRIS unterscheidet **ikonische Zeichen** mit anschaulich bildhafter Darstellung des Originals und **symbolische Zeichen** ohne direkten Bezug zum Original (vgl. [Morr72], S. 97f). Deren Wortschatz wiederum wird in lang- und kurzsymbolisch unterteilt. Zu den langsymbolischen (oder auch natürlichen) Sprachen werden insbesondere Fachsprachen und die Umgangssprache gezählt. Auf Seiten der künstlichen (oder auch kurzsymbolischen) Sprachen werden **formale** (oder auch mathematische) und **nicht formale** Sprachen unterschieden. REMME definiert, dass formale Sprachen eine formale Syntax (Regeln zur Bildung von Relationen zwischen Zeichen) besitzen und sich durch eine mathematisch korrekte Definition ihrer Sprachregeln auszeichnen (vgl. [Remm97], S. 43). In Bezug auf Sprachen zur Bildung von Informationsmodellen wird häufig der Begriff der **semi-formalen** Sprache geprägt, bei der zumindest „... Teile der Sprachelemente als formale Sprache zu verwenden sind.“ ([Remm97], S. 44) Er dient vor allem der Abgrenzung nicht formaler (aber kurzsymbolischer) Modellierungssprachen von langsymbolischen Sprachen wie der Umgangssprache. Ein Vertreter dieser Kategorie ist die in Ereignisgesteuerten Prozessketten (EPK) nach SCHEER verwendete Sprache, zu der keine formale Definition ihrer Semantik existiert (vgl. [Lan$^+$98], S. 286).

DIETZSCH beschreibt den Grad der Formalisierung als Voraussetzung für die Durchführung der in den Sprachen beschriebenen Verfahren durch einen Automaten. Formal beschriebene Lösungsverfahren sind demnach vollständig automatisierbar.[21] Für semiformale Beschreibungen lässt sich diese Möglichkeit nicht ausschließen, grundsätzlich wird jedoch von einer teilweisen Automatisierbarkeit ausgegangen. Informale Beschreibungen sind nicht automatisierbar

[21] Dies basiert auf der Annahme, dass sich alle Elemente einer formalen Beschreibung auf maschineninterpretierbare Elemente abbilden lassen (vgl. [Diet02], Fußnote 52).

(vgl. [Diet02], S. 59). Die Abbildung 7 stellt diesen Zusammenhang zwischen dem Grad der Formalisierung einer Beschreibung und deren Automatisierbarkeit im Überblick dar.

| | informal | semiformal | formal |
|---|---|---|---|
| formale Syntax | nein | ja | ja |
| formale Semantik | nein | nein | ja |
| Automatisierbarkeit | nein | teilweise | vollständig |

Abbildung 7: Grade der Formalisierung und deren Automatisierbarkeit (vgl. [Diet02], S. 58)

Neben der verwendeten Sprache lassen sich auch Modelle selbst anhand der Dimensionen formale (oder auch strukturelle) und inhaltliche (oder auch materiale) Angleichung an das Original unterscheiden (vgl. [Stac73], S. 140f). Formale Modelle sind nach STACHOWIAK „... Repräsentanten von Gedankendingen ..., denen keine beobachtbaren Signalmannigfaltigkeiten der äußeren Welt zugeordnet sind." ([Stac73], S. 244f)[22] Informationsmodelle können in diesem Sinne formal oder inhaltlich an ihr Original angeglichen werden.

### 2.5.2 Sichten und Paradigmen

Allgemein werden Modelle zur Komplexitätsbewältigung bei der Erstellung betrieblicher IS eingesetzt. Die hierfür eingeführten Methoden stellen zumeist eine mehrstufige Abstraktion und spezielle Sichten auf den Gegenstandsbereich zur Verfügung (vgl. [FeSi01], S. 119). Sichtweisen auf ein Modellsystem werden als **Metaphern** bezeichnet. Diese beschreiben allgemein die Übertragung von sprachlichen Ausdrücken von einem Bedeutungszusammenhang auf einen anderen, ohne dass diese miteinander vergleichbar sind (vgl. [Wiss00], S. 865).[23] In der Systementwicklung bezeichnen Metaphern die „... Sichtweise, die der Modellierer der Erfassung des Objektsystems zugrundelegt und auf die Spezifikation des Modellsystems überträgt." ([FeSi01], S. 122f)

Neben Metaphern bieten **Sichten** auf ein IS eine Hilfestellung bei der Erstellung von Modellen. Diese werden durch das *Prinzip der Strukturierung* gebildet (vgl. Abschnitt 3.2). Anders als bei der Identifikation und Abgrenzung von Elementen eines Systems steht hier deren Gruppierung im Vordergrund. Mit diesem Hilfsmittel lassen sich Teilspezifikationen bilden, die auf unterschiedlichen Modellierungssprachen beruhen können (vgl. [FeSi01], S. 127) oder unter Verwendung der gleichen Sprache Subsysteme entstehen lassen (vgl. [Knut95], S. 91ff). FERSTL und SINZ unterteilen Sichten grundsätzlich in *statisch* (auf die Funktionen, Datenstrukturen und Kommunikationskanäle des IS) und *dynamisch* (auf die Vorgänge). Diese wer-

[22] Abgezielt wird insbesondere auf die kurzsymbolisch mathematische Sprachverwendung, bei welcher kein direkt bildhafter Bezug des Modells zum Original entsteht.

[23] Zum Beispiel „das Haupt der Familie"

den entsprechend als Funktionssicht F, Datensicht D, Interaktionssicht I und Vorgangssicht V (vgl. [FeSi01], S. 127) bezeichnet. Abbildung 8 stellt ausgewählte Modellierungsansätze mit praktischer Relevanz im Überblick dar.[24]

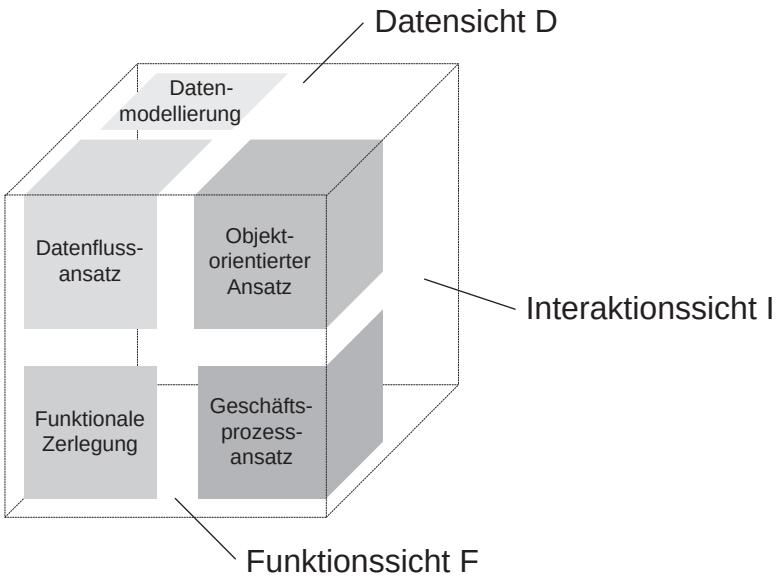



Abbildung 8: Sichten auf IS nach FERSTL und SINZ (i. A. a. [FeSi01], S. 127)

Modellierungssprachen und die damit entstehenden Sichten basieren auf der Verwendung von **Paradigmen**. Im Allgemeinen auch als Beispiel oder Weltsicht verstanden (vgl. [Wiss00], S. 865), bezeichnet es bei der Modellbildung das verwendete Denkmuster. In den wissenschaftstheoretischen Ausführungen von KUHN beschreiben Paradigmen wissenschaftliche Leistungen, die für eine bestimmte Zeit einer Gemeinschaft von Fachleuten maßgebende Probleme und Lösungen liefern (vgl. [Kuhn99], S. 10). Sie bestimmen die Art der Wahrnehmung von Sachverhalten und beeinflussen in diesem Sinne die Pragmatik von Informationsmodellen. Die Entstehung neuer Paradigmen auf dem Gebiet der Informationsmodellierung drückt sich vor allem durch die Definition neuer Sprachen aus, die eine das jeweilige Paradigma unterstützende Semantik besitzen.

### 2.5.3 Referenzmodelle

In der Literatur werden zahlreiche Definitionen für Referenzmodelle genannt. HARS reduziert diese Definitionen auf einen gemeinsamen Kern: „Bei einem Referenzmodell handelt es sich um ein Modell, das für den Entwurf anderer Modelle herangezogen werden kann“ ([Hars94],

[24] Eine Klassifikation, die auch organisatorische Aspekte zur Abbildung soziotechnischer oder soziologischer Systeme in Unternehmen zulässt, wird in Abschnitt 7.2.1.2 vorgestellt. Dort werden die Sichten auf Informationssysteme grundsätzlich in Aufbau-, Aufgaben-, Objekt- und Prozesssicht unterteilt.

S. 15). SCHÜTTE, SCHEER und BECKER ET AL. führen zusätzlich einen normativen Aspekt an (vgl. [Schü98], S. 69; [Sche97], S. 8f; [BeSc97], S. 428). Nach ihrer Meinung gibt ein Referenzmodell Gestaltungsempfehlungen und kann damit als Informationsquelle beispielsweise für die Gestaltung von Geschäftsprozessen herangezogen werden ([Sche97], S. 8). Somit bilden Referenzmodelle Begriffssysteme im Sinne der [Deut93] und sind die Grundlage für eine Vereinheitlichung und Normung der Terminologie (vgl. [Deut93], S. 5). SCHEER betont, dass sich ein Referenzmodell mindestens einmal ohne Veränderung einsetzen lassen muss (vgl. [Sche97], S. 4). HARS hingegen reduziert diesen Aspekt auf die potentielle Nützlichkeit von Referenzmodellen für spezifische Modelle (vgl. [Hars94], S. 15).

Für diese Arbeit wird definiert: Ein **Referenzmodell** ist das immaterielle Abbild der in einem realen oder gedachten Objektsystem verarbeiteten Informationen, das für die Zwecke des Informationssystem- und Organisationsgestalters Empfehlungscharakter besitzt und als Bezugspunkt für die Überführung in spezifische Modelle nützlich ist (in Anlehnung an [BeSc97], S. 428). Nach dieser Definition folgt, dass sich Referenzmodelle auf der selben semantischen Stufe[25] befinden wie die daraus spezialisierten Modelle (vgl. [Schü98], S. 72). Damit grenzt sich das Referenzmodell vom im nächsten Abschnitt diskutierten Metamodell ab, welches auf einer semantisch höheren Stufe als seine Instanzen (Modelle) liegt. Des Weiteren wird nicht gefordert, dass das spezifische Modell vollständig und konsistent gegenüber seinem Referenzmodell sein muss, wie dies bei einem Metamodell der Fall ist (vgl. [Hars94], S. 15). Fasst man zusammen, ergibt sich, dass ein Metamodell von der Semantik des Modells und ein Referenzmodell von dessen Syntax „abstrahiert“ (vgl. [Schü98], S. 72).

Der im Abschnitt 2.4 geschaffene Modellbegriff schließt die Klasse der **Referenz-Informationsmodelle (RIM)** explizit mit in die Überlegungen ein. Dieses stellt „... das Ergebnis einer Konstruktion eines Modellierers, der für Anwendungssystem- und Organisationsgestalter Informationen über allgemeingültig zu modellierende Elemente eines Systems zu einer Zeit als Empfehlungen mit einer Sprache deklariert ...“ ([Schü98], S. 69) dar. Folglich ist bei der Erstellung eines RIM kein expliziter Problem- bzw. Subjektbezug vorhanden, wie er in der hier verwendeten Modellintension (vgl. Abschnitt 2.3) gefordert wurde. Trotzdem werden RIM in die Betrachtung aufgenommen, da sie Typisierungen möglicher Originale darstellen, deren Gültigkeit erst durch die konkrete Anwendung beurteilt werden kann ([Schü98], S. 70). Hierzu ist ihre Konfiguration für die Zwecke des Anwenders notwendig, wodurch sowohl Problem- als auch Subjektbezug des letztendlich angewendeten Modells wieder gegeben sind. Um als Ausgangspunkt für spezifische Modelle nützlich zu sein, ergeben sich eine Reihe von Anforderungen an Referenzmodelle. Zum einen muss ein solches Modell einen gewissen Grad an *Allgemeingültigkeit* besitzen (vgl. [Hars94], S. 15). Um dies zu gewährleisten werden Refe-

[25] Zu den semantischen Stufen siehe [Stac73], S. 196ff.

renzmodelle i. d. R. induktiv aus bestehenden Modellen entwickelt (vgl. [Sche99], S. 7).[26] Deduktiv werden theoretische Kenntnisse eingebracht (vgl. [BeSc97], S. 428). Um möglichst viele spezifische Modelle aus einem Referenzmodell ohne große Veränderungen ableiten zu können, kommt der *Anpassbarkeit* des Referenzmodells eine hohe Bedeutung zu. Ferner sollen Referenzmodelle leicht *anwendbar* sein, um ohne großen Aufwand die spezifischen Modelle zu erhalten. Die letzte Forderung nach *Vollständigkeit*, also der Umfang und der Detaillierungsgrad der abgebildeten Konzepte im Referenzmodell, ist bereits Bestandteil der hier zu Grunde gelegten Definition.

Werden diese Anforderungen konsequent erfüllt, ergibt sich ein hoher Nutzen aus den Referenzmodellen. *Kosten* können eingespart werden, da bei dem Entwurf von Modellen oder Modellteilen auf bereits bestehende Konstrukte des Referenzmodells zurückgegriffen werden kann. Weiterhin tragen Referenzmodelle zur „... Vereinheitlichung unterschiedlicher Sichtweisen innerhalb des Unternehmens ...“ ([Hars94], S. 32) sowie zur Normierung der genutzten Termini bei (vgl. [BeSc97], S. 433). Die damit verbundenen Vereinheitlichungskosten können somit eingespart werden. Eng verbunden mit dem Kostenaspekt ist der *Zeitaspekt*. Da in einem Referenzmodell bereits die grundlegenden Konzepte vorgedacht sind, müssen diese lediglich noch geprüft und angepasst werden (vgl. [Hars94], S. 33). Tendenziell wird ebenfalls die *Qualität* der durch Referenzmodelle erzeugten Modelle höher sein als bei einem neu erstellten Modell, da ein Referenzmodell bereits die Vereinigung von erprobten oder sogar Best-Practice-Lösungen darstellt und ebenfalls theoretische Überlegungen mit einschließt. Bei der Entwicklung von Unternehmensmodellen wird ebenfalls das *Risiko* vermindert, da „... bereits eine rudimentäre Lösung vorliegt, die als Richtschnur für die Entwicklung des unternehmensspezifischen ...“ Modells verwendet werden kann ([Hars94], S. 33).

Den genannten Vorteilen von Referenzmodellen steht der Nachteil gegenüber, dass durch die angestrebte Standardisierung beim Einsatz von Referenzmodellen die Unternehmen ihre strategischen Wettbewerbsvorteile verlieren (vgl. [BeSc97], S. 434). Dem ist aber entgegenzuhalten, dass Referenzmodelle noch entsprechend anzupassen sind. In genau diesen Anpassungen spiegeln sich die Kernkompetenzen und die strategischen Vorteile der Einzelunternehmen wider. Folglich kommt der beschriebene Nachteil erst zum Tragen, wenn eben diese unternehmensspezifische Anpassung unterbleibt (vgl. [BeSc97], S. 434).

[26] Referenzmodelle sollen nach SCHEER möglichst eine „best common practise“, also die sowohl weit verbreitete als auch beste Praktik, beschreiben (vgl. [Sche99], S. 7).

### 2.5.4 Metamodelle

Um eine klare Definition des Begriffes Metamodell zu erhalten, ist es unablässig, den Präfix Meta zu diskutieren.[27] Häufig wird zu diesem Zweck auf einen griechischen Ursprung verwiesen, in welchem es nach Meinung vieler Autoren für das Wort *über* steht.[28]

Abstraktion ist eines der Prinzipien in der Modellbildung (siehe Abschnitt 3.2). Existieren nach deren Anwendung unterschiedliche Abstraktionsstufen, sollten diese auch begrifflich durch den Präfix **Meta** verdeutlicht werden (z. B. werden diese Stufen in einigen Fällen durch das Adjektiv „generisch“ oder den Präfix „Super“ verdeutlicht). Allein aus den Begriffen ist beispielsweise eine Unterscheidung zwischen Objekttyp, Objektklasse, Meta-Objekt oder Superobjekt nicht mehr möglich. Zusammenfassend ergeben sich folgende Forderungen (vgl. [Stra99], S. 1ff) :

- Fundierung des Meta-Begriffes: Es muss eine Basis definiert sein, von der abstrahiert wird (Verankerungspunkt). Eine klare Formulierung des Kriteriums, welches der Abgrenzung der Abstraktionsstufen zugrunde liegt, ist ebenfalls unabdingbar.

- Harte Kriterien bei der Ebenenbildung: Die Abstraktionsstufen bei der Modellierung im Sinne des Abschnitts 2 bezeichnet STRAHRINGER als „hart“. Unscharf werden die Ebenen beim Einsatz generischer (flexibler) Modelle oder beim Einsatz von Referenzmodellen, die mehrere Varianten beinhalten.

- Konsistente Fortführung des Meta-Begriffes: Durch die Bildung mehrerer Abstraktionsstufen entstehen Hierarchien. Wechselt man innerhalb dieser das Kriterium ihrer Abgrenzung, sollte man Teilhierarchien bilden, innerhalb derer das Kriterium gleich bleibt.

Der mit der Meta Object Facility (MOF) definierte und in der Wirtschaftsinformatik häufig verwendete Meta-Begriff der Object Management Group (OMG) ist aufgrund seiner Flexibilität nicht unproblematisch in der Anwendung, da er zu Irritationen führen kann. Dort wird ein Metamodell als „. . . model of some kind of meta-data“ ([Obje00a], S. 2-5) und Meta-Daten als „. . . data whose purpose is to describe other data“ ([Obje00a], S. 2-5) definiert. Offene Fragen sind hier z. B. , ob es sich bei Metamodellen stets um Datenmodelle handeln muss bzw. ob generische Modelle[29] stets auch Metamodelle sind, woraus sich weiterführend die Frage ableitet, ob Daten,

[27] Dessen uneinheitliche Verwendung zeigt sich am Beispiel der Extensible Markup Language (XML), in deren Umfeld häufig Begriffe wie *Meta-Spezifikation* oder auch *Meta-Sprache* verwendet werden, deutlich.

[28] Dies ist jedoch nicht zutreffend, da die Übersetzung des griechischen in *inmitten, zwischen, nach* und *hinter* die korrekte ist. Als Präfix wird es allgemein zur Bezeichnung des Wandels und des Wechsels verwendet (z. B. bei Metamorphose oder Metaphysik). Erst mit der Neubildung der modernen Wissenschaftssprache ab dem 18. Jh. wird Meta als Präfix auch produktiv verwendet, so auch im Begriff der Metakritik als Kritik an der Kritik (vgl. [Pfei00], S. 866).

[29] Modelle, die Daten zu deren Anpassung beinhalten

die andere Daten in Beziehung setzen, stets auf einer höheren Ebene im MOF liegen.[30] Die Ebenen stellen sich wie folgt dar (in Anlehnung an [Obje00a], S. 2-2f):

- **M0**: Die **Objektebene** „... is comprised of the information that we wish to describe." ([Obje00a], S. 2-2) Die Ebene enthält also Elemente, die mit der Realität korrespondieren. Eine Unterscheidung zwischen beiden wird jedoch vorgenommen. Die Probleme dieser einfachen Sichtweise wurden in Abschnitt 2.1 angedeutet.

- **M1**: Die **Modellebene** enthält Informationsmodelle (zur Herleitung des Begriffes siehe 2.4) der Objektebene M0, die unter Verwendung einer bestimmten Modellierungssprache erstellt wurden.

- **M2**: Die **Meta-Ebene** enthält Metamodelle. Die Aussage „... A meta-model can also be thought of as 'language' for describing different kinds of data" ([Obje00a], S. 2-5) wird dahingegen präzisiert, dass Metamodelle Informationsmodelle der in Informationsmodellen der Ebene M1 verwendeten Sprache sind.[31]

- **M3**: Die **Meta-Meta-Ebene** enthält Meta-Meta-Modelle (Meta-Metamodelle). Diese sind somit Informationsmodelle der in den Metamodellen der Ebene M2 verwendeten Sprache.

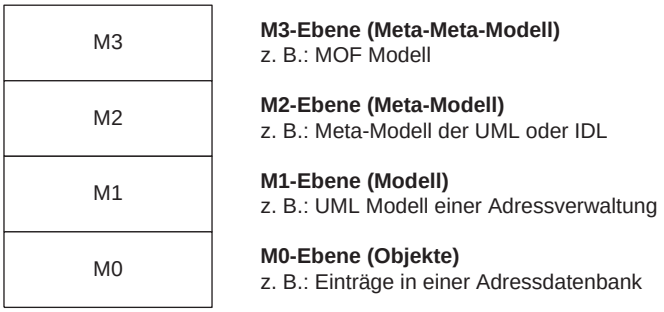



Abbildung 9: Lineares 4-Ebenen Modell der MOF (i. A. a. [Obje00a], S. 2–4)

Die Modellelemente einer Ebene werden in der darunter liegenden Ebene **instanziiert**. Jeder entstehenden Instanz muss sich ein Modellelement auf der übergeordneten Ebene zuordnen lassen und es wird als dessen Ausprägung und Exemplar (engl. *instance*) verstanden. Die Ebenen

[30] was eine interessante Perspektive auf die Relationship-Typen im Entity-Relationship Model (vgl. [Chen76], S. 9ff) wirft

[31] Dem sprachbasierten Meta-Ansatz wird stellvertretend von BRINKKEMPER ET AL. und ROSEMANN (vgl. [Bri+98] bzw. [Rose96]) entgegengesetzt, dass Metamodelle auch ein Aktivitätsmodell zum Umgang mit diesem enthalten. Dies widerspricht jedoch der Forderung nach harten Kriterien der Ebenenbildung und Anwendung nur eines Metaisierungsprinzips. Diese Arbeit folgt damit den Argumentationen von STRAHRINGER (vgl. [Stra96], S. 23) und FRANK (vgl. [Fran94], S. 171).

im MOF führen nach ATKINSON ET AL. zu einer **seichten Instanziierung** der Modellelemente. Diese basiert auf der Prämisse, dass eine Klasse nur die Semantik der direkten Instanzen festlegt. Werden von diesen in Architekturen mit multiplen Meta-Ebenen wiederum Instanzen gebildet, führt dies zu den unerwünschten Effekten der vieldeutigen Klassifikation und der wiederholten Definition von Konzepten (vgl. [AtKü01], S. 20ff). Auf dieses Problem wird im Rahmen dieser Arbeit im Abschnitt 6 eingegangen.

Es ist denkbar, oberhalb von M3 weitere Ebenen zu definieren. Inhaltlich werden diese jedoch der Ebene M3 entsprechen. Folgerichtig sprechen ÁLVAREZ ET AL. in diesem Fall von einer Schachtelung der Ebenen. Des Weiteren ist die Verankerung der Ebenen auf der Stufe M0 des MOF unzweckmäßig, da mit der Definition einer Sprache bereits deren Verwendung eingeschränkt wird (vgl. [Álv+01], S. 34ff). Von den Autoren wird stattdessen vorgeschlagen, die Ebene der Sprache zur Metamodellierung (M3) zu verankern. Als Bezugsrahmen für die Einordnung bei der *Anwendung* eines Metamodells ist eine MOF-konforme Bezeichnung sehr wohl geeignet. Die Abbildung 10 stellt das dieser Arbeit zugrunde liegende Verständnis der Meta-Ebenen zusammenfassend dar.

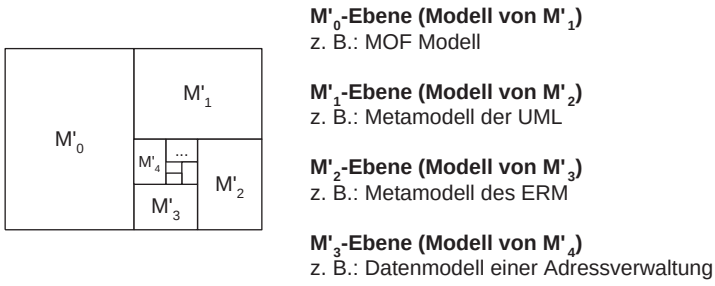



Abbildung 10: Geschachteltes Multi-Ebenen-Modell (vgl. [Álv+01], S. 38)

### 2.5.5 Produkt und Prozess

Entsprechend dem vorigen Abschnitt existieren bei der Erstellung von Informationsmodellen verschiedene Ebenen der Abstraktion. Aus diesen lässt sich nach Meinung von BRINKKEMPER im Gegenstandsbereich der Wirtschaftsinformatik eine Einteilung ihrer Disziplinen in das Method Engineering, die Systementwicklung und die Arbeiten auf der Gegenstandsebene ableiten. Jede Ebene definiert dabei die Rahmenbedingungen (in Form von Produkten und Prozessen) für die untergeordnete(n) Ebene(n). Hilfsmittel hierfür können wiederum Informationsmodelle sein, die i. w. S. eine Dokumentation der Spezifikationen auf der jeweiligen Ebene zulassen.

Jede Ebene definiert **Produkte** unter Verwendung von **Prozessen**. Produkte des ME sind z. B. Vorgehensmodelle (siehe Abschnitt 3.3) und Notationen für die Systementwicklung. Prozesse der Systementwicklung sind z. B. die Analyse und der Entwurf von Prozessen im Unternehmen

und die Implementierung von Anwendungssystemen. Produkte und Prozesse wiederum bestehen aus **Konzepten** und deren **Repräsentationen**, bzw. *semaionomen* und *semaion* im Sinne des Abschnitts 2.1.

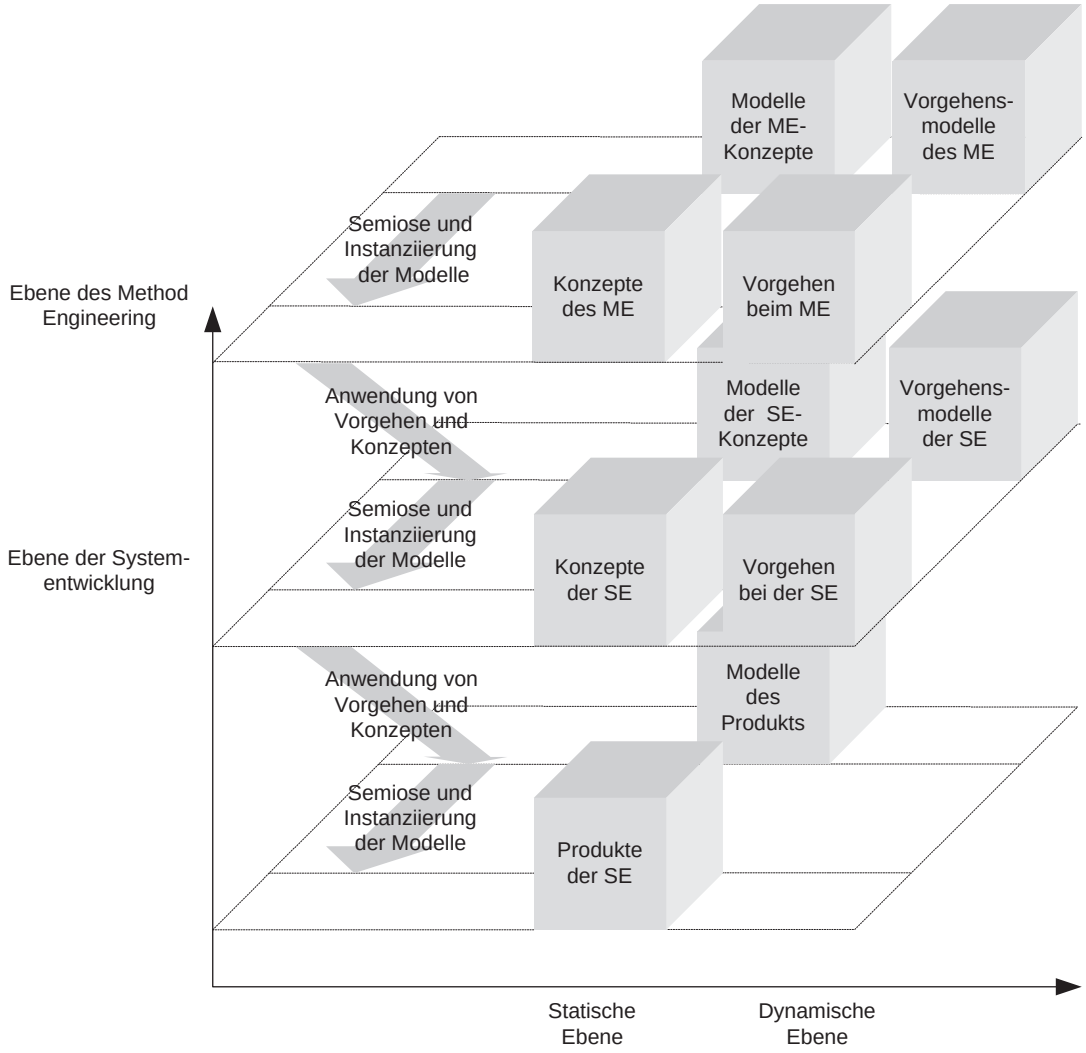



Abbildung 11: Ebenen der Informationsmodellierung

## 2.6 Zusammenfassung

Modelle spielen eine bedeutende Rolle bei der Analyse und Gestaltung von IS. Sie können selbst als Zeichen bzw. System von Zeichen verstanden werden, das die Interpretation natürlicher oder künstlicher Originale zulässt. Es werden nie alle Attribute des Originals modelliert, die entsprechende Auswahl obliegt dem Modellersteller, womit Modelle stets subjektiv durch eine Konstruktionsleistung entstehen. Zusätzlich besitzen Modelle stets einen Zeitbezug, der ihre Gültigkeit eingrenzt.

> **Definition 1:** Ein **Modell** ist ein System von Zeichen, das der verkürzten und zielgerichteten Darstellung natürlicher oder künstlicher Originale in einem bestimmten Zeitrahmen dient.

In der Wirtschaftsinformatik wird hauptsächlich an Informationsmodellen gearbeitet. Diese bilden Sichten auf ein betriebliches IS und besitzen stets einen messbaren Grad an Präzision, der sich aus der Zeichenklassifikation in formal, semiformal und nicht formal unterteilen lässt. Informationsmodelle können einen unterschiedlichen Abstraktionsgrad besitzen, je nachdem ob sie im Rahmen einer Systementwicklung oder z. B. bei der Definition eines Vorgehens für die Systementwicklung verwendet werden.

> **Definition 2:** Ein **Informationsmodell** dient der Planung und Darstellung betrieblicher Informationssysteme.

# 3 Erstellung von Modellen

Dieser Abschnitt widmet sich der Erstellung von Modellen und den Möglichkeiten, die dabei stattfindenden Vorgänge zu beschreiben. Er liefert damit auch die Anforderungen an eine Sprache, die eine Vorgehensbeschreibung erlaubt. Somit werden zwei wesentliche Voraussetzungen für den Teil II erarbeitet, welcher unter anderem eine Vorgehensbeschreibung bei der Konstruktion einer Methode enthält und ein Metamodell definiert, mit dem Vorgehensbeschreibungen bei der Modellierung erzeugt werden können.

Für die Erstellung von Modellen existieren zahlreiche Gründe, z. B. die strategische Planung, Unternehmensintegration, Entwicklung von Informationssystemen oder das Redesign von Geschäftsprozessen, um nur einige zu nennen. Motive lassen sich für den Einsatz auf dem jeweiligen Gebiet viele finden, der vorige Abschnitt hat einige von diesen aufgezeigt. PERSSON ET AL. haben in einer explorativen Studie untersucht, welche Ziele bei der Erstellung von Modellen in der Praxis verfolgt werden. Entstanden ist die Hierarchie der häufigsten Nennungen aus Abbildung 12, dabei sind vor allem die übergeordneten Ziele „Develop the business“ und „Ensure the quality of business operations“ genannt worden. In Summe wird damit die These der vielseitigen Verwendbarkeit der Modelle untermauert, die dabei eingesetzten Methoden dienen jedoch auch stets übergeordneten Zielen: „Modelling for the sake of modelling is not really useful. We have empirically found that method developers and researchers often forget this.“ ([PeSt01], S. 468)

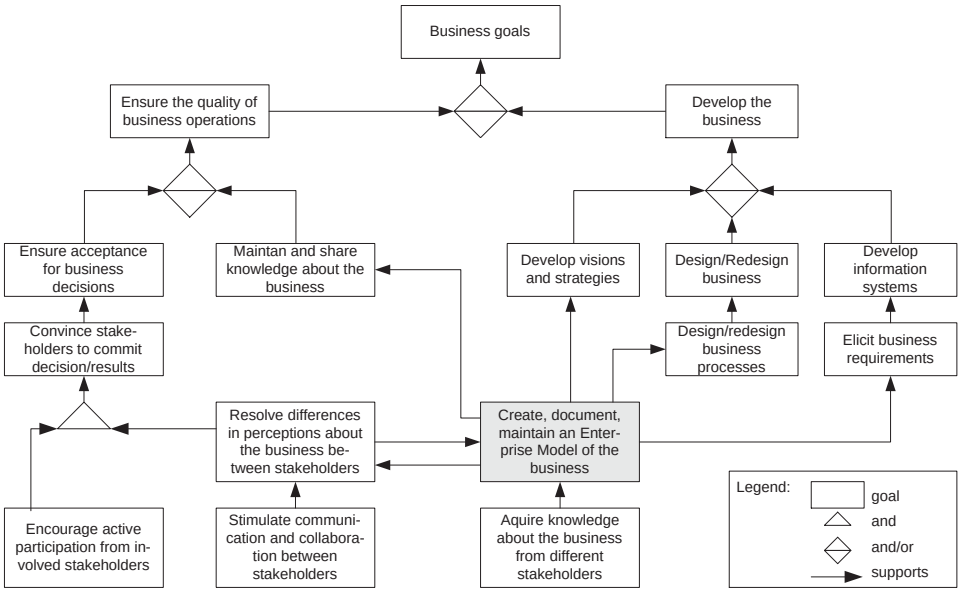



Abbildung 12: Zielhierarchie der häufigsten Intensionen für die Modellerstellung in Unternehmen ([PeSt01], S. 466)

Als ein wichtiges Ergebnis dieses Abschnitts wird im Kapitel 3.1 der Begriff der Methode

herausgearbeitet. Mit den Vorgehensmodellen des Kapitels 3.3 werden deren prinzipienordnende Handlungsanweisungen spezifiziert. Die der Systementwicklung zugrunde liegenden Prinzipien werden in Abschnitt 3.2 erläutert. Das Konfigurationsmanagement aus Abschnitt 3.4 verbindet Produkte und Prozesse unter besonderer Beachtung der Veränderungen am Produkt einer Methode. Nicht zuletzt die hohe Anzahl an Iterationszyklen in agilen Methoden aus Kapitel 3.5 motivieren dessen methodische Integration zusätzlich. Das Kapitel 3.6 beschreibt zum Abschluss anhand einer Studie von IVARI und HUISMAN den Zusammenhang zwischen Methoden und Unternehmenskultur.

## 3.1 Methoden

Mit den unterschiedlichen Möglichkeiten der Verwendung von Modellen existieren verschiedene Formen der Integration von Prozessen der Modellerstellung und -verwendung in die jeweilig übergeordneten Vorgänge. Für Software werden die beiden Hauptabschnitte Entwicklungs- und Nutzungszeit unterschieden (vgl. [StHa97], S. 252). KNUTH bezeichnet in diesem Zusammenhang die Modellierung als ein Hilfsmittel zum Abbau der Kommunikationsproblematik zwischen den Beteiligten an Projekten zur Softwareentwicklung. Ihre Rolle bei der frühzeitigen Fehlererkennung, Kostenminimierung und Dokumentation sind in aktuellen Veröffentlichungen der Wirtschaftsinformatik unbestritten (vgl. [Knut95], S. 34).

Informationsmodelle unterliegen einem kontinuierlichen Wandel ihres Gegenstandsbereiches. Ihre stets statische Abbildung von Abläufen und Strukturen ist nicht unproblematisch, da ihr Wert mit dem Verlust an Realitätsbezug stark abnimmt. Deshalb wird an dieser Stelle der Lebenszyklus von Modellen, also deren Zustand im Zeitbezug, näher beschrieben. Hierfür wird die Phase der **Modellerstellung** definiert und um die **Nutzungszeit** der Modelle ergänzt. Erstere umfasst dabei die im Abschnitt 2.3 beschriebene Konstruktionsleistung zur Erstellung von Modellen und deren Abgleich mit ihren Nutzern und letztere die oben beschriebene Anwendung des Modells als Kommunikationssystem. Für den Übergang stellt REMME fest, dass die Konstruktion abgeschlossen ist, wenn ein Modell den in der Analyse definierten Rahmenbedingungen entspricht und den Zielen des Unternehmens gerecht wird (vgl. [Remm97], S. 5).

Die Modellerstellung ist als Teil der Entwicklung von Anwendungssystemen ein komplexer Prozess, der sich nach Meinung von STAHLKNECHT „... nicht schon zu Projektbeginn als Ganzes planen läßt ...“ ([StHa97], S. 253). Dementsprechend existieren seit den 50er Jahren Konzepte, die zumindest das Vorgehen plan- und wiederholbar gestalten sollen. Dieses Vorgehen wird aus Sicht der Wirtschaftsinformatik insbesondere den Ingenieurdisziplinen bei der Konstruktion von technischen Systemen unterstellt (vgl. z. B. [FeSi01], S. 119), weshalb es als **ingenieurmäßiges Vorgehen** bezeichnet wird. KAZMEIER beschreibt dieses als Anwendung eines systematischen, disziplinierten, quantifizierbaren Ansatzes auf Strukturen, Maschinen,

Produkte, Systeme oder Prozesse. Dabei werden Wissen, Prinzipien, Techniken und Methoden eingesetzt, die der Erfahrung der Wissenschaft entstammen (vgl. [Kazm98], S. 37). Seit der Einführung des Begriffes **Software Engineering**[32] auf einer vom NATO Science Committee gesponserten Konferenz (siehe dazu [NaRa69]) wird seit Ende der 1960er Jahre versucht, das Gebiet der Softwareentwicklung als Ingenieursdisziplin zu etablieren (vgl. [Thal98], S. 33). Zuvor galt Softwareentwicklung als kreative und ungeordnete Tätigkeit, die Qualität der Resultate ließ sich administrativ weder planen noch steuern, sie wurde im Wesentlichen durch die Fähigkeiten der beteiligten Programmierer determiniert. Erfahrungen aus früheren Projekten ließen sich schwer auf deren Nachfolger übertragen. Vor diesem Hintergrund wurde seitdem versucht, die Prozesse der Systementwicklung entsprechend zu definieren.

Die Anwendung von Methoden bildet den Ausgangspunkt ingenieurmäßigen Vorgehens, in diesem Punkt ist man sich in der Literatur weitgehend einig. Leider existieren wie bei dem Begriff des Modells auch hier zahlreiche unterschiedliche Auffassungen über die Eigenschaften und Bestandteile von Methoden. Bei STAHLKNECHT ET AL. steht der Aspekt der Vorschrift, wie vorzugehen ist, im Vordergrund (vgl. [StHa97], S. 249). FERSTL und SINZ bezeichnen ihr Vorgehensmodell zum Semantischen Objektmodell (SOM) in Kombination mit einer Unternehmensarchitektur[33] als SOM-Methodik (vgl. [FeSi01], S. 180). Bei HENDERSON-SELLERS und BULTHUIS werden Methodologien beschrieben, die ein Framework zur Beschreibung technischer Details von Analyse, Design und Implementierung (vgl. [HeBu97], S. 5) bilden, um nur einige Beispiele zu nennen.

Allgemein beschreiben **Methoden** eine planmäßige Art und Weise des Handelns mit überprüfbaren Ergebnissen. Es wird also ein festgelegtes Regelsystem vorausgesetzt, welches zur Bewertung der Ergebnisse und der Aktivitäten herangezogen werden kann. Ziel von Methoden ist die Erlangung wissenschaftlicher Erkenntnisse oder praktischer Ergebnisse (vgl. [Wiss00], S. 867).[34] Der Begriff der **Methodik**[35] steht im allgemeinen Sprachgebrauch oftmals als Synonym für Methode.[36] Er dient aber alternativ auch als Bezeichnung für die Lehre bzw. Theorie wissenschaftlicher Methoden und wird zur Differenzierung in dieser Arbeit auch so verwendet.

[32] Im Deutschen wird dieser häufig mit *Softwaretechnologie* oder *Softwaretechnik* übersetzt. Software Engineering bezeichnet eine Wissenschaftsdisziplin, mit der eine „... zielorientierte Bereitstellung und systematische Verwendung von Prinzipien, Methoden und Werkzeugen für die arbeitsteilige, ingenieursmäßige Herstellung ... von umfangreichen Software-Systemen ...“ ([Balz96], S. 46) erfolgt. Ähnliche Definitionen finden sich bei SCHRADER und RUNDSHAGEN (vgl. [ScRu94], S. 2) oder WERNER (vgl. [Wern95], S. 388).

[33] Gemeint ist hier das Objektorientierte Modell der Unternehmung (vgl. [FeSi01], S. 37).

[34] Oftmals genügt in der Umgangssprache auch das Kriterium „Er hat sich etwas dabei gedacht“, um ein Vorgehen als methodisch einzustufen, wahrscheinlich aufgrund des griechischen Ursprungs im Wort $methodos$, welches das Nachforschen und Untersuchen bezeichnet (vgl. [Wiss00], S. 867).

[35] Das Wort stammt vom griechischen $methodika$, einer untergegangenen Schrift des Aristoteles (vgl. [Pfei00], S. 867).

[36] CRONHOLM ET AL. merken an, dass der schwedische Begriff „metodik“ für Methodentyp, als Konzept der Generalisierung von Methoden dient. Dies ist insofern von Bedeutung, da zahlreiche Veröffentlichungen zum Method Engineering aus diesem Sprachraum stammen.

**Methodologie** wird entweder synonym zu Methodik verwendet[37] oder beschreibt die Theorie der wissenschaftlichen Methoden (vgl. [Wiss00], S. 867).

CRONHOLM ET AL. liefern eine Zusammenfassung von Begriffen, die mit Methoden in Zusammenhang gesetzt werden. Im Wesentlichen sind dies (vgl. [CrÅg01], S. 3ff):

- **Methodenkette und -verbund:**[38] Methodenketten bestehen aus Methoden, die über die Phasen der Systementwicklung miteinander verknüpft sind. Die aus horizontalen Verknüpfungen bestehenden Methodenverbünde setzen sich aus Methoden der selben Entwicklungsphase zusammen.

- **Framework:** Frameworks werden als Ansatz zur Wiederverwendung von Software insbesondere in der Systementwicklung definiert und adressieren dort „ ... eine bestimmte Problemdomäne, für die sie dem Entwickler eine Infrastruktur für mögliche Anwendungen zur Verfügung stellen.“ ([Diet02], S. 99) Sie stellen in diesem Sinne eine Referenzarchitektur und -implementierung für eine Familie von Softwaresystemen zur Verfügung (vgl. [Diet02], S. 106). Der Begriff lässt sich problemlos in die Domäne des Methodenbegriffs übertragen,[39] und aus den Eigenschaften und Anforderungen an Frameworks lassen sich dann Konsequenzen für Methoden ableiten.

- **Methodenkomponenten und -fragmente:**[40] Mit dem Bestreben, Methoden nicht mehr als monolithische Systeme zu begreifen, um sie situationsbedingt anpassen zu können, wird ein Begriff für die Beschreibung von Methodenteilen benötigt. Hierfür wird von GOLDKUHL ET AL. das Konzept der *Methodenkomponente* (vgl. [Gol+97], S. 4) und als Synonym von HARMSEN *Methodenfragmente* (vgl. [Har+94], S. 169ff) definiert, welche allgemein als Bestandteile von Methoden auf unterschiedlichen Ebenen der Granularität[41] beschrieben werden.

Aus den bisherigen Betrachtungen des Abschnittes lassen sich für Methoden die allgemeinen Merkmale der Anleitung, der Zielorientierung und der Systematik ableiten.

### Anleitungsmerkmal

Methoden geben stets Anweisungen oder Hinweise des Vorgehens. Für Organisationen können daraus Verhaltensregeln für eine Problemdomäne abgeleitet werden. Aus dieser Forderung nach

[37] nicht nur, weil die englische Übersetzung von „Methodik“ „methodology“ ist
[38] engl.: Method chain and alliance
[39] indem man z. B. „Softwaresysteme“ durch „Vorgänge“ ersetzt
[40] engl.: Method components and method fragments
[41] Ebenen bei dieser Betrachtungsweise bilden: *method* (adressiert die vollständige Methode zur Entwicklung von IS), *stage* (adressiert eine Phase aus dem Lebenszyklus des IS), *model* (adressiert eine Perspektive auf das IS), *diagram* (adressiert die Repräsentation eines *models*) und *concept* (adressiert Konzepte, deren Beziehungen und Operationen auf Konzepte im *diagram*) (vgl. [Bri+98], S. 384f).

einer gewissen Allgemeingültigkeit folgt, dass Methoden nicht für nur ein Problem geschaffen werden und in der Anwendung entsprechend stets konkret ausgestaltet werden müssen. In diesem Sinne bieten Methoden eine systematische Anleitung (vgl. [Heym93], S. 14).

### Merkmal der Zielorientierung

Methoden beschreiben die zu ereichenden Ziele zumeist in Form von Ergebnissen von Prozessen oder Anforderungen an diese Ergebnisse. Diese Zielorientierung führt in Analogie zum pragmatischen Merkmal von Modellen dazu, dass Methoden nur in bestimmten Zeitintervallen und für bestimmte Subjekte Gültigkeit besitzen.

### Systematisches Merkmal

Wenn Methoden Anleitungen für die Erreichung bestimmter Ziele liefern sollen, müssen sie derart strukturiert sein, dass sich konkrete Aufgaben und Aufgabenträger für ein Problem ableiten lassen. Entsprechend werden ingenieurmäßige Methoden zumindest Phasen des Vorgehens oder den vollständigen Prozess der Erkenntnisgewinnung definieren. Die Produkte werden auf der Ebene von Konzepten und deren Repräsentationen beschrieben.

Aus diesen drei allgemeinen Merkmalen ergibt sich für den Gegenstandsbereich der Systementwicklung die folgende Definition:

> **Definition 3:** Eine **Methode** der Systementwicklung beschreibt die Prozesse im Lebenszyklus von Informationssystemen. Die entstehenden Produkte werden auf der Ebene von Konzepten und deren Repräsentationen definiert.

Eine *vollständige* Methode beschreibt *alle* Prozesse und Produkte, eine *unvollständige* Methode (oder ein Methodenfragment) Teilprozesse und Teilspezifikationen der Produkte und ihrer Repräsentationen. Die Beschreibung der Prozesse kann in Form von formalen oder semiformalen Informationsmodellen erfolgen, um die Genauigkeit und damit Verbindlichkeit der Beschreibung zu erhöhen, gleiches gilt für die Produktbeschreibung. STAHLKNECHT und HASENKAMP beschreiben bei einer ingenieurmäßigen Vorgehensweise eine enge Kopplung zw. Methoden (als prinzipienordnende Vorschriften) und Verfahren (als gezielte Anweisungen), was zu keiner begründbaren Trennung der Begriffe führt (vgl. [StHa97], S. 249f). Für die Prinzipien der Softwareentwicklung liefert BALZERT eine Zusammenfassung, die Gegenstand des folgenden Kapitels 3.2 ist. Die Vorgehensmodelle des Kapitels 3.3 bilden eine geeignete Spezifikationsform für die Handlungsanweisungen einer Methode. Modelle zur Produktspezifikation von Methoden wurden bereits mit dem sprachbasierten Metamodell aus Abschnitt 2.5.4 eingeführt. Das

Konfigurationsmanagement aus Abschnitt 3.4 verbindet Produkte und Prozesse unter besonderer Beachtung der Veränderungen am Produkt. Der resultierende Methodenbegriff und dessen Bezug zum Modellbegriff wird in Abbildung 13 dargestellt.

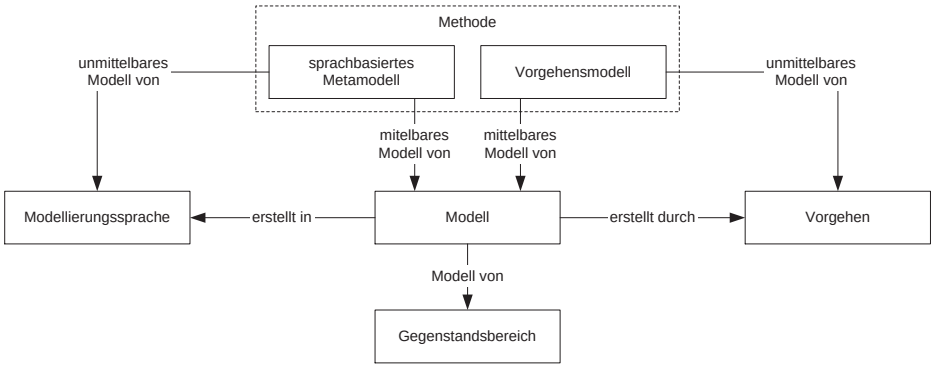



Abbildung 13: Methodenbegriff dieser Arbeit

## 3.2 Prinzipien

Im allgemeinen Sprachgebrauch beschreiben Prinzipien Grundsätze, Regeln oder auch Gesetzmäßigkeiten.[42] Für die Entwicklung von Software werden von BALZERT verschiedene Prinzipien zusammengefasst. Abbildung 14 stellt diese im Überblick dar.

Das *Prinzip der Abstraktion* beschreibt eine Verallgemeinerung mit dem Absehen vom Besonderen und Einzelnen (vgl. [Balz92], S. 27). Es ist in der Literatur das unumstritten jeder Modellbildung zugrunde liegende Prinzip und wird dementsprechend diskutiert. „Die Abstraktion erfüllt so ein menschliches Bedürfnis nach intellektueller Bemächtigung der Welt.“ ([Very92], S. 30ff) Eine zusätzliche Perspektive auf den Begriff eröffnet STACHOWIAK mit der Randbemerkung, dass die Abstraktion ein Variabilisieren von Konstanten ist (vgl. [Stac73], S. 130). Demnach werden bei der Abstraktion von Systemen Elemente nicht weggelassen, sondern durch variable bzw. in diesem Sinne abstrahierte Elemente ersetzt.

Das *Prinzip der Hierarchisierung* definiert für die Elemente eines Systems eine Rangordnung. Für Elemente gleichen Ranges wird eine Ebene der Hierarchie gebildet. Zur Bildung von Hierarchien sind folgende Eigenschaften zu definieren (vgl. [Balz92], S. 31ff):

- Basis der Hierarchie: Inhaltliche Relation der Sprachelemente (z. B. „A detailliert B“ oder „A benutzt B“)
- Struktur der Hierarchie: Eigenschaften der Relation bzw. der grafischen Repräsentation (z. B. netz- oder baumorientiert)

[42] Abgeleitet wurde das Prinzip im 18. Jh. aus dem lateinischen $principium$, das den Anfang, die erste Stelle und den Vorrang bezeichnet (vgl. [Pfei00], S. 1043).

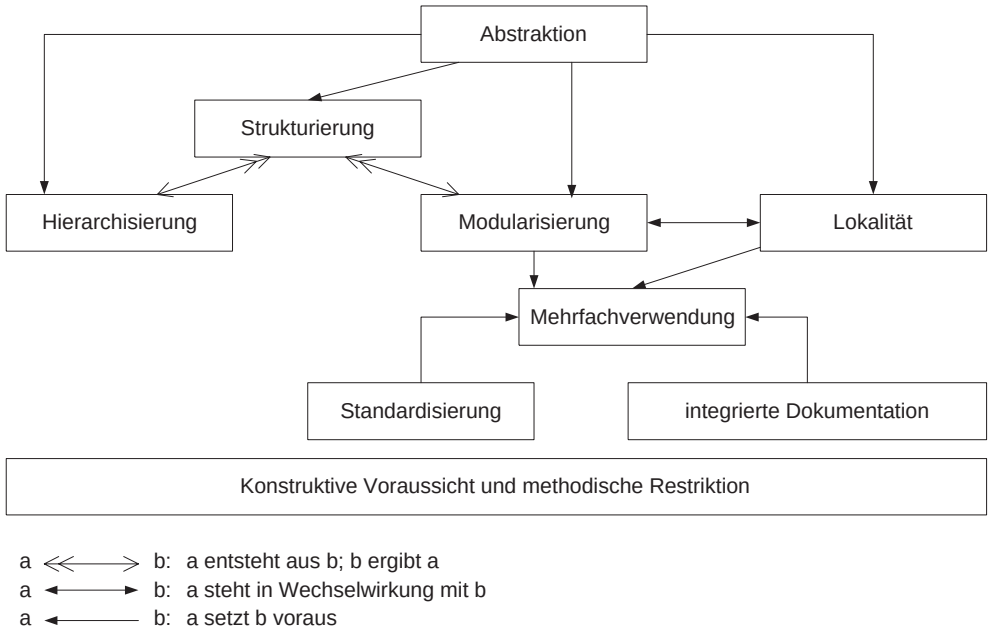



Abbildung 14: Fachsystematisches Netz der Prinzipien ([Balz92], S. 67)

- Zeit der Hierarchie: Zeitspanne der Existenz der Hierarchie (z. B. statisch oder dynamisch)

Von STRAHRINGER werden umfassend Modellhierarchien diskutiert, um den Begriff des Metamodells theoretisch zu untermauern (siehe dazu auch Abschnitt 2.5.4).[43] Basierend auf diesem Prinzip entsteht nach BALZERT durch die *Strukturierung* eine den Charakter des ganzen Systems offenbarende Darstellung. Ein gut strukturierter Entwurf entsteht durch Anwendung der Hierarchisierung und *Modularisierung*. Letzteres definiert Module mit festgelegten Schnittstellen und einer abgeschlossenen Funktionalität. STAHLKNECHT ET AL. sehen in Modulen in sich abgeschlossene Aufgaben, welche sich gegenseitig möglichst wenig beeinflussen[44] und damit die Grundlagen für das Geheimnisprinzip in der Objektorientierung bilden (vgl. [StHa97], S. 292).

Die Kombination aus dem *Prinzip der Lokalität*, als lokale Komprimierung von Informationen, und dem *Prinzip der integrierten Dokumentation*, als Forderung nach Dokumentation der Systemelemente, ermöglicht das *Prinzip der Mehrfachverwendung* von Systemelementen.

[43] Von besonderem Interesse sind hierbei die Untersuchungen zur Population (oder auch Besetzung mit Ausprägungen) einer Hierarchieebene.

[44] Dies bildet die Voraussetzung für die Unabhängigkeit vom Kontext des Moduls (vgl. [Balz92], S. 44).

## 3.3 Phasen- und Vorgehensmodelle

Aus dem Streben nach ingenieurmäßigem Vorgehen in der Softwareentwicklung entstanden zahlreiche Ansätze, den Entwicklungsprozess zu strukturieren. Die hierfür erstellten **Phasenmodelle** gliedern grob den Softwareentwicklungsprozess, indem Tätigkeiten zu Phasen zusammengefasst und deren zeitliche Aufeinanderfolge festlegt werden (vgl. [Wern95], S. 393). Im anglophonen Sprachraum bezeichnet man Phasenmodelle als **Life-Cycle Models**. Die DIN EN ISO 9000-3 definiert ein Phasenmodell als „... Modell, das Prozesse, Tätigkeiten und Aufgaben umfasst, die mit Entwicklung, Betrieb und Wartung eines Softwareprodukts verbunden sind, und das die Lebensdauer des Systems von der Festlegung der Forderungen bis zum Ende der Anwendung beschreibt." ([Deut98], S. 5) Abgeschlossen wird eine Phase durch Erreichen eines **Meilensteins**. Dabei handelt es sich um ein mit einer Zeitvorgabe versehenes Teilziel, an welchem sich der Projektfortschritt insgesamt messen lässt (vgl. [Balz98], S. 31). Ältere Phasenmodelle zeichnen sich durch strenge sequentielle Ordnung und klare Phaseneinteilung aus, wobei die Phasen mehr oder weniger den Softwarelebenszyklus charakterisieren (vgl. [ScRu94], S. 3 und [Leh$^+$91], S. 414). In realen Projekten ist es jedoch kaum möglich, die vorgegebene Reihenfolge und scharfe Trennung der Phasen einzuhalten. Aus diesem Grund wurden im Laufe der Zeit verschiedene Erweiterungen und neue Entwürfe vorgeschlagen, die mehr Flexibilität erlauben (vgl. [Wern95], S. 393).

Der bekannteste Vertreter der Phasenmodelle ist das **Wasserfallmodell nach ROYCE**, dessen ursprüngliche Version im Jahre 1970 vorgestellt wurde (vgl. [Royc70], S. 1f). „Es deckt den ganzen Lebenszyklus ab, beschreibt aber nur die Softwareentwicklung im engeren Sinn und beschränkt sich auf eine sehr grobe Gliederung des Prozesses." ([Chro92], S. 42). Dabei wurde zunächst mit Hilfe eines streng sequentiell vorgegebenen Prozesses versucht, die „... Softwareprojekte besser planbar, organisierbar und kontrollierbar ..." zu gestalten (vgl. [PoBl93], S. 17). ALTMANN und POMBERGER ([AlPo99], S. 650) gehen von einem hohen Anteil kreativer Tätigkeiten in der Softwareentwicklung aus, die eine Formalisierung bzw. Vorabbestimmung des Entwicklungsprozesses nahezu unmöglich machen. Somit konnten streng sequentielle Vorgehensmodelle nicht die Erwartungen erfüllen.

Dennoch weisen nach STAHLKNECHT und HASENKAMP die meisten modernen Vorgehensmodelle die Kernphasen des Wasserfallmodells auf: Analyse, Entwurf, Realisierung und Einführung (vgl. [StHa97], S. 253). Die Strukturierung durch Phasen ist demnach sinnvoll, für eine genaue Planung und Steuerung von komplexen Softwareprojekten ist sie jedoch zu grob. Deshalb schlägt CHROUST vor, ein Phasenmodell mehr als „... Organisationsprinzip und Maßstab des Projektfortschritts und weniger als Grundlage des wirklichen Vorgehens zu betrachten." ([Chro92], S. 111)

Das **Spiralmodell nach BOEHM** (vgl. [Boeh88], S. 61ff) gilt als Weiterentwicklung des Wasserfallmodells. Die vier Kernphasen Analyse, Design, Implementierung und Test werden

dabei in mehreren Iterationen durchlaufen (vgl. [ScRu94], S. 4). Das Spiralmodell bildet somit den Softwareentwicklungsprozess als evolutionären Prozess ab. Jeweils am Ende einer Iteration erfolgt eine Risikoabschätzung, anhand derer entschieden wird, ob und wie das Projekt weitergeführt werden soll. Die strenge zeitliche Reihenfolge des Wasserfallmodells suggeriert, dass Phasen grundsätzlich nach dem Kriterium der Zeit gebildet werden. Dass dies nicht sein muss, zeigt HESSE, indem er eine Phase als „... die zeitlich, begrifflich, technisch und/oder organisatorisch begründete Zusammenfassung von Tätigkeiten in einem Software-Projekt ...“ ([Hes+92], S. 29) definiert. Diese Verallgemeinerung, die sich nicht auf Softwareprojekte beschränken muss, gestattet beliebige Gruppierungen nach sachlogischen Gesichtspunkten. So könnte ein Projekt nach organisatorischen Gesichtspunkten unter anderem in die Phasen *Entwicklung*, *Qualitätssicherung* oder *Marketing* gegliedert werden. Eine solche Gliederung entspricht weitgehend einer funktionalen Organisation eines Projektes. Eine Gliederung in zeitliche Phasen hingegen kommt der traditionellen Strukturierung nach dem Wasserfall-Modell gleich. Dabei entstehen klassische Phasen wie beispielsweise *Analyse*, *Design* und *Implementierung*. Daneben können jedoch auch Tätigkeiten zusammengefasst werden, die auf gemeinsame technische Bereiche des Projektes bezogen sind. Denkbar wären hier Phasen wie beispielsweise *Datenbank*, *Middleware*, oder *Benutzerschnittstellen*.[45]

Bereits im Abschnitt 3.1 wurde auf die Aspekte einer ingenieurmäßigen Softwareentwicklung hingewiesen. **Vorgehensmodellen** wird als Teil der Spezifikation von Methoden dabei eine hohe Bedeutung beigemessen. Sie stellen die Umsetzung dessen dar, was BALZERT als „... systematische Verwendung von Prinzipien, Methoden und Werkzeugen ...“ bezeichnet. Somit bestimmen Vorgehensmodelle „... die wesentlichen Arbeitsetappen mit ihren Inhalten und deren zeitliche Reihenfolge.“ ([Wern95], S. 433) Es werden also Arbeitspakete gebildet und durch einen Ablaufplan zeitlich in Beziehung gesetzt.[46]

Ein systematisches und methodisches Vorgehen wird hauptsächlich in umfangreichen Projekten notwendig (vgl. [Balz96], S. 46). Gegenüber kleinen Projekten zeichnen sich diese dadurch aus, dass die Abhängigkeitsbeziehungen von Tätigkeiten zu zahlreich sind, als dass sie ohne geordnetes Vorgehen handhabbar wären. Mit Vorgehensmodellen werden Arbeitsabläufe strukturiert und wiederholbar gestaltet (vgl. [ScRu94], S. 2). CHROUST ergänzt, dass ein Vorgehensmodell „... eine der wesentlichen Grundlagen für einen arbeitsteiligen, einheitlichen,

[45] Wie allerdings eine Gruppierung nach begrifflichen Gesichtspunkten aussehen könnte, bleibt unklar, da HESSE in seinen Ausführungen auf Beispiele verzichtet.

[46] Mit der Entwicklung und Einführung der modernen objektorientierten Programmiersprachen wurden in Nachfolge zu den ersten Phasenmodellen neue Vorgehensmodelle notwendig (vgl. [NoSc99], S. 166), welche in der Mehrzahl ebenfalls einen iterativen Entwicklungsprozess verfolgen. Dabei wurde das Phasenkonzept teilweise durch aktivitätsbasierte Ansätze abgelöst (z. B. Rational Unified Process (RUP) vgl. auch [NoSc99], S.173f). Dabei stehen nicht die einzelnen Phasen in der Kritik, sondern deren streng sequentielle Anordnung. Deshalb wird in den neueren Ansätzen von einer unterschiedlich starken Intensität der Ausübung der einzelnen Phasen ausgegangen. Im Systemmodell von MOLIÈRE (vgl. [Moli84], S. 207ff) werden die zeitlichen Bezüge der einzelnen Tätigkeiten dagegen gar nicht betrachtet.

wiederholbaren Prozessablauf ... und damit eine Vorbedingung für die Industrialisierung ...“ ([Chro92], S. 37) ist. Es dient in der Softwareentwicklung der Unterteilung der Entwicklung „... in überschaubare Abschnitte und soll dadurch eine schrittweise Planung, Durchführung, Entscheidung und Kontrolle ermöglichen.“ ([PoBl93], S. 17)

Entsprechend dem Modellbegriff bilden Vorgehensmodelle wiederholbare Prozesse unabhängig von konkreten Projekten ab, indem Tätigkeiten vom konkreten Anwendungsfall abstrahiert werden. Nur in dieser verallgemeinerten Form lassen sich positive und negative Erfahrungen aus bereits durchgeführten Projekten in das laufende Projekt einbringen (vgl. [Chro92], S. 37). Damit beschreibt ein Vorgehensmodell „... in idealisierter Form, wie die Softwareentwicklung ablaufen soll, einen idealisierten Prozess“ ([Chro92], S. 39).

Zusammenfassend lässt sich festhalten, dass ein Vorgehensmodell **idealisierte**, **wiederholbare** und **strukturierte** Arbeitsabläufe eines Projektes beschreibt. Eine wesentliche Rolle spielt dabei die Wiederholbarkeit, die eine verallgemeinerte Tätigkeitsbeschreibung erfordert und einen Reifeprozess der Arbeitsabläufe ermöglicht. Vorgehensmodelle legen **Arbeitspakete und deren Reihenfolge** fest und bieten so eine Möglichkeit, ingenieurmäßiges Vorgehen in umfangreichen Projekten anzuwenden. Die zitierten Autoren fokussierten ihre Arbeiten auf das Gebiet der Softwareentwicklung. Vorgehensmodelle müssen sich jedoch nicht auf dieses Gebiet beschränken, sie sind in den verschiedensten Arten von Projekten anwendbar.

> **Definition 4:** Ein **Vorgehensmodell** beschreibt idealisierte, wiederholbare und strukturierte Arbeitsabläufe eines Projektes. Es legt Arbeitspakete und deren zeitliche Reihenfolge fest.

Der Einsatz von Vorgehensmodellen ist nur sinnvoll, wenn ein wirtschaftlicher Nutzen entsteht. Der Vorteil von Vorgehensmodellen erwächst in erster Linie aus der Anhebung des Qualitätsniveaus eines Projektes. Die Qualitätssteigerung kann sich einerseits auf das Produkt selbst beziehen, andererseits auf den Prozess, der zur Herstellung des Produktes führt. Der wirtschaftliche Nutzen eines Vorgehensmodells äußert sich also im Zuwachs der Erfolgswahrscheinlichkeit eines Projektes. Grundlegend für die Wirkung von Vorgehensmodellen auf die Prozessqualität (zum Begriff siehe Abschnitt 4.1) ist die Forderung nach **universeller Verwendung**, da nur so mit ein und demselben Vorgehensmodell verschiedene konkrete Prozesse beeinflusst werden können. Um den allgemeingültigen Charakter nicht zu verlieren, dürfen Vorgehensmodelle nur soweit fachspezifisch ausgeprägt sein, wie es der universellen Verwendung in ihrem Einsatzbereich nicht entgegensteht.

Günstige Wirkung auf die Prozessqualität eines Projektes entsteht zum ersten dadurch, dass sich Prozesse unter Anwendung eines Vorgehensmodells besser planen, steuern und kontrollieren lassen. Projekte ab einer gewissen Größe werden sogar erst durch Vorgehensmodelle handhabbar. Damit können Vorgehensmodelle unter Umständen zum kritischen Erfolgsfaktor eines

Projektes werden. Durch die Beschreibung eines idealisierten Handlungsablaufs bieten Vorgehensmodelle über den Steuerungsaspekt hinaus die Möglichkeit zur Prozessverbesserung. Kontinuierlich weiterentwickelte Vorgehensmodelle konservieren Erfahrungen aus bereits durchgeführten Projekten und dienen somit als Vorbild für konkrete Prozesse. Aus diesem Grund sind Vorgehensmodelle bei der Erlangung von Prozessreife und letztlich Prozesskultur in einem Unternehmen von zentraler Bedeutung.

Weiteres Potential zur Prozessverbesserung liegt in der Automatisierung des Vorgehensmodells. Sind Vorgehensmodelle in ausführbaren formalen Sprachen niedergelegt, können diese in geeignete Softwareumgebungen eingebettet und automatisiert werden. So können Mitarbeiter von bestimmten Routineaufgaben entlastet und bei anderen Tätigkeiten unterstützt werden. Grundlage der Automatisierung bildet jedoch eine Formalisierung des Modells, wie sie in Abschnitt 2.5.1 beschrieben wurde.

Vorgehensmodelle mit starren und bürokratischen Handlungsanweisungen können jedoch auch den Handlungsspielraum und damit die Kreativität der Mitarbeiter in Projekten zu sehr einengen.[47] Auch ein geringes Maß an Regulierung kann aufgrund des nicht unerheblichen organisatorischen Aufwands nur über den oben angeführten wirtschaftlichen Nutzen gerechtfertigt werden. Dies ist normalerweise nur in Projekten oberhalb einer gewissen Komplexitätsstufe und Größenordnung möglich.

## 3.4 Konfigurationsmanagement

Neben dem Einfluss auf die Entwicklung des Projektmanagements hatte der Einsatz von Versionierung bzw. **Konfigurationsmanagement (KM)**[48] in der Softwareentwicklung eine entscheidende Bedeutung für die Sicherstellung der Qualität der verwalteten Produkte. Ein ähnlicher Effekt wird für die Modellbildung vermutet. Da Methoden u. a. die Produkte des Modellbildungsprozesses spezifizieren, werden in diesem Abschnitt die Gründe für die Integration eines KM in die Methodenbeschreibung näher erläutert. Ausgangspunkt bildet dabei die Fragestellung, welche Gründe für Modellveränderungen vorliegen und wie diese das Modell beeinflussen. Aufbauend darauf wird aufgezeigt, welchen Beitrag ein geeignetes Modell-KM für die Verwaltung und Kontrolle von Modelländerungen leisten kann. Im Anschluss daran wird unter Verwendung eines Analogiemodells analysiert, ob sich die Vorteile aus dem Software-Konfigurationsmanagement (SKM) auch auf Modelle übertragen lassen.

Die dieser Arbeit zugrunde liegende Modelldefinition (vgl. Ausführungen Abschnitt 2.3) ist durch die Modellmerkmale nach STACHOWIAK (vgl. [Stac73], S. 132ff) und die Konstrukti-

[47] ADAMS nennt hierzu das Beispiel einer übermäßigen Beschäftigung mit der Selbstorganisation „Demand frequent status reports to explain why the team doesn't have enough time to meet deadlines“ ([Adam96], S. 233).
[48] zunächst im Bereich der Hochtechnologie und des Militärs (vgl. [Sayn98], S. 11f) eingesetzt

onsleistung des Modellerstellers gekennzeichnet. Dabei impliziert insbesondere das pragmatische Modellmerkmal die Notwendigkeit von Änderungen und deren Verfolgung, da Modelle stets Modelle „... für bestimmte ... Subjekte, innerhalb bestimmter Zeitintervalle ...“ sind ([Stac73], S. 132f). Somit stellt der dadurch ausgedrückte Subjektbezug von Modellen eine erste Motivation für die Verfolgung von Änderungen dar. Dieser setzt sich dabei aus dem Realitätsbezug und dem Problembezug zusammen (vgl. [Moli84], S. 90f). Daraus folgt, dass ein Modell nur dann als solches vom Modellanwender akzeptiert wird, wenn es einen Nutzen für die Lösung seines Problems offeriert. Dabei ist das Problemempfinden aufgrund der Abhängigkeit von Zielen des Modellanwenders stets subjektgebunden zu betrachten (vgl. [Moli84], S. 79). Wie ROSEMANN zeigt, üben Modellierungsziele einen erheblichen Einfluss auf die Modellbildung aus (vgl. [Rose96], S. 18f). Durch die Bildung von Zielen definiert das Subjekt eine gewünschte Soll-Situation und kann somit erst die Diskrepanz zur Ist-Situation erkennen, die nach DRESBACH ([Dres99], S. 76) Voraussetzung für das Vorliegen einer Problemsituation ist. Teil der Zieldefinition ist dabei eine Strukturierung der Soll-Situation, womit nach SCHÜTTE ein charakteristisches Merkmal des Modellbildungsprozesses erfüllt ist ([Schü98], S. 58). Somit kann die Zielbildung als die Erstellung eines internen Modells der Soll-Situation betrachtet werden. Nach MÜLLER-MERBACH sind für die interne Modellbildung die folgenden Determinanten von Bedeutung: Gesamtheit der Kenntnisse, Psyche und Wertesystem des Subjekts (vgl. [Müll80], S. 471ff). Dabei unterliegen die Determinanten der Gesamtheit der Kenntnisse und des Wertesystems einem stetigen Wandel, da nach POPPER Wissen nicht unveränderlich ist (vgl. [Popp94], S. XVIII).

Aufgrund der Betonung der Konstruktionsleistung des Modellerstellers beim hier verwendeten Modellbegriff (vgl. Ausführungen in Abschnitt 2.3) hat auch der Modellersteller einen erheblichen Einfluss auf die Erstellung und Entwicklung des Modells. Somit wirkt auch die Entwicklung seines Wissens auf den Modellbildungsprozess ein. Dies wird auch in die Ausführungen von DAHME und RAEITHEL deutlich, die explizierte Modelle als Mittel zur Transformation von implizitem Wissen in mitteilbares, öffentliches Wissen sehen (vgl. [DaRa97], S. 10). Daraus lässt sich schlussfolgern, dass ein Modell auch maßgeblich Ausdruck des Wissens des Modellerstellers ist und somit aufgrund der Wissensveränderung des Modellerstellers ebenfalls Änderungen unterliegt.

Ein Änderungsmanagement in Modellen unterstützt direkt bzw. indirekt alle Dimensionen der Modellbildung, wie sie bei DRESBACH als Frageschema: „... Modell wovon, für wen, wann und wozu.“ zusammengefasst werden ([Dres99], S. 79). Ein geeignet strukturiertes **Konfigurationsmanagementsystem (KMS)** stellt sicher, dass Beweggründe für getroffene Entscheidungen und deren Revision aufgrund neuer Erkenntnisse dokumentiert und somit nachvollziehbar werden. Des Weiteren können damit auch Änderungen an den Modellierungzielen und ihre Auswirkungen auf das Modell erfasst werden. Das KMS bildet somit einen wichtigen Grund-

stein im Management von Modellierungsprojekten. Zusätzlich sind auf Grundlage eines Basismodells verschiedene angepasste Modelle denkbar, deren Unterschiede durch die differierenden Zielstellungen motiviert sind. Als Beispiel für derartige Basismodelle können Referenzmodelle angeführt werden, die nach SCHÜTTE einen Bezugspunkt für die Entwicklung unternehmensspezifischer Modelle bilden und somit eine Klasse von Anwendungsfällen repräsentieren (vgl. [Schü98], S. 69).

Deren, insb. im Bereich der Wirtschaftsinformatik, zunehmende Bedeutung (vgl. [Schü98], S. VII und 1) eröffnet in diesem Zusammenhang weitere Anwendungspotentiale. Sie werden nicht für einen konkreten Nutzer erstellt, sondern für eine Klasse von Anwendungsfällen (vgl. [Schü98], S. 69). Die Ableitung individueller Modelle durch Anwendung von Build- bzw. Runtime-Operatoren (vgl. [Schü98], S. 244ff) bildet folglich Varianten des Referenzmodells. Ihre Aufzeichnung kann, in Verbindung mit den ebenfalls verwalteten Anpassungsprämissen, die zukünftige Entwicklung von Individualmodellen für vergleichbare Situationen erheblich beschleunigen. Andererseits erlaubt sie aber auch die Analyse von Gemeinsamkeiten der Varianten und unterstützt somit die Entwicklung branchenbezogener Referenzmodelle.

Die Tatsache, dass die Einführung des KM in der Softwareentwicklung in der Literatur übereinstimmend als ein wesentlicher Beitrag zur Verbesserung der Qualität von Softwaresystemen gesehen wird (vgl. z. B. [Estu00], S. 1), motiviert die Entwicklung eines KM auf dem Bereich der Modellbildung zusätzlich. PRESSMAN sieht SKM sogar als den Schlüssel zur Beendigung der in den 1960er Jahren deklarierten Softwarekrise (vgl. [Pres92], S. 693ff). Es ist nach Meinung von DART eine „... discipline for controlling the evolution of software systems.“ ([Dart90], S. 1)[49]

Aufgrund der Vorzüge und Leistungen eines wohlimplementierten SKM-Systems ist dieses mittlerweile zum unabdingbaren Bestandteil der Zertifizierung von Softwareunternehmen nach DIN ISO 9001 geworden (vgl. [Deut98], S. 22f in Verbindung mit [Deut96]). Auch Modelle für die Bewertung des Prozessreifegrades dieser Unternehmen beinhalten die Forderung nach einem KM als eine der grundlegenden Anforderungen. Beispielsweise ist der Einsatz eines SKM Voraussetzung für das Erreichen der zweiten Stufe im Capability Maturity Model (CMM) des Software Engineering Institute (SEI) (vgl. [Pau+93], S. 17f). Somit stellt die Implementierung eines KM für die Softwareentwicklung bereits heute einen wichtigen Qualitäts- und Wettbewerbsfaktor dar. Aufgrund des engen Zusammenhangs von Informationssystemen, insb. seines automatisierten und in Software definierten Teils des Anwendungssystems (vgl. [FeSi01], S. 3f), und der in dieser Arbeit betrachteten Informationsmodelle (vgl. [Schü98], S. 63) stellt sich die Frage, ob der Erfolg des SKM nicht auch ein KM für Modelle notwendig macht, da sowohl Software als Modell betrachtet werden kann als auch Modelle zur Erstellung von Soft-

[49] Eine ausführliche Diskussion des Begriffes erfolgt in Abschnitt 9.1.2.

ware verwendet werden.[50] Dies wird auch durch die Argumentation von DAHME und RAEITHEL gestützt, welche Software als von der virtuellen in die berechenbare Welt transformierte Modelle darstellen ([DaRa97], S. 9). Aufgrund der engen Beziehung zwischen Software und Modellen ist in der Modellerstellung mit denselben Problemen wie in der Softwareentwicklung zu rechnen, die zur Einführung von KM führten.

## 3.5 Agile Methoden

In den letzten Jahren treten mit den **agilen Methoden** neue Ansätze den Tendenzen der Überregulierung von Softwareentwicklungsprojekten entgegen (vgl. [Fowl02]). Traditionelle Vorgehensweisen und Praktiken aus der ingenieurmäßigen Softwareentwicklung werden aufgegriffen, kritisch hinterfragt und neu interpretiert. Mit agilen Methoden werden nicht große Teile des Softwareentwicklungsprojektes im Detail für einen langen Zeitraum geplant. Vielmehr steht die Resistenz des Softwareproduktes gegenüber Änderungen der Anforderungen im Vordergrund. Dies wird zumeist durch iterative Verfahren[51] realisiert und durch eine ständige Anpassung der Methode unterstützt. Ergänzend ist ein besonderer Schwerpunkt dieser Methoden die Motivation der Projektbeteiligten, welche vor allem durch eine geeignete Teambildung und Delegation von Verantwortung realisiert wird (vgl. [Mart02], S. 12f).

Agile Methoden und ingenieurmäßige Softwareentwicklung stehen nicht im Widerspruch zueinander. Dass die Diskussion über einen scheinbaren Gegensatz so ausdauernd und engagiert geführt wird, zeigt, welche Bedeutung der Frage von Arbeits- und Projektorganisation in Softwareprojekten beigemessen wird (vgl. z. B. im Forum zum Extreme Programming (XP) [Mart00]). Sicherlich dürfen die Vorzüge von definierten und damit reifenden Prozessen nicht gering geschätzt werden. Dennoch können Vorgehensmodelle zu einem nicht gerechtfertigtem Vertrauen an einen zum Erfolg führenden Automatismus verleiten, den ein Vorgehensmodell allein aber nicht bieten kann. Die Standardisierung der Softwareentwicklung dient nicht der Definition von starren Handlungsanweisungen, sondern vielmehr der ständigen Verbesserung der Prozesse in den Projekten, die ohne die Schaffung von Kriterien zu deren Bewertung nicht möglich ist (vgl. [Harm97], S. 9f).

FOWLERS Forderung nach möglichst individueller Auswahl und Anpassung einer Entwicklungsmethode an das jeweilige Team und Aufgabengebiet ist einer der Schwerpunkte des Si-

[50] JÄGER klassifiziert Modelle unter anderem nach deren Darstellungsmittel (vgl. [Jäge82], S. 144). Somit sind in formalen Sprachen spezifizierte Modelle in die Klasse der symbolischen (abstrakten) Modelle einzuordnen. Programmiersprachen sind gemäß den Ausführungen von KNUTH (vgl. [Knut95], S. 36f und S.39f) in den Bereich der künstlichen Sprachen einzuordnen. Ergänzend dazu sind Darlegungen von APPELRATH und LUDEWIG zu betrachten, die die künstlichen Sprachen in den Bereich der formalen Sprachen einordnen (vgl. [ApLu95], S. 22f). Eine weiterführende Analyse zeigt, dass Software auch die Modellmerkmale nach STACHOWIAK (vgl. [Stac73], S. 131ff) erfüllt.

[51] Die Phasen Analyse, Design, Implementierung und Testen werden über kurze Iterationszyklen und neuartige Arbeitstechniken enger miteinander verflochten (vgl. [Fowl02]).

tuational Method Engineering, wie es von HARMSEN bereits 1994 vorgestellt wurde (siehe [Har+94]). Die vielen bewusst kurz gehaltenen Iterationen während der Softwareentwicklung, die sich beim XP zeitlich in einer Größenordnung zwischen zwei und drei Kalenderwochen bewegen (vgl. [Jeff00] und [High00], S. 6f), stellen eine besondere Herausforderung für das zugrunde liegende Modell- und Softwarekonfigurationsmanagement da, zumal am Ende jeder Iteration eine lauffähige, getestete und mit dem Kunden abgestimmte neue Produktversion steht.

## 3.6 Methoden und Unternehmenskultur

Der Zusammenhang zwischen dem Einsatz von Methoden und der Unternehmenskultur wurde von IIVARI und HUISMAN untersucht. Dabei wurde nach der Befragung von Softwareentwicklern entsprechend dem *Competing Value Framework* ihr Unternehmen in die folgenden idealtypischen Dimensionen der Unternehmenskultur eingeordnet (vgl. [IiHu01], S. 236):

- **Group Culture (GC)**: Wandelbar, interner Fokus auf Flexibilität und menschliche Beziehungen; Zugehörigkeit, Vertrauen und Teamarbeit als wesentliche Werte
- **Development Culture (DC)**: Wandelbar, sehr zukunftsorientierter externer Fokus; Effektivität und Wachstum, Kreativität und Anpassung als wesentliche Werte
- **Hierarchical Culture (HC)**: Stabil, intern auf Sicherheit, Routine und Ordnung fokussiert; Kontrolle, Stabilität und Effizienz durch die Befolgung von Regeln
- **Rational Culture (RC):** Stabil, sehr leistungsorientierter externer Fokus; Produktivität, Effizienz und Zielerreichung als höchste Werte

Innerhalb der Studie wurde untersucht, ob eine Abhängigkeit zwischen der Wahrnehmung von Methoden der Systementwicklung und der Unternehmenskultur besteht. Entwickler und Manager wurden dazu getrennt befragt. Die Ergebnisse der Regressionsanalyse sind in Abbildung 15 dargestellt. Bemerkenswert ist vor allem die Tatsache, dass sich die Anwendung der Methoden in den unterschiedlichen Kulturen nicht unterscheidet. Das gleiche gilt für die Nutzung von Methoden als Instrument der Planung, Regelung und Priorisierung von Aktivitäten (Support as control technology). HC orientierte Organisationen sind nach Meinung der Entwickler eng mit der Methodenanwendung verbunden, sei es weil die Methoden als Teil der Regeln der Hierarchie verstanden werden oder weil die Hierarchien ein methodisches Vorgehen besonders gut unterstützen.

Die kritische Einstellung der Manager gegenüber den Methoden in besonders RC orientierten Organisationen mag verglichen mit den Entwicklern dadurch begründet sein, dass Manager die Ziele stärker fokussieren. Die Frage, ob die bestehenden Methoden der Systementwicklung gemessen an Produktivität und Effizienz unzureichend sind, kann die Studie nicht beantworten,

| | SDM use | Support as production technology | Support as control technology | Support as cognitive & coordination technology | Impact on quality of development systems | Impact on quality and productivity SD process |
|---|---|---|---|---|---|---|
| GC | | | | | | |
| DC | | De: ++ | | | | Ma: + |
| HC | De: + | De: +<br>Ma: ++ | De: + | De: + | | |
| RC | Ma: - | Ma: - | | Ma: - | | Ma: -- |

| | | | |
|---|---|---|---|
| SD | Software Development | De | Developer |
| SDM | Software Development Methodology | Ma | Manager |

Abbildung 15: Beziehung zwischen der Organisationskultur und der Anwendung von Methoden (vgl. [IiHu01], S. 247)

zumal der Beitrag der Methoden zu diesen Kriterien schwer zu messen ist. Daneben ist bekannt, dass die vorhandenen Methoden meist keine Hilfestellung zur Lösung von Krisensituationen in Softwareprojekten bieten - ein Punkt, der bei den Managern dieser Projekte zu einer starken Unzufriedenheit führt. Die Konsequenzen aus dieser Studie sind für die Anwendung von Methoden vor allem folgende (vgl. [IiHu01], S. 248):

- Die Chance, dass eine Methode akzeptiert wird, ist in HC orientierten Organisationen am höchsten.
- In DC orientierten Unternehmen müssen die Methoden insbesondere die Kreativität und Anpassung an das Umfeld des Unternehmens ermöglichen.
- Die Schaffung von Ordnung und Routine bei wenig kreativen Aktivitäten der Systementwicklung schafft für die Entwickler freie Zeit für Kreativität.
- Methoden können Hierarchien in Organisationen manifestieren. Wenn ein Unternehmen sich nicht in eine HC orientierte Richtung entwickeln möchte, muss bei der Einführung von Methoden ein besonderer Wert auf die Vermeidung hierarchischer Vorgaben und bürokratischer Anweisungen gelegt werden. Methoden können hier als generelle Ansätze und weniger als Sammlung vollständig dokumentierter Techniken eingesetzt werden.

## 3.7 Zusammenfassung

Im Abschnitt 3 wurde dargestellt, wie Methoden den Prozess der Modellbildung und Systementwicklung unterstützen. Deren Merkmale der Anleitung, Zielorientierung und Systematik stehen im engen Zusammenhang mit der Art ihrer Spezifikation. Phasenmodelle gliedern die

Prozesse durch Zusammenfassung von Tätigkeiten und Vorgehensmodelle bilden insbesondere deren zeitliche Reihenfolge ab. Durch die Spezifikation idealisierter, strukturierter und wiederholbarer Arbeitspakete wird ein ingenieurmäßiges Vorgehen bei der Verwendung von Methoden möglich. Ob Vorgehensmodelle Handlungsempfehlungen oder Anweisungen darstellen, ergibt sich aus dem Kontext der Methode und führt insbesondere für agile Methoden zu einem Instrument lernender Organisationen (vgl. [Fowl02]) bzw. Ausdruck einer bestimmten Unternehmenskultur.

Wichtige Bestandteile moderner Methoden sind die in Vorgehensmodellen beschriebenen Rückkopplungen und Iterationen (vgl. [Balz98], S. 98ff). Ein Instrument zur Beherrschung der dabei entstehenden Produktversionen und -varianten stellt ein Konfigurationsmanagement dar, welches entsprechend der Methode zugrunde zu legen ist. Ähnlich wie Software unterliegen auch Modelle in ihrem Lebenszyklus einer mehrfachen Überarbeitung. Dabei ist es notwendig, die neu getroffenen bzw. revidierten Entscheidungen in der Modellbildung aufzuzeichnen und zu kontrollieren. Die Erfassung der Modifikationen und ihrer Ursachen ermöglicht eine rückwirkende Analyse derselben und kann somit Informationen für Entscheidungen in zukünftigen Entwicklungen liefern. Der Softwareentwicklungsprozess wurde in der Vergangenheit entschieden durch die Integration von Verfahren zur Aufzeichnung und Kontrolle von Veränderungen im Rahmen eines Konfigurationsmanagements geprägt. Die Schaffung eines Modell-KMS wird einen ähnlichen Einfluss auf den Modellbildungsprozess ausüben können.

# Teil II
# Method Engineering

Mit diesem Teil der Arbeit werden durch die Beschreibung der Qualitätsanforderungen an Methoden wesentliche Rahmenbedingungen zu deren Bewertung und Konstruktion definiert. In der Wirtschaftsinformatik existieren zahlreiche Veröffentlichungen, die sich mit dem Begriff der Qualität, deren Messung und Gewährleistung auseinandersetzen. Angewendet auf Methoden werden zahlreiche Vorschläge unterbreitet, deren Ergebnis- und Prozessqualität zu bestimmen und zu verbessern. Das folgende Kapitel 4 wird diese ordnen und mit dem *Quality Function Deployment* ein Instrument der Methodenauswahl, -bewertung und damit letztlich auch -konstruktion vorstellen. Mit dem Kapitel 5 wird ein Überblick über den Stand der Forschung auf dem Gebiet des Method Engineering gegeben. Im Sinne der Zielstellung dieser Dissertation werden ausgewählte Ansätze durch den Autor dergestalt subjektiv beurteilt, dass die angelegten Kriterien offen, nachvollziehbar und ggf. durch den Leser anpassbar vorgestellt werden.

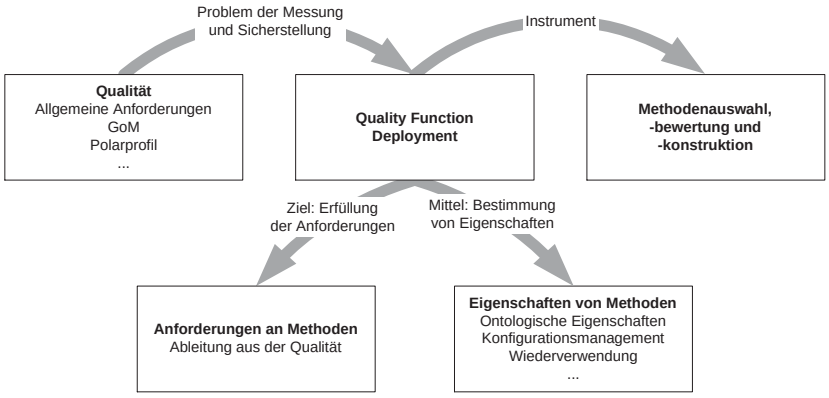



Abbildung 16: Gedankengang in Teil II

# 4 Bewertung von Methoden

In den letzten Abschnitten wurden das Vorgehen bei der Modellerstellung und die unterschiedlichen Modellarten innerhalb der Systementwicklung diskutiert. Dabei wurde bereits die Bedeutung der verwendeten Methoden für die Ergebnisqualität angedeutet. Dieser Abschnitt wird ein Instrumentarium für eine allgemeine objektivierte Beurteilung von Methoden liefern. Die Ergebnisse werden im Abschnitt 5 eingesetzt, um die wesentlichen Werkzeuge des Method Engineering[52] vorhandener Ansätze zu bewerten. Die Bedürfnisse der Methodenanwender stehen dabei im Mittelpunkt des Abschnittes. Besonderer Wert wurde auf die praktische Anwendbarkeit der Qualitätsbestimmung gelegt, nur so sind die Ergebnisse für den Einsatz im Method Engineering zur Qualitätssicherung der entstehenden Methoden und zur Bewertung der Methoden des ME selbst einsetzbar.

In einer ersten Annäherung ist es unerlässlich, den Begriff der Qualität näher zu untersuchen. Nach diesen Ausführungen in Abschnitt 4.1 wird mit dem Quality Function Deployment in Abschnitt 4.2 ein Instrument vorgestellt, welches die oben skizzierten Anforderungen erfüllt. Dieser Ansatz basiert auf der getrennten Erfassung von Anforderungen und Merkmalen des jeweiligen Gegenstandes der Qualitätssicherung. Erstere werden durch die Abschnitte 4.3 bis 4.7 erarbeitet und in Abschnitt 4.8 zusammengefasst. Durch die Ausgestaltung der Merkmale für das Gebiet des ME kann dieses Verfahren auf ausgewählte Methoden im Abschnitt 5 angewendet werden.

## 4.1 Qualität und Maßnahmen zu deren Sicherung

Seit in den 1970er Jahren in Deutschland Konsumentenbedürfnisse nach besonderen industriellen Konsumgütern auftraten, die von einheimischen Produzenten nicht mehr erfüllt werden konnten, wurde intensiv über Qualitätsanforderungen von Produkten nachgedacht. Die deutsche Phono- und Photoindustrie musste sich einem zunehmenden Konkurrenzdruck aus fernöstlichen Ländern stellen und ihre Rolle als Aushängeschild deutscher Technik war beendet (vgl. [Oess93], S. 18). Das wachsende Qualitätsbewusstsein führte zu Begriffen wie dem der „Qualität der Arbeit“ oder „Qualität des Wettbewerbs“ und damit auch in den Unternehmen zu Ansätzen, diese zu gestalten.

Die Kundenorientierung[53] kann nach HERZWURM zum Element eines Wertesystems im Unternehmen und damit zur Unternehmenskultur werden oder als Strategie u. a. das bestimmende Element konstruktiver Qualitätssicherungsmaßnahmen bilden. Im Folgenden wird die Kundenorientierung als Entscheidungsgrundlage bei der Auswahl von Handlungsalternativen herange-

[52] insbesondere die Vorgehensmodelle und Metamodelle

[53] als Ausrichtung einer Leistungserstellung am Abnehmer der Leistung (vgl. [Wiss00], S. 962 bzw. [Pfei00], S. 744)

zogen und in diesem Sinne als Strategie verstanden (vgl. [Herz00], S. 18).

Der Begriff der Qualität ist vieldeutig und hängt nicht nur von den Perspektiven (z. B. die des Verbrauchers oder des Herstellers) ab (vgl. [Oess93], S. 31). Aus diesem Grund wurde für den Begriff der Qualität eine deutsche Industrienorm geschaffen. Qualität ist demnach „... die Gesamtheit von Merkmalen (und Merkmalsausprägungen) einer Einheit bezüglich ihrer Eignung, festgelegte und vorausgesetzte Erfordernisse zu erfüllen." ([Deut94], S. 33)

GARVIN beschreibt fünf Ansätze zur Definition von Qualität (vgl. [Garv84], S. 25ff). Der *transzendente* Ansatz basiert auf der Annahme, dass Qualität universell erkennbar und ein Synonym für kompromisslos hohe Standards und Ansprüche an die Funktionsweise eines Produktes ist. Demnach kann Qualität nicht präzise definiert und gemessen, sondern nur durch Erfahrung bewertet werden. Somit bietet dieser Begriff wenig praktische Hilfe bei der Sicherstellung von Qualität für ein Produkt (vgl. [Oess93], S. 32). Nach dem *produktbezogenen* Ansatz ist Qualität präzise messbar. Dabei auftretende Abweichungen spiegeln Unterschiede in der vorhandenen, beobachtbaren Quantität bestimmter Eigenschaftsausprägungen wider. Diese Betrachtungsweise erlaubt es, Rangordnungen von verschiedenen Produkten gleicher Kategorien anzugeben. *Anwenderbezogen* wird die Qualität durch den Benutzer und weniger durch das Produkt selbst festgelegt. Verschiedene Benutzer haben unterschiedliche Wünsche und Bedürfnisse. Produkte, die diese am besten befriedigen, werden als qualitativ hoch stehend angesehen. Das größte Problem bei diesem Ansatz ist die häufige Gleichsetzung einer optimalen Bedürfnisbefriedigung mit Qualität. „Nicht jedes Produkt, das häufig gekauft wird, ist deswegen besser." ([Oess93], S. 32) Der *prozess-* oder *fertigungsbezogene* Ansatz setzt Qualität mit der Einhaltung von Spezifikationen gleich. Er basiert auf der Idealvorstellung, dass eine Tätigkeit beim ersten Mal richtig ausgeführt wird. Er ist auf die heutige Wirtschaft und Industrie ausgerichtet und spielt eine bedeutende Rolle bei der Automatisierung, um den Produktionsprozess bezüglich geringerer Ausschuss- und Nachbearbeitungskosten zu optimieren. Problematisch ist hierbei, dass für einen Kunden ein den Anforderungen des Herstellers genügendes Produkt noch lange kein Qualitätsprodukt sein muss (vgl. [Oess93], S. 33). Unter wirtschaftlichen Gesichtspunkten im *Preis-Nutzen-bezogenen* Ansatz ist ein Qualitätsprodukt ein Erzeugnis, dass einen bestimmten Nutzen zu einem akzeptablen Preis oder eine Übereinstimmung mit Spezifikationen zu akzeptablen Kosten erbringt.

Maßnahmen zur Sicherung von Qualität können in folgende Kategorien eingeteilt werden (vgl. [Wall90], S. 24ff):

- planerisch-administrative: Maßnahmen für Aufbau, Einführung und Pflege eines Qualitätssicherungssystems auf projektübergreifender, projektinterner oder phasenbezogener Ebene

- konstruktive: Maßnahmen zur präventiven Qualitätsgestaltung zur Verhinderungen von Fehlern und Qualitätsmängeln
  - technische: z. B. Verwendung von Prinzipien, Methoden und Werkzeugen des Software-Engineering
  - organisatorische: z. B. Verwendung von Vorgehensmodellen, Pläne zur Konfigurationsverwaltung
  - menschliche: z. B. Schulungen, psychologisch orientierte Maßnahmen
- analytische: Maßnahmen zum Erkennen und Lokalisieren von Fehlern und Mängeln, also Maßnahmen zur Bewertung der Qualität
- psychologisch orientierte: Maßnahmen für die Menschen als Entwickler, Projektleiter und Projektmanager mit Fokus auf die Fähigkeiten der Teammitglieder[54]

Eine ähnliche Einteilung erfolgt bei HERZWURM zur Klassifizierung der Instrumente der Produktenwicklung.[55] *Konstruktive Instrumente* dienen dabei der Informationsgewinnung und Generierung von Produkten. *Analytische Instrumente* unterstützen die Entscheidungsfindung bei alternativen Produkten (vgl. [Herz00], S. 46ff) in Form von auswertenden und prüfenden Instrumenten. In die erste Kategorie ordnen sich u. a. die Werkzeuge des Requirements Engineering[56] ein. Mit Bezug auf die Softwareentwicklung werden hierunter im engeren Sinne alle Aktivitäten zu Beginn eines Softwareprojektes zusammengefasst, die auf eine Präzisierung der Problemstellungen abzielen (vgl. [Part91], S. 25).

Eine Untersuchung der Techniken zur Produktentwicklung aus Abbildung 17 wurde von HERZWURM (vgl. [Herz00], S. 200ff) durchgeführt. Dabei wurde festgestellt, dass die Instrumente des Requirements Engineering sich zur Transformation natürlichsprachlicher Benutzeranforderungen in formale oder semiformale Spezifikationen eignen, es wird jedoch weder zwischen Kunde und Benutzer getrennt,[57] noch wird die Wichtigkeit festgestellter Anforderungen und Informationen über die Konkurrenz erfasst. Die Ergebnisse des Requirements Engineering sind zudem nur mit erheblichen Problemen kommunizierbar, nicht zuletzt weil sie sich an technischen Aspekten der Informationssysteme orientieren. Die anderen Instrumente

[54] WALLMÜLLER diagnostiziert eine stark technische Überbetonung des Prozesses der Softwareentwicklung und fordert eine stärkere Akzeptanz nicht technischer Wissenschaftsgebiete in der Informatik (vgl. [Wall90], S. 25f).

[55] Unter den Instrumenten werden Modelle, Methoden, Verfahren und technische Hilfsmittel zur Erreichung von Produktenwicklungszielen zusammengefasst (vgl. [Herz00], S. 45).

[56] Wissenschaft, die sich mit der Erfassung, Beschreibung und Überprüfung von Anforderungen an informationsverarbeitende Systeme auseinandersetzt. Ziel ist die Erarbeitung methodischer Konzepte und deren Umsetzung in ingenieurmäßigen Arbeitstechniken (vgl. [Kazm98], S. 41f).

[57] Dies ist insofern von Bedeutung, da so nur unmittelbare Forderungen der Benutzer erfasst werden und mittelbare Forderungen anderer Interessengruppen, die zwar *Kunde* jedoch nicht *Benutzer* sind, außer Acht gelassen werden.

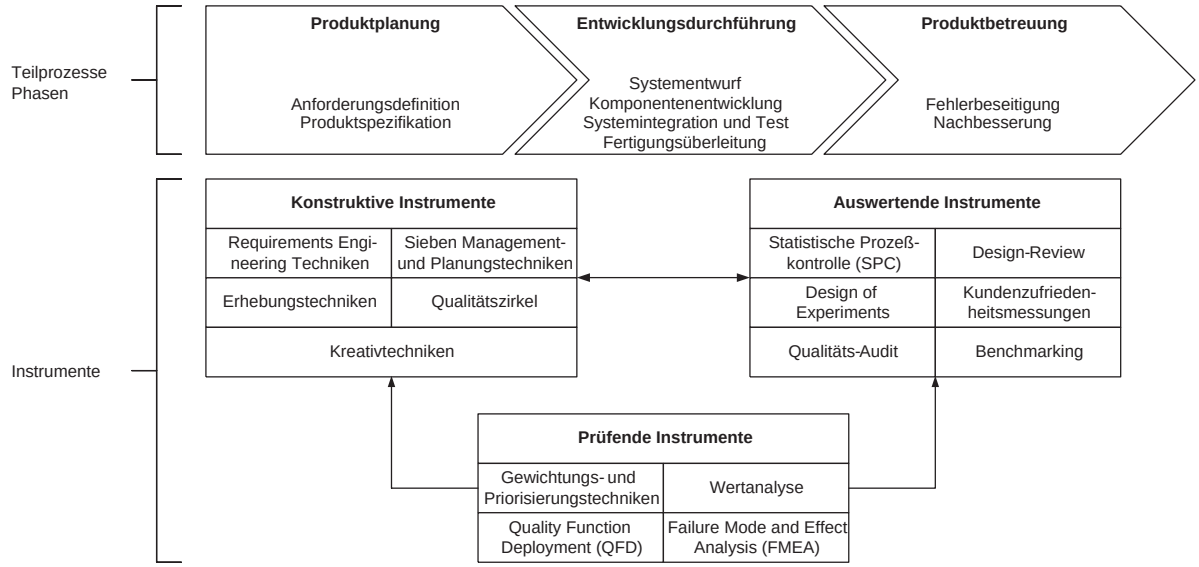



Abbildung 17: Instrumente der Produktentwicklung ([Herz00], S. 51)

der Produktentwicklung decken nur Teilaufgaben ab und sind zudem nicht zur Entwicklung von Informationssystemen geeignet. Einen Überblick über die Bewertung liefert der Anhang 13.3.1.

Die Erfassung und Überprüfung von Anforderungen erweist sich sowohl aus philosophischer als auch sozialer Perspektive als problematisch. Mit dem konstruktivistischen Modellbegriff wurde bereits die Möglichkeit einer objektiven Wahrnehmung der Welt ausgeschlossen (siehe Abschnitt 2.3). Entsprechendes muss auch für die Erfassung von Anforderungen und daraus abgeleiteten Qualitätsmerkmalen gelten, zumal aus sozialer Sicht u. a. unterschiedliche Bildungsniveaus, Aufgabengebiete und Machtverhältnisse innerhalb einer Organisation widerspruchsfreie Anforderungen an die Qualität nahezu ausschließen und zudem einem ständigen Wandel unterwerfen.

## 4.2 Quality Function Deployment

Bezüglich der Effizienzkriterien aus Anhang 13.3.1 ist das **Quality Function Deployment (QFD)** am besten zu bewerten. Beim QFD dominiert „… die Zielsetzung einer hohen Kundenzufriedenheit, die über den zusätzlichen Schritt einer nicht formalen Lösungsbeschreibung und die kohärente, kundenfokussierte Ausrichtung des Entwicklungsprozesses erreicht werden soll.“ ([Herz00], S. 203) Das QFD hat somit die Erfassung und Gewichtung von Kundenbedürfnissen zum Ziel (vgl. [Zink95], S. 284ff). In diesem Verfahren liegt ein erhebliches Potential für die methodische Unterstützung der Kundenorientierung in einem Unternehmen (vgl. [Malo99], S. 59f). Aus dieser Tatsache wird die Verwendbarkeit bei der Bewertung von Produkten abge-

leitet, wie sie in Abschnitt 2.5.5 vorgestellt wurden.

Das von AKAO Mitte der 1960er Jahre für Schiffswerften entwickelte QFD (vgl. [Akao92]) wird in Deutschland insbesondere durch das QFD Institut Deutschland e. V. gefördert (siehe dazu [Herz01]). Es dient der Umsetzung von Kundenbedürfnissen in Produkt- und Prozessanforderungen. Im Zentrum des QFD steht das *House of Quality* (HoQ) (vgl. [Schö94], S. 1ff). Das HoQ ermöglicht in einer Priorisierungsmatrix[58] die Strukturierung und Objektivierung von möglicherweise nicht quantifizierbaren Kundenanforderungen, indem diesen einzelne in der Fachsprache der Produktentwickler definierte Messgrößen zugeordnet werden. Aus Anforderungen des Marktes werden im HoQ somit schrittweise Parameter von Produkten und Dienstleistungen des Unternehmens abgeleitet, auf die ein unmittelbarer Einfluss möglich und messbar ist.

Die Abschnitte 4.3 bis 4.7 werden den Nachweis erbringen, dass nicht nur sehr unterschiedliche Qualitätskriterien existieren, sondern zusätzlich eine Subjektivität der Bewertung durch unterschiedliche Rollen im Umgang mit den Methoden beachtet werden muss. Festzustellen ist weiterhin, dass im Kontext des Methoden Engineering im Vergleich sehr wenige Qualitätskriterien existieren, die für die *Ersteller* von Methoden von Interesse sind. Dies birgt die Gefahr in sich, dass aus eigenem Antrieb nur qualitativ minderwertige Methoden definiert werden. Mittelbar führt dies jedoch zu Nachteilen in der Anwendung. Das QFD verspricht bei konsequenter Anwendung Abhilfe zu verschaffen, indem durch den Kunden bewertete Produktcharakteristika ermittelt werden können. Weitere Vorteile der Anwendung des QFD liegen vor allem in folgenden Punkten (vgl. [Her+00], S. 7ff):

- Effiziente Systematik für die Kommunikation über Anforderungen und der Lösungen, ohne zusätzlich Schritte zu fordern, die über das „normale“ Requirements Engineering hinausgehen
- Kundenorientierung bei der Erfassung und Evaluierung von Anforderungen mit der Möglichkeit des Vergleichs mit Wettbewerbern aus Kunden- und Entwicklersicht
- Zielorientiertes Requirements Engineering schafft Transparenz bei der Entscheidungsfindung, welche die Planung und Weiterentwicklung des Produkts erleichtert

Die einfache Anwendbarkeit führte zu einem weltweiten Einsatz des QFD auf unterschiedlichen Gebieten, wie beispielsweise in der Software- oder Automobilindustrie (vgl. [Her+00], S. 12). Wenn mit Hilfe des QFD im Rahmen dieser Arbeit eine Methode für die Entwicklung von Methoden abgeleitet wird, ist ein erster wichtiger Schritt die Identifikation von Kunden. Es

[58] Das im HoQ verwendete Matrix-Diagramm wird in Anhang 13.3.2 dargestellt. Es ermöglicht die Analyse der Beziehungen zwischen zwei Arten von Information und eine Quantifizierung der Größe der Korrelation zwischen beiden. Im Ergebnis ist eine Priorisierung der einen Information durch Gewichtung der anderen Information möglich (vgl. [Her+00], S. 22).

erscheint in Ermangelung eines offensichtlichen *Marktes* für Methoden sinnvoll, unter dem Begriff des Kunden vor allem die Benutzer einer Methode und deren organisatorischen Kontext, z. B. Projektleiter, zu verstehen. Indirekt werden damit auch bewusst die Bedürfnisse Dritter erfasst, die ein Interesse an den Produkten der Methode besitzen, also z. B. Kunden einer mit der Methode entwickelten Software.

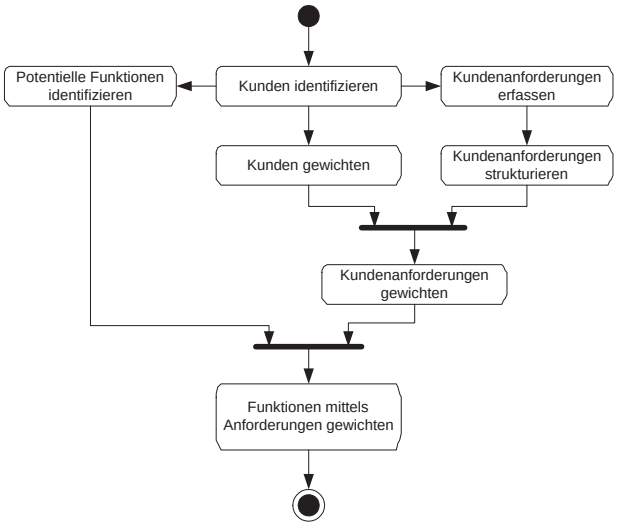



Abbildung 18: Vorgehen beim QFD (zusammengefasst aus [Her+00], S. 44ff)

Vereinfacht wird eine Untersuchung und Erfassung der Kundenbedürfnisse für die Methodenentwicklung vor allem durch die Tatsache, dass sie bereits von zahlreichen Autoren strukturiert und beschrieben wurden, da sie zu einem der Schwerpunkte in der Forschung auf dem Gebiet der Wirtschaftsinformatik gehören.[59] Die folgenden Abschnitte 4.3 bis 4.7 werden die vorhandene Literatur auf diesem Gebiet vorstellen. Die Ergebnisse der Zusammenfassung aus dem Abschnitt 4.8 werden im Rahmen dieser Arbeit auf zwei Arten angewendet:

- **Technik bei der Definition von Methoden:** Das QFD wird in den Prozess des ME integriert. Die im Folgenden erarbeiteten allgemeinen Anforderungen bilden einen ersten Anhaltspunkt für situationsangepasste Anforderungen an Methoden.
- **Grundlage für die Bewertung von Methoden des ME:** Im Kapitel 5 werden Methoden des ME einer Bewertung unterzogen. Die formulierten Anforderungen dienen der Schwachstellenanalyse dieser Methode und als Motivation zur Entwicklung der Methode aus Abschnitt 6.

Für beide Anwendungsfälle sind neben den Anforderungen Merkmale von Methoden zu erheben, die sich direkt überprüfen lassen. Auf eine Angabe allgemeingültiger Merkmale wird

[59] Problematisch ist diese Tatsache nur, wenn die Beschreibungen lediglich der guten Positionierung der eigenen Methode dienen.

verzichtet, vielmehr werden diese im Abschnitt 5.1 für den zweiten Anwendungsfall formuliert. Für den ersten Anwendungsfall wird die Erhebung der Merkmale in das Vorgehen zur Methodendefinition integriert, da auch hier gilt, dass insbesondere die Mächtigkeit einer Methode, deren ontologische Eigenschaften sowie deren Wiederverwendungsansätze sehr stark kontextabhängig sind. Die Forderung nach möglichst konkreten und messbaren Methodenmerkmalen schränkt deren Allgemeingültigkeit naturgemäß ein.

## 4.3 Allgemeine Anforderungen

Modelle bilden zum einen in ihrer Eigenschaft als Produkt von Methoden und zum anderen als deren mögliche Spezifikationsform einen geeigneten Ausgangspunkt bei der Bewertung der Qualität von Methoden. Modelle bilden die Voraussetzung für die erfolgreiche Planung und Gestaltung von komplexen Systemen und sind ein Mittel zur Kommunikation für den Systementwickler und interessierten Nutzern (vgl. [Fran98a], S. 5). Jedoch erscheinen nur wenige Kriterien der Modellqualität objektiv messbar. REMME fordert z. B. von der in Modellen und bei der Erstellung verwendeten Sprache[60] folgende Eigenschaften (vgl. [Remm97], S. 43):

- Nach kurzer Einarbeitungszeit leicht erlernbar, damit die Modelle als Instrument der Kommunikation eingesetzt werden können
- Übersichtliche und klare Darstellungen
- Veränderungen müssen mit angemessenem Aufwand möglich sein
- Komplexität des Realitätsbezuges muss reduziert werden, Analysen dessen müssen möglich sein
- Integrierbarkeit in die Sprache des Unternehmens

FRANK unterstreicht den interdisziplinären Charakter der Bestimmung einer Modellqualität bzw. der Qualität der verwendeten Modellsprache. Zusätzlich existiert die Forderung nach einer Investitionssicherheit für Unternehmen, die Mitarbeiter in der Modellierung schulen oder Werkzeuge beschaffen (vgl. [Fran98a], S. 5f). Aspekte, wie z. B. Intuitivität und Verständlichkeit, sind Herausforderungen mit erkenntnistheoretischem, psychologischem, soziologischem und philosophischem Hintergrund, wenn WITTGENSTEIN formuliert „The limit of my language means the limit of my world.“ ([Witt81], §5.6)

Für den Vergleich von Metamodellen und den zugehörigen Methoden im Business Process Redesign (BPR)[61] wurde in [HeBr96] ein systematischer Überblick erstellt. Das dort verwen-

[60] im Sinne des Abschnitts 2.5.4 also deren Metamodelle

[61] Bei HAMMER und CHAMPY steht diese Abkürzung in [HaCh93] für Business Process Reengineering; auf eine Trennung der Begriffe wird hier nicht weiter eingegangen.

dete universelle Beschreibungsraster bietet einen ersten Anhaltspunkt und wird in Tabelle 2 dargestellt.

| Kategorie | Fragestellung |
|---|---|
| Zielsetzung | Was ist die erklärte Zielsetzung der Methode? |
| Metamodell | Was sind die wichtigsten Gestaltungsobjekte? Wie sind die wichtigsten Objekte und deren Beziehungen definiert? |
| Vorgehen und Ergebnisse | Welche Ergebnisse werden in welchen Phasen erstellt? |
| Techniken | Welche Techniken unterstützen das Erstellen von Ergebnissen? |
| Rollen | Wer arbeitet im Projekt wann mit? |
| Einbettung | Ist die Methode Bestandteil eines umfassenden Gesamtmodells / Gesamtansatzes des Autors? |
| Anwendungsgebiet | Ist die Anwendung der Methode auf bestimmte Branchen oder Prozesstypen beschränkt? |
| Toolunterstützung | Unterstützt ein computergestütztes Werkzeug die Methode? |

Tabelle 2: Beschreibungsraster von Methoden (vgl. [HeBr96], S. 4)

## 4.4 Grundsätze ordnungsmäßiger Modellierung

Die **Grundsätze ordnungsmäßiger Modellierung (GoM)**[62] bieten Kriterien, nach denen Modelle bewertet werden können (vgl. [Schü98], S. 112). Die GoM wurden eingeführt, um semantische Gestaltungsempfehlungen für die Modellierung zu geben und vorhandene Freiheitsgrade bei der Modellierung so einzuschränken, dass die Zielgerichtetheit der Informationsmodellierung erhöht wird. Damit wurde nach Meinung von BECKER ET AL. die Grundlage für intersubjektiv vergleichbare Modelle gelegt (vgl. [Bec$^+$95], S. 437). SCHÜTTE verfolgt in seiner Arbeit die Ziele der Bewertung von Modellen anhand der Grundsätze ordnungsmäßiger Modellierung und die Zielvorgabe durch die GoM für die Modellierung selbst. Er legt dabei den nutzerbezogenen Qualitätsbegriff[63] zu Grunde (vgl. [Schü98], S. 112). Eine Übersicht über die Elemente und deren Zweckbezug liefert die Abbildung 19.

Die **Konstruktionsadäquanz** zielt „... auf die problemangemessene Nachvollziehbarkeit der Modellkonstruktion ...“ ([Schü98], S. 113). Sie bildet das wichtigste Kriterium und stellt sich dem Problem der Bewertung des Realitätsbezuges eines Modells[64] durch die Forderung nach dem Konsens zwischen den am Erstellungsprozess beteiligten Subjekten. Der Konsens umfasst das repräsentierte Problem und dessen Darstellung.

Die Bewertung der Eignung und Richtigkeit der Modellierungssprache ist Gegenstand des Grundsatzes der **Sprachadäquanz**. Die erste enthaltene Forderung nach einer möglichst geeigneten Sprache für die Modellierung zielt auf die Auswahl der zur Modellierung eingesetz-

[62] In dieser Arbeit wird Bezug auf die *neuen* Grundsätze ordnungsmäßiger Modellierung genommen, die in [Schü98] vorgestellt wurden. Die Arbeit in [Bec$^+$95] wurde hier ergänzt und neu strukturiert.

[63] zum nutzerbezogenen Qualitätsbegriff vgl. [Balz98], S. 256 bzw. die Ausführungen im Abschnitt 4.1

[64] der aufgrund des konstruktivistischen Modellverständnisses nicht feststellbar ist

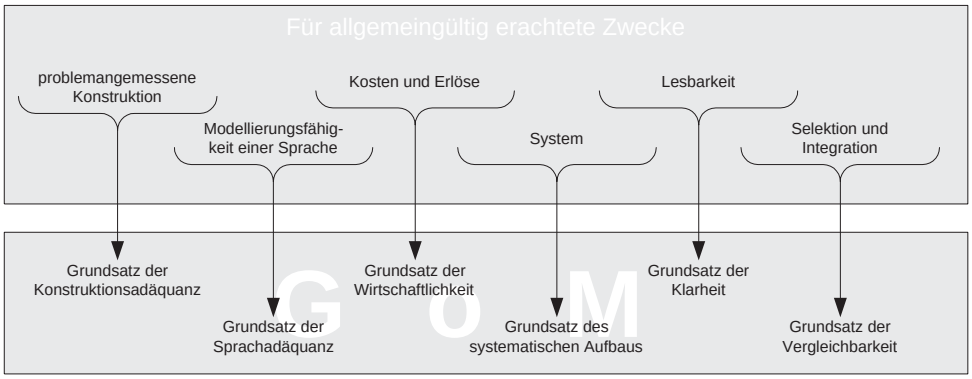



Abbildung 19: Zweckbezogene Ableitung der GoM (vgl. [Schü98], S. 115)

ten Metamodelle.[65] Unterschieden werden die Kriterien semantische Mächtigkeit[66], Formalisierungsgrad[67] und Verständlichkeit der Sprache. Die Richtigkeit der Sprachanwendung lässt sich anhand ihrer Grammatik bzw. ihres Metamodells nachweisen. Ausgedrückt wird dieser Umstand durch die Forderung nach *Konsistenz* zum Metamodell. Alle Elemente des Modells müssen vollständig im Metamodell spezifiziert sein.

Modelle binden während ihrer Erstellung Ressourcen, was wiederum zur Entstehung von Kosten führt. Dementsprechend unterliegen sie im betriebswirtschaftlichen Sinn der Forderung nach **Wirtschaftlichkeit**. Eine angestrebte Gewinnmaximierung im Unternehmen kann die Erstellung von Modellen fördern, sofern diese zu einer Kostensenkung oder Erlössteigerung führen, auf der anderen Seite jedoch auch im Ressourceneinsatz begrenzen. Auf dieser Grundlage erheben BECKER und SCHÜTTE die Forderung nach *Flexibilität* der Modelle. Kern dabei bildet der Zeitbezug von Modellen, der bereits im Abschnitt 3 vorgestellt wurde. Für Modelle existiert eine geforderte Amortisationsdauer, die durch eine sich ändernde Umwelt verkürzt wird. Nach Schütte wird die Wirtschaftlichkeit von Modellen nur nach Einrichtung einer Modellkosten- und -leistungsrechnung bewertbar (vgl. [Schü98], S. 129). Da diese noch nicht existiert, sind Erlöse und Kosten bei der Anwendung oder Erstellung von Modellen nicht prognostizierbar oder im Nachhinein zuordenbar. Neben der Flexibilität kann die *Kompliziertheit* der verwendeten Modellsprache die Wirtschaftlichkeit der Modelle beeinflussen.

Aus dem Qualitätsmerkmal des **Systematischen Aufbaus** leitet SCHÜTTE die Notwendigkeit eines sichtenübergreifenden Metamodells ab. Sein Fokus liegt dabei auf der Unterscheidung von struktur- und verhaltensorientierter Sicht (vgl. [Schü98], S. 131). Da dieser Grundsatz bereits direkt auf die Eigenschaften von Metamodellen abzielt, müssen diese nicht abgeleitet werden.

[65] In diesem Sinne sind Ereignisgesteuerte Prozessketten zur Abbildung der Ablauforganisation eher geeignet als das Entity-Relationship Model.

[66] SCHÜTTE verweist hierzu auf die Arbeiten von ZELEWSKI. Dieser formuliert in [Zele96] den Begriff der Modellierungsfähigkeit.

[67] vgl. dazu die Abstufungen aus Abschnitt 2.5.1

Unter dem Qualitätsmerkmal der **Klarheit** „... werden die Ziele der adressatengerechten Hierarchisierung, Layoutgestaltung und Filterung[68] subsumiert." ([Schü98], S. 131) Zum einen wird hier also der Einsatz des Prinzips der Hierarchisierung (siehe hierzu Abschnitt 3.2) und zum anderen eine geeignete grafische Aufbereitung der Modelle gefordert.

Die **Vergleichbarkeit** zweier Modelle „... zielt auf den semantischen Vergleich zweier Modelle ab." ([Schü98], S. 133) Es werden zwei Ebenen des Vergleichs beschrieben. Der Vergleich der Metamodelle geschieht mit dem Ziel festzustellen, welche der verwendeten Sprachen die mächtigere ist, um bei der Überführung zugehöriger Modelle einen Informationsverlust prognostizieren zu können. Der Vergleich von Modellen hat als Ziel die Feststellung einer Deckungsgleichheit bzw. Abweichung zwischen zwei Modellen. Solche Informationen werden beispielsweise benötigt, wenn Referenzmodelle eingesetzt und deshalb mit unternehmensspezifischen Modellen verglichen werden sollen.

Zusammenfassend definieren die GoM zum einen Rahmen für die Einordnung von Modelleigenschaften und zum anderen Forderungen an gute Modelle. Die GoM umfassen jedoch weder direkt messbare Kriterien für Modelle noch Regeln oder eine Beschreibung des Vorgehens für deren Erstellung. SCHÜTTE formuliert, dass vor Anwendung der GoM umsetzbare Handlungsanleitungen für konkrete Modellierungsvorhaben abgeleitet werden müssen (vgl. [Schü98], S. 134). Bevor aus den GoM eine Entscheidungsgrundlage für die Modellbildung entstehen kann, müssen messbare Kriterien gefunden und zugeordnet werden. Zusammen mit einer definierten Präferenzrelation[69] sind qualitative Aussagen zu Modellen möglich.

## 4.5 Anforderungen an das Konfigurationsmanagement

Der Abschnitt 3.4 hat aufgezeigt, dass Modelle im Zeitablauf Veränderungen unterliegen. Eine Methode der Modellerstellung muss entsprechend eine Änderungskontrolle und Verwaltung unterstützen. Gegenstand der folgenden Abschnitte ist die Erarbeitung zu berücksichtigender Anforderungen an Methoden aus Sicht des Modell-KMS. Hierzu wird in Abschnitt 4.5.1 zunächst die Grundlage für die weiteren Betrachtungen geschaffen. Anschließend untersucht Abschnitt 4.5.2 allgemeingültige Anforderungen an ein KMS. Die Ergebnisse werden in Abschnitt 4.5.3 für das hier relevante Gebiet der Modelle präzisiert und erweitert.

### 4.5.1 Grundlage der Anforderungsanalyse

Der Erfolg eines KMS hängt nicht nur maßgeblich von seinem Einsatz im Projekt, sondern insb. von der Umsetzung der Verfahren durch die Projektmitarbeiter ab. Voraussetzung dafür

[68] Die inhaltliche Filterung dient der Veränderung des Detaillierungsgrades; Die sprachliche Filterung führt zur (bei SCHÜTTE nicht weiter beschriebenen) Konfiguration des Metamodells.

[69] z. B. entstanden aus dem Konsens von Modellierer, Projektleiter und Kunde

ist ein hoher Akzeptanzgrad desselben. Hierzu muss es eine echte Unterstützung des Entwicklungsprozesses bei minimaler Behinderung der Arbeiten ermöglichen (vgl. auch Silver Bullet AntiPattern ([Bro+99], S. 63ff) in Verbindung mit $P^2$ AntiPattern ([Bro+99], S. 179)). Somit ist die Erfassung der Anforderungen potentieller Benutzer in den Mittelpunkt der folgenden Untersuchungen zu stellen. Um in der Analyse von einer konkreten Organisationsstruktur abstrahieren zu können, wird das Konzept der *organisatorischen Rolle* verwendet, wie es KIESER und KUBICEK definieren (vgl. [KiKu83], S. 397). Mit den einzelnen Rollen verknüpfte Ziele und Aufgaben bilden folglich die Basis der Anforderungen an ein KMS und stellen somit den Ausgangspunkt zur Bestimmung der notwendigen Strukturen und Operationen dar.

Als Grundlage des hier verwendeten Rollenmodells werden Erfahrungen des SEI der Carnegie Mellon University in der Softwareentwicklung herangezogen (vgl. [Dart91], S. 2f und [Bro+91], S. 4f), die für die Zwecke dieser Arbeit generalisiert wurden. Demzufolge sind die Aufgaben und Ziele der Rollen Projektmanager, KM-Verantwortlicher, Entwickler, Tester, Qualitätsbeauftragter und Kunde im KMS zu unterstützen (vgl. [Dart91], S. 2f). Die hiermit verbundene Strukturierung potentieller Benutzer dient gleichzeitig auch als Basis zur Vermeidung von Interessenkonflikten zwischen den Projektbeteiligten. Aufgrund ihrer unterschiedlichen Aufgaben und Ziele benötigen sie jeweils nur eine Teilmenge der verfügbaren Funktionalität. Die erforderlichen Zugriffsbeschränkungen können mit den Rollen verknüpft werden und bilden somit den Ausgangspunkt zur Schaffung eines Berechtigungskonzepts innerhalb einer möglichen Entwicklungsumgebung.

### 4.5.2 Allgemeine Anforderungen an ein KMS

Zur Untersuchung allgemeiner Anforderungen an ein KM sind im Folgenden grundlegende und ergänzende Erfordernisse zu unterscheiden. Erstere werden durch die Bestimmungen von DIN 10007 (vgl. [Deut96]) gebildet, die das Minimum abzudeckender Funktionen in einem KM darstellen. Ergänzungen resultieren primär aus den KM-Erfahrungen in der Softwareentwicklung. Ihre Umsetzung vermeidet übermäßige Behinderungen des Entwicklungsprozesses durch den erhöhten Verwaltungs- und Dokumentationsaufwand.

Die Bestimmungen der Norm DIN EN ISO 10007 zum Konfigurationsmanagement identifizieren vier grundlegende Bereiche eines KM (vgl. [Deut96], S. 7f):

- Konfigurationsidentifikation,
- Konfigurationsbuchführung,
- Konfigurationsüberwachung und
- Konfigurationsaudit

Neben der Abdeckung dieser Teilgebiete zeichnet sich ein umfassendes KMS insb. auch durch deren Integration aus. Sie wird durch einen zusätzlichen Bereich *KM-Planung und Organisation* übernommen, der in Abhängigkeit vom bearbeiteten Projekt zu gestalten ist (vgl. [Sayn98], S. 13). Die hier zu erarbeitende Konzeption kann jedoch lediglich Ansatzpunkte für die Integration der Teildisziplinen spezifizieren, deren Nutzung durch den Bereich der Organisation und Planung projektabhängig zu entscheiden ist.

Die Hauptaufgaben der **Konfigurationsidentifikation (KI)** liegen sowohl in der Bestimmung zu verwaltender Elemente (Konfigurationseinheiten, Dokumente) als auch in der Erstellung verbindlicher Produktversionen (vgl. [Deut96], S. 7). Sie gliedert sich dazu in einen *fachlich-inhaltlichen* und einen *formalen* Teil (vgl. [Sayn98], S. 15). Letzterer legt neben Konfigurationseinheiten und Dokumenten auch Aufzeichnungsverfahren und Nummerierungsregeln fest. Darauf aufbauend definiert der inhaltlich-fachliche Teil Anforderungen und Bestimmungszeitpunkte für **Bezugskonfigurationen (BK)**. Sie ist die „... formell zu einem vorgegebenen Zeitpunkt festgelegte Konfiguration eines Produkts, die als Grundlage für weitere Tätigkeiten dient.“ ([Deut96], S. 5)

Aufbauend auf den in der KI geschaffenen Grundlagen kommen die Maßnahmen zur Überwachung der Änderungen an einer Konfigurationseinheit, also zur Erstellung von Versionen derselben zum Einsatz. Sie werden unter dem Begriff der **Konfigurationsüberwachung (KÜ)** zusammengefasst (vgl. [Deut96], S. 5). Anmerkung 3 der Norm DIN EN ISO 10007 ergänzt außerdem, dass die KÜ auch „... die Maßnahmen zur Bewertung, Koordination, Genehmigung oder Ablehnung und von Änderungen ...“ ([Deut96], S. 5) einschließt. Sie werden unter den Begriff des **Change Request (CR)** Verfahrens subsumiert. Hierbei sind insb. auch Anpassungsmöglichkeiten an die Projektgröße und den Projektfortschritt vorzusehen (vgl. [Pres92], S. 704ff).

Gegenstand der **Konfigurationsbuchführung (KB)** ist die Sicherstellung der erforderlichen Dokumentation sowohl bez. des notwendigen Umfangs als auch der Aktualität. Sie besitzt somit im Wesentlichen Dienstleistungscharakter für andere Funktionsbereiche. Die erfassten Informationen erhalten zusätzliche Relevanz, da sie gleichzeitig als Grundlage des Konfigurationsaudits dienen.

Neben den dokumentierenden Tätigkeiten der KI und der KÜ sowie der zu großen Teilen protokollierenden Tätigkeit der KB, kommt dem **Konfigurationsaudit (KA)** die kontrollierende Rolle zu. Es werden hierzu zwei wesentliche Teilbereiche, der *funktionsbezogene* und der *physische* KA unterschieden (vgl. [Deut96], S. 10). Der erstere bezieht sich auf den Abgleich des Produkts mit der zugrunde liegenden Anforderungsspezifikation, während im **physischen KA** eine Gegenüberstellung der aktuellen Konfiguration mit den vorhandenen Konfigurationsdokumenten erfolgt. Allerdings ist ein funktionsbezogener KA für Modelle nicht direkt anwendbar, da aufgrund der subjektiven Realitätswahrnehmung durch die Modellersteller ihr objektiver Test

nicht möglich ist. Stattdessen muss dem späteren Anwender die Möglichkeit gegeben werden, die Eignung des Modells für seine Zwecke zu prüfen.[70] Der Begriff des Audit ist aber nicht allein auf das verwaltete Produkt zu beschränken, sondern auch auf das KMS zu erweitern. Neben der Wirksamkeit implementierter Abläufe und Strukturen ist insb. auch die Übereinstimmung der angewandten Verfahren mit der Dokumentation des KMS zu überprüfen (vgl. [Deut96], S. 18).

Erfahrungen aus der Softwareentwicklung zeigen, dass die aus der DIN Norm ableitbaren KM-Aspekte zumindest um die Bereiche Prozessunterstützung, Teamwork und Buildunterstützung zu erweitern sind (vgl. auch [Dart91], S. 3ff; [Dart92], S. 2 und [Bro+91], S. 5ff). Mit einer möglichst optimalen **Unterstützung des Entwicklungsprozesses** wird eine hohe Akzeptanz der durch das KMS bedingten, unvermeidlichen Zusatzbelastungen bei den betroffenen Entwicklern erreicht. Sie ist Voraussetzung dafür, dass die Bestimmungen und Verfahren des KM im Projekt eingehalten werden. Aufgrund des hier zugrunde gelegten konstruktivistisch geprägten Modellbegriffes erhält insb. die Konstruktionsleistung des einzelnen Modellierers eine besondere Bedeutung. Bei mehreren Modellerstellern muss auch von unterschiedlichen Modellbildungsprozessen ausgegangen werden. Für die Konzeption des Modell-KMS ergibt sich daraus die Schwierigkeit, die vorab nicht bestimmbaren Vorgehensweisen möglichst ohne Einschränkungen für den einzelnen Entwickler unterstützen zu müssen. Gleichzeitig sind aus Gründen der Qualitätssicherung und der Nachvollziehbarkeit durchgeführter Arbeiten das Erreichen bestimmter Meilensteine in der Entwicklung durch das KMS überprüfbar sicherzustellen.

Die Unterstützung des **Teamwork** durch das KMS bezieht sich sowohl auf die Zusammenarbeit zwischen Modellanwender und -ersteller als auch zwischen den Entwicklern untereinander. Erstere umfasst insb. die Schaffung einer gemeinsamen Kommunikationsbasis, sodass Modellanforderungen und Mittel zu ihrer Überprüfung am Produkt erarbeitet werden können. Hiermit wird gleichzeitig auch Fehlern dritter Art (vgl. [Gait79], S. 8), also der Erstellung von Lösungen für nicht existente Probleme effektiv vorgebeugt. Eine erste Strukturierung der Zusammenarbeit innerhalb des Entwicklerteams erfolgte bereits durch die Einführung von Rollen im KMS. Allerdings bedarf auch die Kooperation innerhalb der identifizierten Benutzergruppen geeigneter Regelungen. So sind z. B. aufgrund der unterschiedlichen Modellbildungsprozesse die Arbeiten einzelner Modellersteller voneinander zu isolieren. Gleichzeitig müssen aber auch geeignete Verfahren bereitgestellt werden, die eine kontrollierte Integration der Teilarbeitsergebnisse ermöglichen.

Mit dem Begriff des *Build* wird in der Softwareentwicklung zumeist die Erstellung einer lauffähigen Anwendung aus den entsprechenden Quelltexten durch den Einsatz von Compiler

[70] Dies entspricht der Sicherstellung des pragmatischen Modellmerkmals (siehe dazu auch Abschnitt 2.3).

und Linker verknüpft. Wie STACHOWIAK ausführt, können auch Modelle Originalcharakter besitzen, also als Vorlage für weitere Abbildungen dienen (vgl. [Stac73], S. 131). So lässt sich z. B. aus einem UML-Klassenstrukturdiagramm der Quellcode für die entsprechenden Java-Klassendefinitionen automatisiert erzeugen. Im KMS sind demzufolge entsprechende Verfahren der **Buildunterstützung** vorzusehen, die eine Überführung der verwalteten Modelle erlauben.

### 4.5.3 Anforderungen an ein KMS aus Modellsicht

Nachdem zunächst allgemeine Anforderungen an ein KMS erarbeitet wurden, stehen nun notwendige Ergänzungen und Erweiterungen zur Verwaltung von Modellen im Mittelpunkt der Betrachtungen. Hierbei sind, neben der Erfassung des Modellinhalts bzw. seiner Veränderungen, insb. auch prozessbezogene und organisatorische Anforderungen zu berücksichtigen.

Gemäß den Ausführungen von WAND (vgl. [Wand89], S. 540f) werden Entitäten des Originals aufgrund bestimmter Eigenschaften wahrgenommen. Hierbei ist es jedoch nicht erforderlich alle zu erfassen (Konzept der Präterition bei STACHOWIAK, vgl. [Stac73], S. 155). Die Darstellung der wahrgenommenen Eigenschaften erfolgt durch Zuordnung entsprechender Attribute zu den Modellelementen. Zieht man die Darlegungen STACHOWIAKs hinzu (vgl. [Stac73], S. 155), so bilden Modellelemente „... Merkmale und Eigenschaften von Individuen, Relationen zwischen Individuen, Eigenschaften von Eigenschaften, Eigenschaften von Relationen usw. ...“ ([Stac73], S. 134) ab. Die Entscheidung, ob beliebige Elemente eine Attributfunktion erfüllen oder als Individuen fungieren, ist vom jeweils bearbeiteten Zusammenhang abhängig. Für die Verwaltung von Modellen im KMS bleibt somit festzuhalten, dass geeignete Strukturen zur Ablage von **Modellelementen** (inkl. ihrer Attribute) und deren Beziehungen zu berücksichtigen sind.

Ein zentraler Bestandteil des hier verwendeten Modellbegriffes wird durch den **Problembezug** gebildet, der die Realisierung des pragmatischen Modellmerkmals sicherstellt (vgl. auch Abschnitt 2). Er dient der Erfassung von Anforderungen an das zu erstellende Modell und stellt daher einen wichtigen Vertragsbestandteil zwischen dem Modellnutzer und den Modellentwicklern dar. Probleme entstehen jedoch während seiner Erarbeitung, da der Nutzer zu Beginn des Projektes nicht „... genau weiß, was er will ...“ bzw. keine Vorstellungen über die potentiellen Möglichkeiten hat (vgl. [DaRa97], S. 5).

Die Zusammenstellung des Problembezugs wird des Weiteren zumeist durch das Fehlen einer Kommunikationsbasis, also eines gemeinsamen Begriffsverständnisses erschwert. Aus diesem Grund ist zusätzlich eine **Namenskonvention** im Projekt zu führen, die den verwendeten Basiswörtern die jeweils zutreffende Definition bzw. Semantik zuordnet. Allerdings greift die alleinige Verwaltung von Begriffsdefinitionen zu kurz.

Im Rahmen der allgemeinen Anforderungen an KMS wurde bereits die Notwendigkeit zur Unterstützung von Teamarbeit hervorgehoben. Übertragen auf den hier vorliegenden Fall der

Modellverwaltung bedeutet dies, dass einzelne Entwickler an Teilmodellen arbeiten, die abschließend integriert werden. Hierbei sollte aber nicht nur eine Auswahl, sondern eine echte Verschmelzung der Arbeitsergebnisse möglich sein. In diesem Rahmen treten gemäß den Darlegungen von HARS (vgl. [Hars94], S. 176ff) und ROSEMANN (vgl. [Rose96], S. 187ff) Namens-, Typ- und Strukturkonflikte auf. Voraussetzung zur Lösung ersterer ist die Vermeidung von *Homonymie* und *Synonymie* Sprachdefekten (vgl. [Rose96], S. 187). Während sich letztere durch Verwaltung entsprechender Synonymlisten zu den einzelnen Basiswörtern vermeiden lassen, kann das Homonymieproblem nur durch geeignete Regeln zur Erstellung und Anwendung der Namenskonvention behoben werden.

Wie bereits Abschnitt 4.5.2 darlegte, sind gewünschte Veränderungen am Modell durch entsprechende Change Requests in den Entwicklungsprozess einzubringen. Sie bedürfen somit einer geeigneten Dokumentation und Verwaltung im KMS. Um zusätzlich auch die Entwicklung einzelner Komponenten nachvollziehen zu können, sind außerdem Dokumentationen einzelner Versionen, sowohl des Modells als auch seiner Bestandteile, in Form von Versionsdokumenten zu führen.

Die Modellentwicklung erfordert, neben der Bereitstellung geeigneter Verwaltungs- und Modellieroperationen, insb. auch eine Unterstützung der Zusammenarbeit zwischen Anwender und Entwickler bzw. zwischen den Entwicklern. Dieses Gebiet wird unter den Begriff der *kollektiven Modellbildung* zusammengefasst (vgl. [Herr91], S. 326ff und [Schü98], S. 61). Erstere erfordert eine Ausrichtung der individuellen Konstruktionsleistungen der Modellierer auf die Ziele des Anwenders. Das hierzu notwendige *mutual understanding* zwischen Anwender und Entwickler (vgl. [Herr91], S. 328f) entsteht jedoch nur bei Kenntnis der jeweiligen psychologischen Typologien. Dieser Prozess kann deshalb nicht durch das Modell-KMS unterstützt werden.

Die Kooperation der Modellersteller untereinander erfordert primär die möglichst optimale Unterstützung der verschiedenen Vorgehensweisen aufgrund der individuellen Ausprägungen von Kenntnissen, Psyche und Wertesystem. Dies bedingt private Arbeitsbereiche, im Folgenden als **Workspaces** bezeichnet, die den jeweiligen Entwickler vor Manipulationen anderer schützen, gleichzeitig aber auch seine eigenen Änderungen auf den geschützten Bereich beschränken. Für die Erstellung des Gesamtmodells aus den einzelnen Arbeitsergebnissen sind zusätzlich Verfahren zur kontrollierten Integration von Teilmodellen vorzusehen.

Durch die Anwendung des in Abschnitt 4.5.1 entwickelten allgemeinen Rollenmodells wird insb. dem Kunden eine weitere Spezialisierung zuteil. Der durch sie repräsentierte Modellanwender kann sowohl unternehmensinternen als auch -externen Charakter besitzen. Hieraus leiten sich wiederum Konsequenzen für die erforderlichen Test- und Qualitätssicherungsmaßnahmen ab. So wird u. U. in unternehmensinternen Modellierungsprojekten der Entwicklungsge-

schwindigkeit eine höhere Priorität als der Qualitätssicherung eingeräumt. Dementsprechend sind im KMS Optionen vorzusehen, die eine Anpassung der Konzeption an die Gegebenheiten des aktuellen Projektes bzw. des einsetzenden Unternehmens ermöglichen.

### 4.5.4 Zusammenfassung

Aus Sicht der Norm DIN EN ISO 10007 sind im KMS einer Methode die vier grundlegenden Bereiche Konfigurationsidentifikation, -buchführung, -überwachung und -audit vorzusehen. Diese sind gemäß der Erfahrungen aus der Softwareentwicklung um eine Unterstützung des Teamworks, Builds und Entwicklungsprozesses zu ergänzen. Zur Repräsentation und Verwaltung des Modellinhalts muss ein KMS Modellelemente zur Darstellung der erfassten Entitäten des Originals, deren Problembezug und die Namenskonventionen enthalten. Aus Sicht der durch Methoden beschriebenen Prozesse der Modellentwicklung sind entsprechende Modellieroperationen in getrennten Workspaces zu spezifizieren, deren Ausführung an die Rechte der jeweiligen Rolle im Projekt zu knüpfen sind.

## 4.6 Ontologische Qualifizierung von Modellierungssprachen

Ziel der in diesem Abschnitt beschriebenen Qualifizierung ist die Bewertung der Qualität der Repräsentation der Wahrnehmung. Als bekanntester Vertreter dient das Bunge-Wand-Weber Modell (BWW) als Hilfsmittel zur ontologischen Qualifizierung von Modellierungssprachen der Darstellung der Realwelt. Es definiert eine Menge von *Kern-Phänomenen*, die in der Entwicklung von Informationssystemen bei der Modellierung eingesetzt werden (vgl. [WaWe95], S. 203ff). Um deren strukturelle und dynamische Eigenschaften beschreiben zu können, werden von WAND und WEBER in ihrer Gesamtheit für diese Aufgabe ausreichende Konzepte gefordert, welche die Tabelle im Anhang 13.1 im Überblick darstellt. Nach der Analyse mit Hilfe des BWW lassen sich zu folgenden Punkten Aussagen über die Konzepte formulieren (vgl. [GrRo00], S. 77):

- **Ontologische Unvollständigkeit**: Ein *konstruktives Defizit* existiert für die Sprache genau dann, wenn einem der ontologischen Konstrukte kein Konzept aus der Grammatik der Modellsprache zugeordnet werden kann.
- **Ontologische Klarheit**: Diese ist gegeben, wenn für die analysierte Grammatik folgende Defekte ausgeschlossen werden können:
  - **Konstruktüberladung**: In der Grammatik existiert für ein ontologisches Konstrukt mehr als ein Konzept.
  - **Konstruktredundanz**: In der Grammatik existiert ein Konzept, das sich mehreren ontologischen Konstrukten zuordnen lässt.

– **Konstruktüberschuss**: In der Grammatik existieren Konzepte, die sich keinem ontologischen Konstrukt zuordnen lassen.

Nach einer Untersuchung der bisher mit dem BWW erzielten Ergebnisse bei der Analyse von Modellierungssprachen,[71] stellen GREEN und ROSEMANN fest, dass die Konstrukte *concievable state space*, *concievable event space* und *lawful state space* bisher in noch keiner Grammatik identifiziert wurden.[72] An dieser Stelle ist nach Meinung von GREEN und ROSEMANN eine Reduzierung der Anforderungen notwendig, da das BWW in diesem Punkt „over-engineered" ist. Wenn das BWW nicht zu umfangreiche Anforderungen stellt, so definieren Modellierungssprachen zumeist bewusst eine bestimmte Perspektive. Dementsprechend muss das BWW vor der Anwendung für den entsprechenden Fokus „individualisiert" werden. Daneben existiert auch eine Notwendigkeit der Erweiterung des BWW. GREEN und ROSEMANN weisen darauf hin, dass in Unternehmen Konstrukte, wie das der Strategie oder des unternehmensweiten Ziels, im BWW keine Entsprechung besitzen (vgl. [GrRo00], S. 82f). OPDAHL und HENDERSON-SELLERS haben bei der Anwendung des BWW zur Bewertung des Object-oriented Process, Environment and Notation (OPEN) (vgl. [Gra+97], S. xv) folgende Ergebnisse formuliert (vgl. [OpHe99]):

- Das BWW ist geeignet zur Analyse von Modellkonstrukten zur Repräsentation *konkreter Dinge*, weniger jedoch für *mentale Konzepte*[73].
- Neben *konkreten Dingen* und *mentalen Konzepten* werden *soziale Konstrukte* als dritte Kategorie von Phänomenen identifiziert, die zwar in Modellen abgebildet jedoch vom BWW nicht ontologisch auf Vollständigkeit überprüft werden können.

Zusammenfassend wird ersichtlich, dass das BWW nach einer geeigneten Anpassung ein Hilfsmittel bei der Bewertung von Modellierungssprachen ist, wenn diese sich auf eine konkrete Gegenstandsebene beziehen. Im Rahmen dieser Arbeit führte der Versuch der Anwendung auf Sprachen zur Definition von Methoden zu keinen sinnvollen Ergebnissen.[74] Für die ontologische Qualifizierung von Modellierungssprachen bei der Entwicklung von Methoden wird

[71] Angewendet wurde das BWW bisher für Datenflussdiagramme (DFD), das Relationenmodell, das Entity-Relationship Model (ERM), die Object Modelling Language (OML) der OPEN Spezifikation, die Unified Modeling Language (UML) und einige andere daten- und funktionsorientierte Grammatiken.

[72] Dies wird auch durch die später von OPDAHL und HENDERSON-SELLERS durchgeführte Untersuchung der UML bestätigt (vgl. [OpHe02], S. 49ff).

[73] OPDAHL und HENDERSON-SELLERS bezeichnen z. B. eine mentale Aggregation als Zuordnung von vorher nicht in Beziehung stehenden Dingen zu einem Ganzen, ohne diese Dinge physisch zu bewegen oder zu verändern; Eine physische Aggregation ist die Zusammenfügung von Teilen zu einem Ganzen (vgl. [OpHe99]).

[74] Dazu muss eine Methode als Informationssystem verstanden werden, um vor allem die Kern-Phänomene *thing*, *class*, *property*, *value*, *state*, *transformation*, *system* und *subsystem* zuzuordnen. Entsprechendes ist sicherlich denkbar, es bleibt jedoch fraglich, ob dies noch im Sinne des BWW geschieht, da es mit dem *thing* eine begriffliche Verankerung besitzt, die bei einer „normalen" Anwendung keiner weiteren Klärung bedarf, wohingegen das Begreifen eines Konzepts oder Vorgehens als Ding eine Abstraktion darstellt, die eine anschließende *Bewertung* der Sprache subjektiviert.

deshalb im Folgenden mit Bezug auf die Abschnitte 2.5, 3.3 und 3.4 die Menge der Konzepte zur ontologischen Qualifizierung in Tabelle 3 festgelegt. Hierzu wurde vor allem die Abstraktionsebene des ursprünglichen BWW angehoben,[75] wesentliche Konstrukte wurden hinzugefügt (z. B. das von OPDAHL in [Opda97] vorgeschlagene Konstrukt der Perspektive) und aus Sicht des Autors nicht benötigte Konstrukte (z. B. *lawful state space*) entfernt. Zur Untersuchung der Konzepte von Modellierungssprachen des ME wird im späteren Verlauf der Arbeit die *ontologische Unvollständigkeit* und *Klarheit* entsprechend dieser Übersicht überprüft.

| **Konstrukt** | **Erklärung** |
|---|---|
| Modellelement | Modellelemente sind elementare Bestandteile eines Modells. Die Realwelt besteht aus Dingen, die von Modellelementen abgebildet werden. Modellelemente können zusammengesetzt werden. |
| Eigenschaft | Modellelemente werden durch Eigenschaften beschrieben. Eigenschaften verbinden Modellelemente mit *Werten*, die ihren Bezug zur Realwelt darstellen. |
| Konzept | Konzepte beschreiben eine Teilmenge der Eigenschaften von Modellelementen. Sie entstehen, wenn Modellelemente unter einer bestimmten *Perspektive* wahrgenommen werden. |
| Perspektive | Die Nutzer einer Methode beschreiben ihre Sichten auf das Problem in Perspektiven. Eine Perspektive besteht aus einer Menge an Konzepten und deren Eigenschaften. |
| Repräsentation | Perspektiven können durch Repräsentationen der Konzepte bzw. Modellelemente dargestellt werden. |
| Klasse | Eine Klasse ist eine Menge an Modellelementen, die über den Besitz einer bestimmten einzelnen Eigenschaft definiert wird. |
| Art | Eine Art ist eine Menge an Modellelementen, die über den Besitz mehrerer Eigenschaften definiert wird. |
| Zustand | Ein Vektor aus Werten für alle Eigenschaften eines Modellelements ist dessen Zustand. |
| Transformation | Eine Transformation ist der Übergang von einem Zustand in einen anderen. |
| Historie | Die chronologische Abfolge von Zuständen eines Modellelements ist dessen Historie. |
| Wirken auf | Ein Ding wirkt auf ein anderes ein, wenn dessen Existenz die Historie des anderen Dinges beeinflusst. |
| Beziehung | Zwei Dinge stehen in Beziehung, wenn ein Ding auf das andere einwirkt. Des Weiteren können sie durch eine verbindende Eigenschaft in Beziehung stehen. |
| Modell(-system) | Eine Menge von untereinander in Beziehung stehenden Modellelementen ist ein Modellsystem oder verkürzt ein Modell. |
| Zusammensetzung | Die Modellelemente in einem Modell sind dessen Zusammensetzung. |
| Modellumgebung | Modellelemente oder Dinge die nicht im Modell enthalten sind, aber mit den Modellelementen interagieren, bilden die Modellumgebung. |
| Modellstruktur | Die Menge der Beziehungen zwischen den Modellelementen des Modells und den Beziehungen zwischen den Modellelementen und der Umgebung bilden die Struktur des Modells. |
| Teilmodell | Ein Teilmodell ist ein Modell, dessen Zusammensetzung und Struktur eine Teilmengen der Zusammensetzung und Struktur eines anderen Modells ist. |
| Ziel | Ein Ziel beschreibt eine Menge von Zuständen für eine Perspektive. |
| Strategie | Eine Strategie beschreibt eine Gruppe von Transformationen, die zu einem bestimmten Ziel führen. |

Tabelle 3: Ontologische Qualifizierung von Methoden

[75] Aus Dingen wurden z. B. Modellelemente.

## 4.7 Methodenbewertung

Die bisherigen Ausführungen in Abschnitt 4 beziehen sich auf die Bewertung einzelner Methodenbestandteile. In diesem Teil wird vor allem Bezug genommen auf die gesamte Methode und deren Anwendung. Die einzelnen Punkte entziehen sich zumeist einer direkten Messbarkeit und unterliegen zudem einer subjektiven Bewertung durch den Methodenanwender. SINZ formuliert beispielsweise allgemein für die Anforderungen an „gute“ Methoden folgende Schwerpunkte (vgl. [Sinz98], S. 27f):

- Unterstützung bei der Aufstellung umfassender Modelle: Abbildung von Struktur und Verhalten von Systemen
- Unterstützung bei der Aufstellung richtiger Modelle: Konsistenz und Vollständigkeit bzgl. Begriffssystem; Struktur- und verhaltenstreu gegenüber abgebildeter Realität
- Unterstützung bei der Aufstellung geeigneter Modelle: Bezug zu Modellierungszweck (nicht weiter ausgeführt, sehr unscharf)
- Unterstützung bei Komplexitätsbewältigung von Modellen: Mehrstufige Verfeinerung, Außen- und Innensicht von Systemen, Ebenen und Sichten

Für den Vergleich und die Bewertung von Methoden und die darin enthaltenen Metamodelle nach messbaren Kriterien schlagen VAN HILLERSBERG und KUMAR die nähere Untersuchung folgender Punkte mit den entsprechenden Fragestellungen vor (vgl. [VaKu99], S. 114f):

- **Scope**: Welche Bandbreite besitzen die Konzepte?
- **Basis**: In welcher Teildisziplin der Systementwicklung hat die Methode ihren Ursprung (z. B. Programmierung, Analyse, Entwurf)?
- **Formalität**: Wie präzise ist die Methode definiert (formal, semiformal, informal)?
- **Prozess**: Wie erfolgte die Definition der Methode, wie wurde sie validiert?

Für den Prozess der Validierung von Methoden schlagen die Autoren weiter vor, zu untersuchen, ob die Validierung durch den Erzeuger, ein Komitee oder die breite Öffentlichkeit erfolgt ist. Er unterscheidet die **interne Validierung**, als eine begrenzte und subjektive Bewertung durch die Autoren mit einer Prüfung auf Vollständigkeit und Konsistenz, und die **externe** Validierung, die beispielsweise über drei unabhängige und nicht an der Entwicklung beteiligte Experten erfolgen kann oder bei der die Methode über Seiten im Internet einer breiten Öffentlichkeit zur Validierung zur Verfügung gestellt wird (vgl. [VaKu99], S. 114).

ROSSI und BRINKKEMPER formulieren eine Reihe von Metriken, mit denen ein Ersteller von Methoden diese überprüfen und ein Nutzer der Methoden aufgrund messbarer Kriterien eine Methodenauswahl treffen kann (vgl. [RoBr95], S. 200). Metriken selbst können keine Aussage über die Qualität und Eignung einer Methode treffen, jedoch lassen sie eine Klassifikation und Ordnung der Ansätze zu. Als Voraussetzung für die Messung von Eigenschaften liefern die Autoren eine geeignete Beschreibung. Genutzt wird in ihren Arbeiten ein Metamodell, dass die Elemente Object, Property, Relationship und Role (OPRR)[76] enthält. Zurückzuführen ist dieses auf die Arbeiten von WELKE und die Erweiterungen von SMOLANDER (siehe dazu [Welk92] und [Smol91]).

ROSSI und BRINKKEMPER bestimmen mittels ihrer Metriken die **Komplexität** einer Methode. Unterschieden wird zwischen der Komplexität des Lernens und Verstehens einer Methode und der Komplexität des Produkts, wobei nur letztere sich in numerischen Attributen zur Methode ausdrücken lässt (vgl. [RoBr95], S. 205). Auf **Technikebene** lassen sich als unabhängige Maßzahlen die Anzahl Objekttypen, Beziehungen, Properties, Properties pro Objekttyp, Properties pro Relationship und die Anzahl der Beziehungen, die mit einem Objekttypen verbunden werden können, erheben (vgl. [RoBr95], S. 206ff). Anschließend lassen sich entsprechend aggregierte Maßzahlen bestimmen. Auf **Methodenebene** werden diese entsprechend zu Komplexitätsmaßen der Methode kummuliert.[77]

Bei der Bewertung von Petri-Netzen verfolgt ZELEWSKI den Ansatz einer **relativen Vorteilhaftigkeitsbewertung**, in deren Rahmen die einzelnen Petri-Netz-Varianten auf ihre Eignung zur Modellierung von Systemen hin untersucht werden (vgl. [Zele96], S. 369f). Der von ihm formulierte Kriterienkatalog bewertet im ersten Teil die **Modellierungsfähigkeit** und Universalität des Ansatzes, welche sich aus der Menge an Sachverhalten ergibt, die der Ansatz zu beschreiben vermag (vgl. [Zele96], S. 370).[78] Den zweiten Teil der **Modellierungsgüte** unterteilt er in **pragmatisch determiniert** mit der zentralen Fragestellung „Lassen sich reale Probleme praxisgerecht lösen?“ und **theoretisch determiniert**. Pragmatische Determinanten sind im Wesentlichen:

- **Konstruktivität** (Unterstützung der Konstruktion von Modellen durch Prinzipien, wie hierarchische Modellverfeinerung oder Modulbildung),

[76] ROSSI und BRINKKEMPER liefern für die Elemente folgende Definitionen: „*Object* is a ’thing’ which exists on its own ... *Properties* are the describing/qualifying charecteristics associated with the other meta-types ... *Relationship* is an association between two or more objects ... *Role* is a name given to the link between an object and its connection with a relationship.“ ([RoBr95], S. 202f)

[77] Da in [RoBr95] Methoden als eine Sammlung von Techniken verstanden werden.

[78] ZELEWSKI unterscheidet eine *allgemeine* (auf Basis der Turing Mächtigkeit) und *spezielle* Mächtigkeit (zur Koordinierung von Produktionsprozessen, z. B. mittels Sequenzen oder paralleler Verarbeitung) (vgl. [Zele96], S. 370).

- **Analysierbarkeit** (Vielfalt der Auswertungsmöglichkeiten, die die Methode für ihre Modelle bietet),
- **Effizienz** des Modellierers bei der Anwendung der Methode zur Modellkonstruktion und -auswertung und
- **Benutzerfreundlichkeit** (z. B. durch Unterstützung von grafischen Darstellungen)

Theoretisch determiniert sind beispielsweise Eindeutigkeit und Konsistenz der Konzepte, Grad der Formalisierung, Einheitlichkeit, etc.[79] Für die einzelnen Kriterien werden entweder nominale Skalen, mit den Werten *erfüllt* und *nicht erfüllt*, oder ordinale Skalen, mit den Werten *sehr niedrig*, *niedrig*, *mittelmäßig*, *hoch* und *sehr hoch* definiert. Zusammengefasst werden die Ergebnisse in einem Polarprofil, wie es Abbildung 20 darstellt.

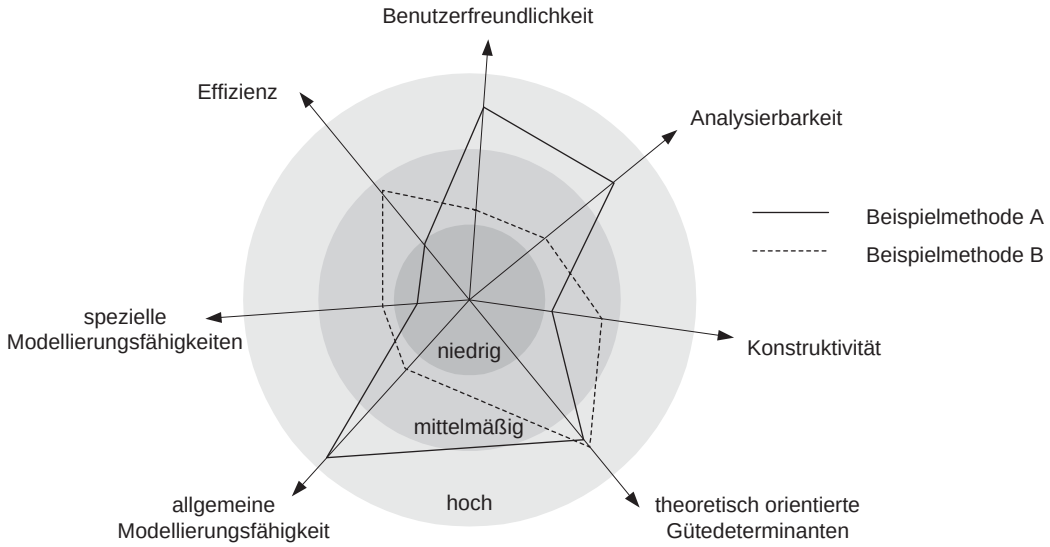



Abbildung 20: Beispielhaftes Polarprofil zur Methodenbewertung (i. A. a. [Zele96], S. 379)

Bei der Anpassung von Methoden darf nach Meinung von BRINKKEMPER ET AL. keine Methode entstehen, die einen der folgenden Defekte aufweist (vgl. [Bri+99], S. 219):

- Interne Unvollständigkeit: Ein Methodenfragment wurde ausgewählt, ohne dass die Elemente, von denen dieses Fragment abhängig ist, integriert wurden (z. B. wurde ein Metamodell zur Datenmodellierung aufgenommen ohne die zugehörigen Verfahrensanweisungen und Werkzeuge)

[79] Beschrieben werden insgesamt die 11 Determinanten Konsistenz, Eindeutigkeit, Formalisierung, Interpretierbarkeit, Operationalität, Realitätsadäquanz, Einfachheit, Einheitlichkeit, Vollständigkeit, Integrationsqualität und Fruchtbarkeit (vgl. [Zele96], S. 372).

- Konkurrierende Überschneidung: Es wurden Fragmente mit dem gleichen Gegenstandsbereich ausgewählt ohne bestimmten Grund (z. B. aufgrund einer Forderung nach mehreren Perspektiven)
- Unfähigkeit der Anwendung: Die Fragmente können im Projekt nicht angewendet werden (z. B. aufgrund einer unzureichenden Schulung der Projektmitglieder)

Die Qualität von Methoden wird entsprechend mit den Kriterien Vollständigkeit, Konsistenz, Effizienz, Zuverlässigkeit und Anwendbarkeit beschrieben (vgl. [Bri+99], S. 219f bzw. [Bri+98], S. 391f). Der Anhang 13.2 liefert einen Überblick über diesen Ansatz. Nach Meinung von BRINKKEMPER ET AL. können aus diesen Anforderungen nur Richtlinien erzeugt werden, die bei der Entwicklung einer Methode einzuhalten sind. Sie bedürfen weitgehend einer Hinterlegung mit messbaren Kriterien.

## 4.8 Zusammenfassung

Die bisher geschilderten Qualitätsansprüche und Metriken bilden zunächst nur eine lose Sammlung von möglichen Anforderungen. Sie ergibt insgesamt eine unstrukturierte Menge teilweise quantifizierbarer und damit auch messbarer und zum Teil weicher Kriterien. Die Abbildung 21 fasst deshalb die im Verlauf des Abschnitts identifizierten Anforderungen geeignet zusammen.

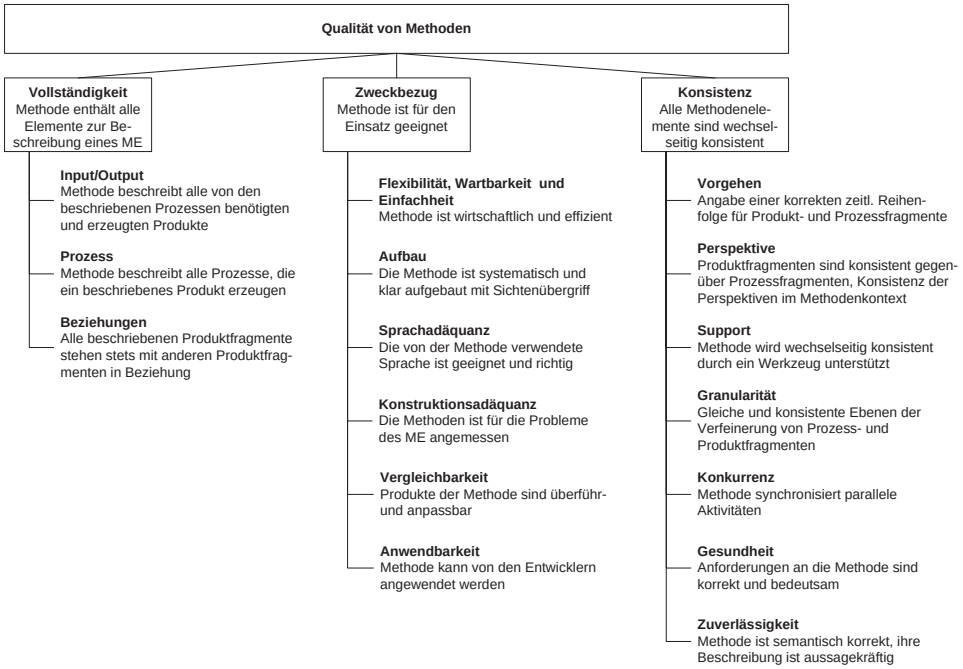



Abbildung 21: Anforderungen an Methoden (zusammengefasst aus dem Abschnitt 4)

Die *Vollständigkeit* und *Konsistenz* der Methoden bezieht sich im Wesentlichen auf die Ausführungen von BRINKKEMPER (vgl. Abschnitt 4.7), die sich vor allem durch die Arbeiten von ZELEWSKI und SINZ belegen lassen. Der Zweckbezug ergibt sich aus Überlegungen zu den GoM (vgl. Abschnitt 4.4).

Für diese Arbeit wurden die dargelegten allgemeinen Anforderungen für das ME gewichtet. Eine entsprechende Übersicht liefert die Tabelle 4. Die vollständigen Ergebnisse des paarweisen Vergleichs sind dem Abschnitt 13.4 im Anhang zu entnehmen. Die entstandene Rangfolge der Anforderungen erhebt nicht den Anspruch auf Situationsunabhängigkeit und unterliegt zudem der subjektiven Beurteilung des Autors. Sie dient zum einen der Demonstration der Technik des paarweisen Vergleichs bei der Gewichtung von Anforderungen an Methoden und zum anderen der Argumentation zur Definition einer neuen Methode des ME aufgrund der Defizite vorhandener Ansätze. Deren Ausmaß und damit auch die Rangfolge der Anforderungen ist letztlich jedoch vernachlässigbar.

| Rang | Anforderung | Relatives Gewicht |
|---|---|---|
| 1 | Zuverlässigkeit | 10,7% |
| 2 | Konstruktionsadäquanz | 10,1% |
| 3 | Sprachadäquanz | 9,8% |
| 4 | Anwendbarkeit | 9,3% |
| 5 | Gesundheit | 9,0% |
| 6 | Aufbau | 7,9% |
| 7 | Vorgehen | 6,9% |
| 8 | Flexibilität, Wartbarkeit und Einfachheit | 6,7% |
| 9 | Vergleichbarkeit | 6,2% |
| 10 | Konkurrenz | 4,6% |
| 11 | Prozess | 4,4% |
| 12 | Perspektive | 4,4% |
| 13 | Granularität | 3,5% |
| 14 | Input/Output | 3,3% |
| 15 | Support | 2,6% |
| 16 | Beziehungen | 0,5% |

Tabelle 4: Zusammenfassung des paarweisen Vergleichs von Methodenanforderungen

# 5 Methoden des Method Engineering

In den vorangegangenen Abschnitten wurde ein Werkzeug erarbeitet, mit dem u. a. eine Bewertung von Methoden der Methodenentwicklung möglich ist. Der folgende Abschnitt wird dieses Werkzeug auf vorhandene Methoden anwenden und deren Schwachpunkte identifizieren. Somit wird die Entwicklung einer eigenen Methode motiviert, die in Teil III vorgestellt wird. Mit dem Ziel der Ausgestaltung eines HoQ werden im folgenden Kapitel 5.1 wahrnehmbare Eigenschaften der Methoden des ME zusammengestellt. Die Anwendung einer Priorisierungsmatrix in diesem Kontext, die zur Erstellung eines HoQ notwendig ist, erfordert eine nominale Qualifizierung der bei den Methoden beobachteten Eigenschaften. Hierzu wird die Skala $\{0;3;9\}$ angewendet, um eine abgestufte Bewertung zu ermöglichen. Das Kapitel 5.2 begründet die Auswahl der in Kapitel 5.3 vorgestellten Methoden des Method Engineering. Ein zusammenfassender Vergleich erfolgt in Kapitel 5.4.

## 5.1 Merkmale von Methoden des ME

### 5.1.1 Reifegrad und Validierung

Die Methode muss eine gewisse Reife erreicht haben, bevor sie eingesetzt werden kann. Um diesen Reifegrad messbar zu gestalten, werden die folgenden im Text hervorgehobenen Kriterien näher untersucht. Von der Methode sollten nach Möglichkeit bereits mehrere **Versionen** existieren. Da eine Versionierung in der Regel nicht explizit erfolgt, werden bereits wesentliche Veränderungen nach der Erstveröffentlichung als Weiterentwicklung betrachtet.

- 0: Nach der Erstveröffentlichung erfolgt keine Änderung mehr an der Methode.
- 3: Es existieren Veröffentlichungen über einen Zeitraum von 5 Jahren, die auf eine Weiterentwicklung schließen lassen. Zudem werden Einflüsse anderer Methoden deutlich.
- 9: Die Methode wird explizit versioniert und besitzt eine nachvollziehbare Versionsfamilie.

Durch das Vorhandensein von **Werkzeugen** zur Methode wird eine gewisse Praxisrelevanz des Ansatzes unterstellt, da im Allgemeinen die Erstellung und Pflege von Methoden computergestützt erfolgen sollte.

- 0: Es existiert kein Werkzeug.
- 3: Es existieren Prototypen, die zumindest einen Nachweis der Funktionstüchtigkeit der Methode liefern können.
- 9: Es existiert ein marktreifes Produkt, dessen Entwicklung noch nicht eingestellt wurde.

Da die Reife einer Methode durch eine wissenschaftliche Fundierung des Ansatzes positiv beeinflusst wird, wird die Art und Anzahl der wissenschaftlichen **Veröffentlichungen** bewertet.

0: Es existieren keine Fachbücher oder Dissertationen zu diesem Thema.
3: Es existieren zahlreiche Tagungsbeiträge über einen Zeitraum von mindestens 5 Jahren.
9: Es existiert mindestens eine Dissertationsschrift oder ein vergleichbares Fachbuch zum Ansatz. In der Regel erscheinen damit auch begleitend Tagungsbeiträge und Veröffentlichungen in Fachzeitschriften.

Die **interne Validierung** der Methode erfolgt durch deren Entwickler. Eine fachgruppenübergreifende Konstellation von Praktikern und Wissenschaftlern ist von Vorteil.

0: Es existieren nur Publikationen eines Autors.
3: Es existieren Publikationen mehrer Personen, die sich aber einer Institution bzw. Fachrichtung zuordnen lassen (z. B. ein Lehrstuhl).
9: Der Ansatz wurde von Personen unterschiedlicher Fachrichtungen entwickelt.

Ohne **externe Validierung** ist eine Bewertung der Methode von außen nicht möglich. Nicht an der Entwicklung beteiligte Personen müssen Zugang zu den Grundlagen und Zusammenhängen innerhalb der Methode erhalten.

0: Es existieren keine frei verfügbaren Veröffentlichungen.
3: Auf Anfrage sind Materialien erhältlich.
9: Die Methode wird transparent für Außenstehende entwickelt. Beispielsweise wird hier eine umfassende Offenlegung des Ansatzes im Internet positiv bewertet.

Ohne deren inhaltliche Qualität zu bewerten, weist bereits eine **Dokumentation** des Ansatzes auf eine gewisse Reife hin.

0: Es existieren weder zum möglicherweise vorhandenen Werkzeug noch zur Methode selbst Dokumente, die diese beschreiben.
3: Die Methode ist dokumentiert, eine Anwendung erfordert jedoch zahlreiche Schritte, die nicht beschrieben sind oder z. B. nur durch Schulungen vermittelt werden.
9: Es existiert eine nachvollziehbare, strukturierte und mit Beispielen belegte Dokumentation des Ansatzes.

Ansätze, die in mehr als zwei Punkten mit 0 bewertet wurden, sind im Rahmen dieser Arbeit nicht näher untersucht worden, da belegbare Aussagen zur Methode nicht möglich waren.

### 5.1.2 Eigenschaften der Meta-Modellsprache

Methoden müssen ihre Produkte definieren. Für Methoden des ME können deshalb die im Folgenden näher erläuterten Merkmale der Modellierungssprache bestimmt werden. Die **Ontologische Klarheit** gewährleistet die Eindeutigkeit der Begriffe einer Methode. Sie basiert auf der im Abschnitt 4.6 erarbeiteten ontologischen Qualifizierung von Modellierungssprachen.

0: Die Sprache ist durch Konstruktredudanz und -überladung gekennzeichnet. Für in dieser Sprache erstellte Modelle kann deren Inhalt nicht eindeutig sein.

3: Die ontologische Redundanz wird kritischer bewertet als die Überladung, da in Methoden vielfach Spezialisierungen der Konstrukte aus Tabelle 3 auftreten werden. Entsprechend werden Sprachen, die Überladungen enthalten aber keine Redundanzen mit diesem Wert beschrieben.

9: Ein Konstruktüberschuss wird nicht nachteilig bewertet, Konstruktüberladung und -redundanz müssen jedoch ausgeschlossen werden, um die Sprache einerseits als flexibel und andererseits als präzise für den Einsatz beim ME zu bezeichnen.

Die Bewertung der **Ontologischen Vollständigkeit** anhand der Tabelle 3 impliziert, dass mit ihr nicht nur alle Begriffe einer Modellierungssprache des ME erfasst wurden, sondern dass diese Zusammenstellung auch abschließend erfolgte. Um die Bewertung zu objektivieren, wird auf eine streng binäre Skala (vollständig oder nicht) verzichtet und an ihre Stelle ebenfalls eine dreistufige angesetzt.[80]

0: Die Sprache ist sowohl bei den Konzepten des statischen Teils (Modellelement, Eigenschaft, Konzept etc.) als auch bei denen des dynamischen Teils (Strategie, Transformation etc.) unvollständig. Insgesamt werden weniger als 50 % der Konstrukte abgebildet.

3: Die Sprache deckt wenigstens den statischen Teil der Konstrukte ab und es werden zwischen 50 % und 80 % der Konstrukte abgedeckt.

9: Die Sprache bildet sowohl den statischen als auch den dynamischen Teil der Konstrukte ab und es werden insgesamt mehr als 80 % der Konstrukte abgedeckt (Es fehlen also maximal 3 Konstrukte).

Die **Formalität** einer Sprache wird entsprechend der Kriterien aus Abschnitt 2.5.1 bestimmt. Einer formalen Sprache wird dabei die höchste Präzision bei der Anwendung unterstellt.

0: Informale Sprache

3: Semiformale Sprache

9: Formale Sprache

[80] Bei einer streng binären Skala aus den Werten 0 und 9 hätte zudem keiner der Ansätze aus Abschnitt 5.3 die „Note“ 9 erhalten, die Sprachen der Methoden wären nicht vergleichbar.

Mit dem Merkmal der ontologischen Qualifizierung lässt sich bereits der Sprachumfang der Modellierungssprache näher beschreiben. Die zugehörigen Merkmale bieten jedoch nur einen groben Anhaltspunkt über die Bestandteile der Sprache. Mit den folgenden Kriterien wird überprüft, ob die Abbildung bestimmter Methodenbestandteile möglich ist. Die Merkmale können dabei entweder erfüllt, nur in Ansätzen oder gar nicht erfüllt werden. Die Bewertung erfolgt entsprechend mit den Werten 0, 3 und 9. Es wird die Abbildung von **Konzepten und Beziehungen**, **Darstellungstechniken**,[81] **Organisatorischen Rahmenbedingungen**, **Abstraktionsebenen**,[82] **Produkt- und Prozessregeln**, **Dokumentationen**, **Sichten** und **Vorgehensmodellen** unterschieden.

### 5.1.3 Eigenschaften des Modellierungsprozesses

Zur Beurteilung des durch die Methode beschriebenen Prozesses wird untersucht, ob eventuell **Wiederverwendungsansätze** integriert werden. Im Mittelpunkt steht nicht die Anzahl der beschriebenen Ansätze, sondern die Qualität ihrer Integration in den Prozess des ME.

0: Die Methode beschreibt keine Form der Wiederverwendung von Methoden oder deren Fragmenten.
3: In den Beschreibungen zur Methode wird auf diese Ansätze Bezug genommen, deren Anwendung wird jedoch nicht explizit dargestellt und in den Prozess des ME integriert.
9: Es existiert zur Methode die Beschreibung von Wiederverwendungsansätzen. Deren Anwendung ist in den Prozess des ME integriert.

Die Methode des ME beschreibt Prozesse, die stets durch eine Organisation umgesetzt werden müssen. Die Festlegung menschlicher und maschineller Aufgabenträger erfolgt mit Hilfe eines **Organisationsmodells**.

0: Die Methode enthält keinerlei organisatorische Rahmenbedingungen.
3: Die Methode enthält Richtlinien, aus denen sich Rollen oder Stellen im Projekt ableiten lassen.
9: Teil der Methode ist ein detailliertes Organisationsmodell, welches beteiligte Rollen und deren Aufgabentypen innerhalb des ME beschreibt.

Die **Integration der Modellierungssprache** in das Vorgehen ermöglicht eine genaue Festlegung der Ergebnistypen. Fehlt dieser Zusammenhang, so stehen Vorgehen und Produkte des

[81] Ansätze, bei denen zur Abbildung von Darstellungsmitteln auf ein zugehöriges Werkzeug verwiesen wurde, wurden mit 0 bewertet, da Werkzeuge wie *Microsoft Visio* sich faktisch jedem Ansatz zuordnen lassen und entsprechend eine Beschreibung von Darstellungstechniken zulassen. Die Wertung mit 9 erfolgt nur, wenn innerhalb der *Sprache* die Definition von Darstellungsmitteln möglich ist.

[82] Mit 9 wurden Sprachen bewertet, die mindestens auf einer Typebene die Methode und auf einer Instanzenebene deren Gegenstandsbereich beschreiben.

ME losgelöst nebeneinander und die Ableitung möglicher Meilensteine im ME-Projekt ist nicht möglich.

0: Der Zusammenhang zwischen Modellierungssprache und Vorgehensbeschreibung ist nicht oder nur grob beschrieben.
3: Die Vorgehensbeschreibung nimmt direkt Bezug auf die Modellierungssprache und damit auf die Ergebnistypen einzelner Aufgabentypen.
9: Zusätzlich existiert eine formale oder semiformale Vorgehensbeschreibung, die entsprechend der Zustände der Ergebnisse zu erledigende Aufgabentypen definiert.

Die **Modularität** des Vorgehens bestimmt die Möglichkeit, einzelne Phasen für ME-Projekte zu definieren. Die Abgeschlossenheit solcher Module gestattet deren Austausch durch alternative Techniken bei der Methodenentwicklung, die bei der Definition der Methode keine Beachtung fanden. Eine geringe Modularität einzelner Arbeitsgänge erschwert zudem die Übersicht über das Vorgehen. Die Modularität kann durch die Definition entsprechender Vorgehens- und Phasenmodelle positiv beeinflusst werden.

0: Es wird ein durchgängiges Verfahren beschrieben.
3: Im Vorgehen sind einzelne Phasen erkennbar, deren Abgrenzung erfolgt jedoch nicht in Bezug auf die Ergebnistypen der Phasen und deren Eigenschaften.
9: Es existieren klar definierte Phasen, deren Ein- und Austritt durch die Beschreibung bestimmter Produktmerkmale charakterisiert wird. Neben den detailliert beschriebenen Techniken existiert eine Beschreibung der Anforderungen an diese, um sie möglicherweise austauschen zu können.

Methoden werden aufgrund bestimmter Anforderungen entwickelt. Die Planung, Überwachung und Steuerung von deren Einhaltung ist Aufgabe der **Qualitätssicherung** (QS).

0: Es wird mit der Methode des ME keine Planung oder Überwachung der Qualität der zu entwickelnden Methode beschrieben.
3: Die QS umfasst mindestens die Erhebung von Anforderungen an die Methode sowie Techniken zu deren Überprüfung.
9: Die beschriebene QS enthält zusätzlich eine Beschreibung konstruktiver Techniken der Planung und Steuerung der Qualität der zu entwickelnden Methode.

**Vorgehens- und Phasenmodelle** dienen der Kommunikation des Vorgehens. Aus entsprechend guten Modellen lässt sich eine Projektplanung ableiten. Mit zunehmender Detaillierung und einer formalen Beschreibung steigt die Möglichkeit, einzelne Aufgabentypen zu automatisieren.

0: Die Beschreibung erfolgt natürlichsprachlich und damit informal.
3: Auf fachlicher Ebene existieren formale oder semiformale Vorgehensmodelle.
9: Das Vorgehen wird unter Angabe von Zielen und Strategien zusätzlich derart beschrieben, dass zum einen eine hohe Modularität gegeben ist und zum anderen ein hoher Grad der Detaillierung vorliegt.

Das **Konfigurationsmanagement** umfasst gemäß Abschnitt 4.5 die Teile Konfigurationsidentifikation, -buchführung, überwachung und -audit.

0: Mit dem Vorgehen wird keiner der Teile des KM ausgestaltet.
3: Im Vorgehen erfolgt (meist mit Hilfe eines Werkzeuges) eine Identifikation von Konfigurationen und deren Buchführung.
9: Neben der Identifikation und Buchführung für Konfigurationen werden Techniken beschrieben, die deren Überwachung und Audits zur Planung und Steuerung von Änderungen an der Methode ermöglichen.

Zusammenfassend ergeben sich die in Abbildung 22 dargestellten messbaren Kriterien einer Methode des ME. Zwischen den Anforderungen aus Abschnitt 4.8 und den Merkmalen dieses Abschnitts wurden die in der Interdependenzmatrix des Anhangs 13.5 auf der Skala {0;1;3;9} in Abstufung ihrer Stärke dokumentierten Zusammenhänge identifiziert.[83]

## 5.2 Auswahl der bewerteten Methoden

Für die Auswahl der in diesem Abschnitt näher untersuchten Methoden des Method Engineering wurden folgende Kriterien angesetzt:

- Es existieren Werkzeuge, die diese Methoden unterstützen, oder diese befinden sich in Entwicklung. Mit diesem Kriterium wird eine gewisse Praxisrelevanz des gewählten Ansatzes unterstellt (Merkmal *Werkzeug* größer 0 bewertet).
- Theoretische Ansätze, die in den letzten Jahren nicht mehr weiter verfolgt wurden, werden nicht betrachtet.
- Methoden der Systementwicklung, die zwar im Grundsatz dazu geeignet sind, Methoden zu entwickeln, für die dem Autoren jedoch bisher keine Anwendungsfälle auf dem Gebiet der Methodenentwicklung bekannt sind, werden nicht betrachtet.[84]

[83] Die in den folgenden Abschnitten 5.3.1 bis 5.3.5 beschriebenen Methoden des ME dienen aus Sicht dieses Abschnittes im Anhang 13.5 als praktisches Beispiel für die Anwendung des vorgestellten Kriterienkataloges.
[84] Methoden können als Informationssystem bzw. als Teil eines solchen betrachtet werden. In diesem Sinne wäre z. B. eine Untersuchung der Ansätze von RUMBAUGH ET AL. (vgl. [Rum+91]) oder GRAHAM ET AL. (vgl. [Gra+97]) denkbar.

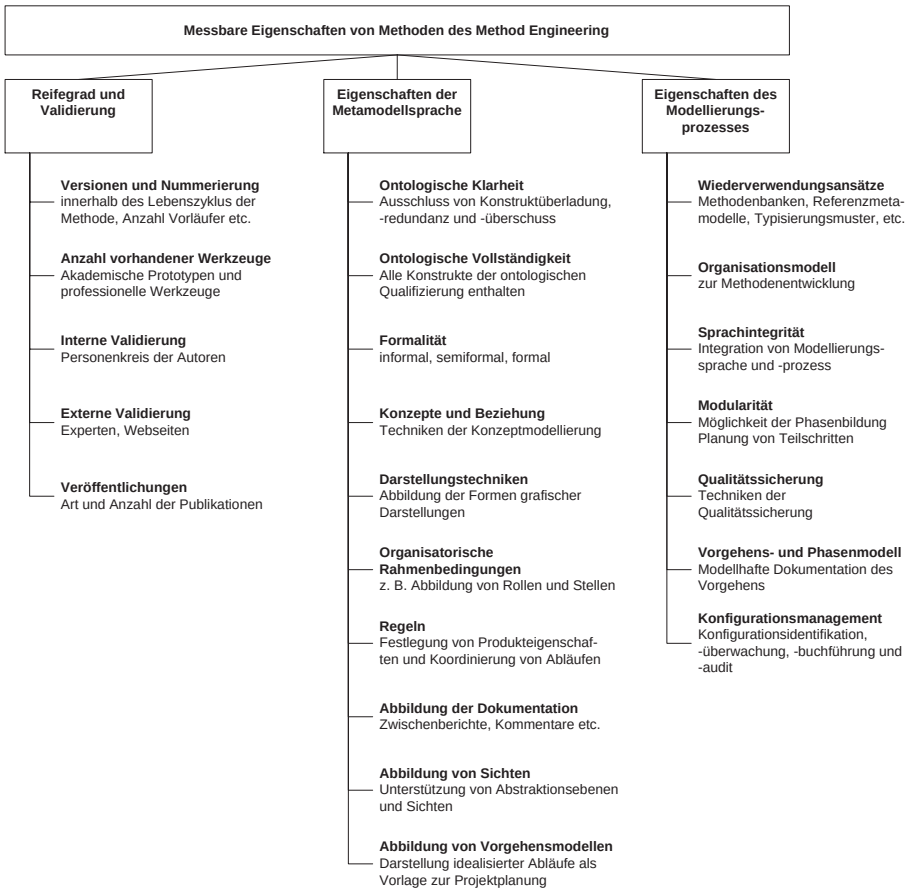



Abbildung 22: Messbare Eigenschaften der Methoden des ME

## 5.3 Methoden des ME

Dieser Abschnitt wird die Ansätze des ME, die die Kriterien aus Abschnitt 5.2 erfüllen, vorstellen. Die Gliederung der jeweiligen Beschreibung orientiert sich an der Einteilung der Merkmale aus Abschnitt 5.1 und beantwortet die Fragestellungen von HESS und BRECHT (siehe dazu Tabelle 2 in Abschnitt 4.3). Die jeweils verwendeten Metamodelle werden in einer UML-Notation im Überblick dargestellt. Oftmals muss aber dabei eine Einschränkung auf den Kern der jeweiligen Methode vorgenommen werden.

### 5.3.1 MEMO

FRANK beschreibt das Multi Perspective Enterprise Modelling (MEMO als eine Methode bzw. Framework zur Erstellung von Unternehmensmodellen (vgl. [Fran98b], S. 4).[85] Gegenstand des MEMO ist die Abbildung von Modellierungssprachen für Informationssysteme. Anwenden lassen sich die mit MEMO spezifizierten Metamodelle bei der Modellierung von Informations-

[85] Diese Aussage korrespondiert nicht mit dem Methodenbegriff aus Abschnitt 3.1.

systemen. Der Schwerpunkt von MEMO liegt dabei eher in der Ebene der Systementwicklung, weniger im Bereich des ME. Als Teildisziplin des ME wird eine Integration von Modellen, die eine unterschiedliche Sprache verwenden, ermöglicht. Beispielsweise können Terminologien von Softwareingenieuren und Unternehmensberatern integriert werden (vgl. [Fran98b], S. 5). Erstellt wurden bereits die Object Modelling Language (MEMO-OML), eine Process Modelling Language (MEMO-PML) und eine Organisation Modelling Language (MEMO-OrgML). Über einen Einsatz im nicht wissenschaftlichen Umfeld liegen keine Informationen vor.

### Reifegrad und Validierung

Der Ansatz wird seit 1998 nicht mehr weiterentwickelt. Die bis dahin vorliegenden Ergebnisse wurden nicht versioniert, was eine Verfolgung der Entwicklung des Ansatzes erschwert. Es existieren zahlreiche Veröffentlichungen in Form von Arbeitsberichten, zudem entstanden seit Mitte der 1990er Jahre Tagungsbeiträge. Die interne Validierung erfolgte primär durch FRANK als Entwickler des Ansatzes. Die Materialien sind über das Internet verfügbar. Das einzige Werkzeug für die Arbeit mit MEMO ist das MEMO Center (vgl. [Fran02]), dessen Object Model Designer (OMD) in der derzeitigen Version keine Metamodellierung mit der spezifizierten MEMO-OML zulässt.

### Eigenschaften der Metamodellsprache

Der Ansatz konzentriert sich auf die Beschreibung von Konzepten und deren Beziehungen. Dazu entstand mit der MEMO Object Modelling Language (MEMO-OML) eine der UML ähnliche Sprache. Die abstrakte Syntax des MEMO wird ebenfalls in der MEMO-OML definiert (vgl. [Fran98c]). Diese wiederum wird in [Fran98b] als Instanz des MEMO Meta-Metamodells beschrieben. Die Abbildung 23 stellt das MEMO Meta-Metamodell in der UML dar.

Damit kann die MEMO-OML als formale Sprache bezeichnet werden. Zur ontologischen Vollständigkeit fehlen vor allem Ausdrucksmittel für Repräsentation, Transformation, Modellumgebung, Teilmodell, Ziel und Strategie. Darstellungstechniken, organisatorische Rahmenbedingungen und Vorgehensmodelle lassen sich nicht abbilden.

### Eigenschaften des Modellierungsprozesses

In [Fran98a] schlägt FRANK ein Vorgehen bei der Evaluation von Modellierungssprachen vor. Für die Entwicklung vollständiger Methoden mit MEMO existiert keine Vorgehensbeschreibung. Entsprechend werden mit MEMO keine Rollen oder Beziehungen der organisatorischen Einbettung des ME beschrieben. Für die Evaluierung von Modellierungssprachen werden die Rollen *Sprachdesigner*, *Linguist*, *Domainenexperte*, *Forscher*, *durchschnittlicher Benutzer* und

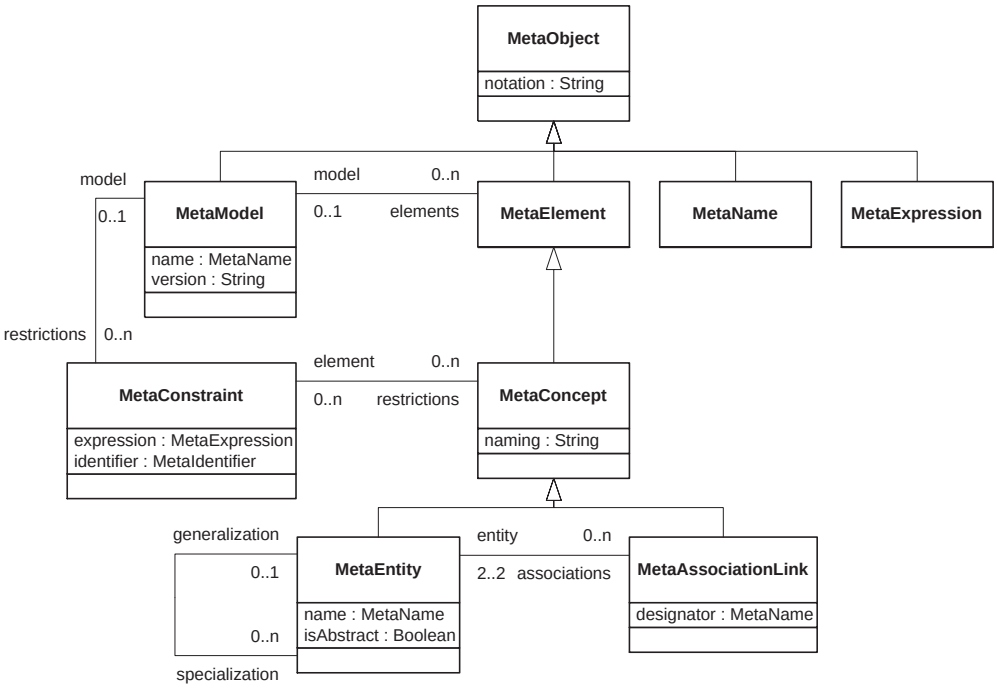



Abbildung 23: Vereinfachtes Metamodell der MEMO-OML (i. A. a. [Fran98b], S. 8)

*Experte* vorgestellt, ohne die Rollen inhaltlich näher zu beschreiben. Bei der Evaluierung werden einer Rolle jeweils eine Anzahl Perspektiven zugeordnet, die diese auf Unterstützung von Aufgaben durch die Sprache besitzen (vgl. [Fran98b], S. 14). Ein Konfigurationsmanagement wird von diesem Ansatz nicht unterstützt.

### 5.3.2 KOGGE/JKogge

Der an der Universität Koblenz am Institut für Softwaretechnologie entwickelte Ansatz zur Methodenentwicklung wird in dem Projekt JKogge zusammengefasst. Dessen Ziel ist die Erstellung von Tools für visuelle Sprachen (vgl. [Uhe00], S. 3). Die Entwicklung des Werkzeuges für diese Aufgabe basiert auf den formal in der Spezifikationssprache Z beschriebenen Ansätzen der erweiterten Entity-Relationship-Diagramme (EER-Diagramme) und der Graph Specification Language (GRAL) zur Beschreibung von gerichteten Graphen (TGraphen).

#### Reifegrad und Validierung

Über den Ansatz existieren seit Anfang der 1990er Jahre zahlreiche Veröffentlichungen in Form von Dissertationen, Tagungsbeiträgen und Arbeitsberichten. Entsprechend viele Autoren sind an der Entwicklung beteiligt, hervorzuheben sind die Beiträge von EBERT, WINTER und FRANZKE. Das Projekt wird zudem ausführlich im Internet beschrieben. Das Werkzeug JKog-

ge ist eine Weiterentwicklung des Meta-CASE Tools KOGGE im Prototypen-Stadium, welches eine im Internet verteilte Systementwicklung unterstützen soll (vgl. [Ebe+98], S. 1).

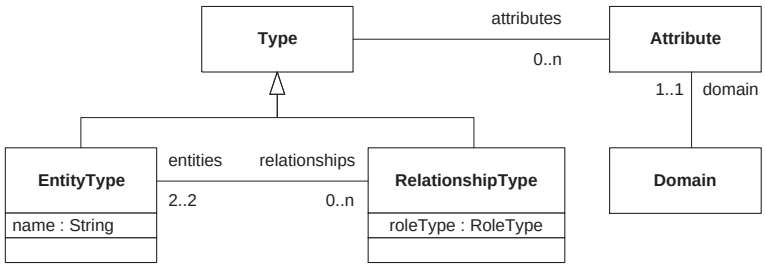



Abbildung 24: Vereinfachtes Metamodell der EER-Diagramme (i. A. a. [Ebe+99], S. 6)

### Eigenschaften der Metamodellsprache

Ähnlich wie beim MEMO-Ansatz liegt auch hier der Schwerpunkt auf der Beschreibung von Konzepten und deren Beziehungen. Die hierfür verwendeten EER-Diagramme bestimmen entsprechend die ontologischen Eigenschaften der Metamodellsprache. Es fehlen vor allem Ausdrucksmittel für Transformationen, Modellumgebungen, Teilmodelle, Ziele und Strategien. Im Gegensatz zum MEMO-Ansatz erfolgt explizit eine Beschreibung von Repräsentationen.

### Eigenschaften des Modellierungsprozesses

Ein vollständiger Prozess der Methodendefinition wird nicht beschrieben. Für die Erstellung von EER-Diagrammen unter Verwendung des Werkzeuges KOGGE bei der Meta-Modellierung existieren Vorgehensbeschreibungen in Textform, zumeist unter Angabe von Beispielen. Mit der Definition eines Referenz-Metaschemas[86] durch WINTER (siehe dazu [Wint00]) wurde u. a. ein Wiederverwendungsansatz geschaffen, der eine fallstudienartige Beschreibung der Definition der Konzepte von ARIS und UML auf Basis dieses Schemas enthält. Ein Konfigurationsmanagement wird nicht beschrieben.

## 5.3.3 MetaEdit+

MetaEdit+ ist ein kommerziell vertriebenes Werkzeug der Firma MetaCase Consulting. Der zugrunde liegende Ansatz und ein erster Prototyp wurde am Department of Computer Science der Universität von Jyväskyla in Finnland entwickelt. Er ist vor allem durch seinen starken Bezug zum CASE gekennzeichnet und konzentriert sich wie beispielsweise auch der MEMO- und KOGGE/JKogge-Ansatz auf die Konzepte und Beziehungen der Methode.

[86] Der Abschnitt 7.2 wird sich mit diesem Ansatz ausführlich beschäftigen.

### Reifegrad und Validierung

Mit der Initiierung des Projektes Syti im Jahr 1988 erschienen vornehmlich von SMOLANDER, TOLVANEN und LYYTINEN zahlreiche Veröffentlichungen auf Tagungen sowie Publikationen im Internet. Daneben wird das Werkzeug regelmäßig auf Messen präsentiert sowie in Broschüren beschrieben.

### Eigenschaften der Metamodellsprache

Aufgrund des eng gefassten Methodenbegriffes werden im Ansatz von SMOLANDER ET AL. vor allem Konzepte und deren Beziehungen und Attribute beschrieben. Diese lassen sich mit formal definierten Object-Property-Role-Relationship-Diagrammen (OPRR-Diagramme) abbilden (siehe dazu [Smol91]). Entsprechend sind Sprachdefekte bei der Abbildung sämtlicher dynamischer Aspekte einer Methode festzustellen.

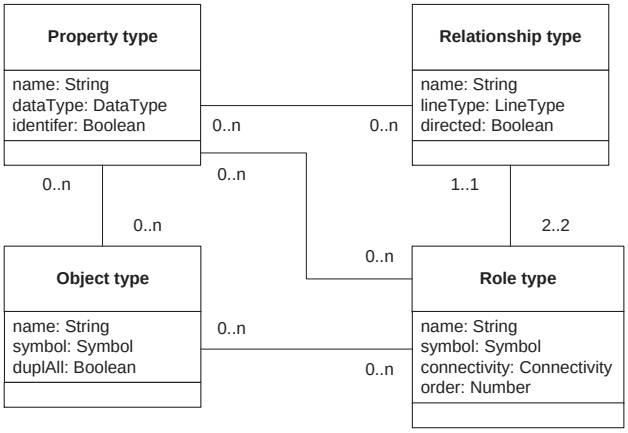



Abbildung 25: Vereinfachtes Metamodell der OPRR-Diagramme (i. A. a. [Smo+91], S. 182)

### Eigenschaften des Modellierungsprozesses

Der von TOLVANEN ET AL. beschriebene Prozess der Methoden-Modellierung orientiert sich am klassischen Vorgehen bei der Datenmodellierung. Nach der Identifikation von Objekttypen und deren Beziehungen und Eigenschaften erfolgt eine Definition der Repräsentationstechniken der Methode, die mit dem Verweis auf eine notwendige Werkzeugunterstützung nicht näher erläutert wird (vgl. [ToLy93]). Ebenso wenig wird eine organisatorische Einbettung beschrieben. Als Wiederverwendungsansatz wird auf die Method-Base von BRINKKEMPER und HARMSEN verwiesen (vgl. u. a. [Har+96], S. 171f). Inwieweit sich diese jedoch auf die Struktur oder Funktionsweise des Werkzeuges auswirkt, ist nicht ersichtlich. Ein Konfigurationsmanagement wird weder in der Theorie noch im Werkzeug abgebildet.

### 5.3.4 ADONIS

ADONIS ist der Name eines Werkzeugs der Business Objectives Consulting (BOC) GmbH zum Management von Geschäftsprozessen. Es unterstützt dabei im Wesentlichen die Erhebung, Abbildung, Analyse, Simulation, Evaluation und Transformation von Prozessmodellen eines Unternehmens. Im Rahmen einer Beratungsleistung wird von der BOC GmbH u. a. auch die Konzeption von Methoden angeboten (vgl. [BOC 02]). Für die Bewertung des Ansatzes ist von Bedeutung, dass die mit dem Werkzeug modellierten Prozesse auf der Objektebene liegen. Es ist entsprechend kein Hilfsmittel bei der Definition von Methoden des Methodenengineering.[87] Entsprechend der Ausführungen zu Beginn des Kapitels werden für diesen Ansatz nicht die theoretischen Möglichkeiten des Werkzeuges auf der Objektebene bewertet. Der Schwerpunkt liegt auch hier auf der Beurteilung des Ansatzes des ME, in den das Werkzeug eingebettet ist.

#### Reifegrad und Validierung

Die Validierung des Ansatzes erfolgt vor allem durch den praktischen Einsatz des Werkzeuges in Projekten. Daneben existieren einige wenige wissenschaftliche Veröffentlichungen von KARAGIANNIS, JUNGINGER und KÜHN. Als Referenzkunden für das Werkzeug werden zahlreiche Banken, Versicherungen, Beratungsgesellschaften und öffentliche Einrichtungen aufgeführt.

#### Eigenschaften der Metamodellsprache

Die Modellierung bildet die Kernkomponente des Werkzeuges. Das verwendete Konzept zur Metamodellierung basiert auf Klassen und Beziehungsklassen, für die sich jeweils Attribute definieren lassen. Deren grafische Darstellung wird als Klassenattribut abgelegt. Das Metamodell enthält explizit die Definition von Sichten und Entwurfsmustern, welche als Metamodellausschnitte definiert werden. Anders als in den anderen Ansätzen werden Metamodelle zur Modellierung auf Objektebene nicht als Instanzen aufgefasst, sondern deren Klassen und Beziehungstypen erben vom beschriebenen Modell. Durch diese Tatsache wird die Problematik der flachen Instanziierung umgangen.

Wird der Prozess der Methodenanwendung selbst als Geschäftsprozess des Unternehmens verstanden, lässt er sich mit dem Werkzeug abbilden, simulieren und Workflow Management Systeme exportieren. Dieses Vorgehen wird jedoch nicht näher beschrieben.

#### Eigenschaften des Modellierungsprozesses

Sprachbasierte Metamodelle werden in einer *method library* unter Nutzung von Entwurfsmustern abgelegt (vgl. [KüJu99]). Die Arbeit mit dieser während des ME wird nicht näher spe-

[87] Wenn auch theoretisch denkbar, ist ein solcher Einsatz nicht vorgesehen.

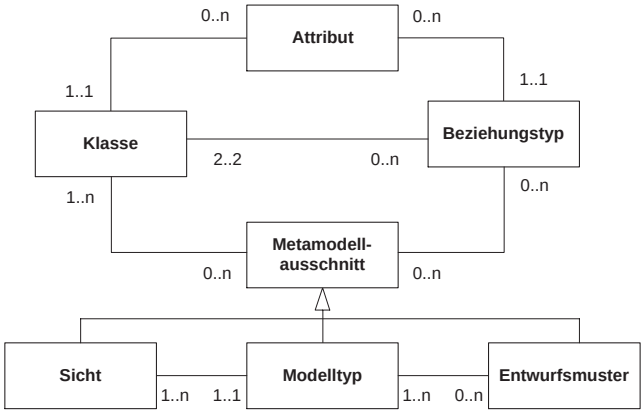



Abbildung 26: Vereinfachtes ADONIS-Metamodell (i. A. a. [Jun+00], S. 395)

zifiziert. Ein Konfigurationsmanagement wird auf Objektebene im Werkzeug ansatzweise zur Verfügung gestellt. Innerhalb des Tools wird im Rahmen einer Konfigurationsbuchführung eine Verwaltung mehrerer Modellversionen angeboten. Für das gesamte Modell können Informationen über den Bearbeiter, eine Versionsnummer und zeitliche Daten abgelegt werden. Modelle können zudem verglichen werden, wobei Informationen über Gemeinsamkeiten und Unterschiede von Modellelementen angezeigt werden können. Eine Zusammenführung oder der Austausch von Modellbestandteilen ist nicht möglich. Organisatorische Rahmenbedingungen, eine Konfigurationsüberwachung und ein Konfigurationsaudit werden nicht beschrieben.

### 5.3.5 ViewPoints

Die Entwicklung von großen und komplexen Softwaresystemen erfordert nach Meinung von NUSEIBEH den Einsatz verschiedener Methoden[88] und deren Integration (vgl. [Nuse94], S. 35). Zu diesem Zweck werden **ViewPoints** definiert, welche jeweils einen Teil des Wissens beschreiben, das bei der Softwareentwicklung notwendig ist. Ziel ist die Integration von Werkzeugen der Softwareentwicklung, die nicht ohne eine Integration der jeweils unterstützten Methode stattfinden kann.

#### Reifegrad und Validierung

Veröffentlichungen zu diesem Ansatz sind seit 1992 entstanden. Gegenwärtig verlagert sich der Schwerpunkt der Tagungsbeiträge zu diesem Ansatz auf das Gebiet des Requirements Engineering. Eine Nummerierung der Versionen ist nicht erfolgt, der Einsatz der ViewPoints auf dem Gebiet des ME wurde seit 1994 nicht weiterentwickelt, obwohl zahlreiche Veröffentlichungen

[88] Welche hier als Prozeduren und Richtlinien zur Spezifikation (inkl. deren Notation) und Entwicklung von Software bezeichnet werden (vgl. [Nuse94], S. 22).

folgten. Die interne Validierung erfolgt hauptsächlich durch NUSEIBEH, FINKELSTEIN, KRAMER und OPDAHL. Extern werden im Internet sehr umfangreiche Materialien zur Verfügung gestellt. Am Imperial College in London wurde das Werkzeug **The Viewer** entwickelt. Über einen Einsatz im kommerziellen Umfeld liegen keine Publikationen vor.

### Eigenschaften der Metamodellsprache

Im Wesentlichen beschränkt sich der Ansatz auf eine Strukturierung des für eine Systementwicklung notwendigen Wissens. Entsprechend fehlen konkrete Notationen für die Darstellung von Konzepten, Eigenschaften oder Modellelementen, wohingegen ViewPoints selbst sich als Perspektive verstehen.[89] Bei der Spezifikation der ViewPoints bediente sich NUSEIBEH einer Fachsprache und einer Analogiebildung zur Objektorientierung. Eine integrierte Methode besteht aus einer Reihe **Templates** für ViewPoints. Ein ViewPoint beschreibt das (vgl. [Nuse94], S. 36ff):

- Domain Knowledge: Domänenwissen wird mit ViewPoints nicht vollständig abgebildet, sie enthalten jedoch einen textuellen Hinweis auf den Kontext des ViewPoints.
- Representation Knowledge: Dieser Bereich deckt das Wissen um die Notation und Sprache bei der Softwareentwicklung ab.[90] Es wird in Form eines **Style** als Objekt-Relation-Diagramm[91] definiert.
- Development Process Knowledge: Durchzuführende Aktionen werden in **Work Plans** abgelegt und z. B. in Prüfoperationen oder Designoperationen unterteilt.
- Specification Knowledge: Das Wissen um das fertige Produkt enthält z. B. Analyse- und Testergebnisse, Spezifikationen und Metriken. Es wird in der Form von grafischen Spezifikationen (**Specification**) und Historien zu den Arbeitsschritten (**Work Record**) gespeichert.

Die Templates werden von ihren **Owners** während der Methodenanwendung zu ViewPoints des zu entwickelnden Systems instanziiert. Eigentümer können Werkzeuge oder menschliche Benutzer sein. Produkt- und Prozessregeln werden zwischen Templates definiert. Ein Vorgehensmodell entsteht implizit durch die Verknüpfung von WorkPlans. Zur Einbettung in die Organisation wird die Rolle des ViewPoint-Owners definiert.

[89] NUSEIBEH bezeichnet die auf dieser Basis durchgeführte Softwareentwicklung als multiperspektivisch.

[90] *Notation* ist in diesem Zusammenhang eine Menge von Konstrukten zur Wissensrepräsentation und *Sprache* eine Menge von Konstrukten und Regeln, wie diese zusammenhängen.

[91] Eine Unterart der Entity-Relationship-Diagramme, wobei Objekte und Relationen mit typisierten Attributen definiert werden. Den Attributen können Werte zugeordnet werden.

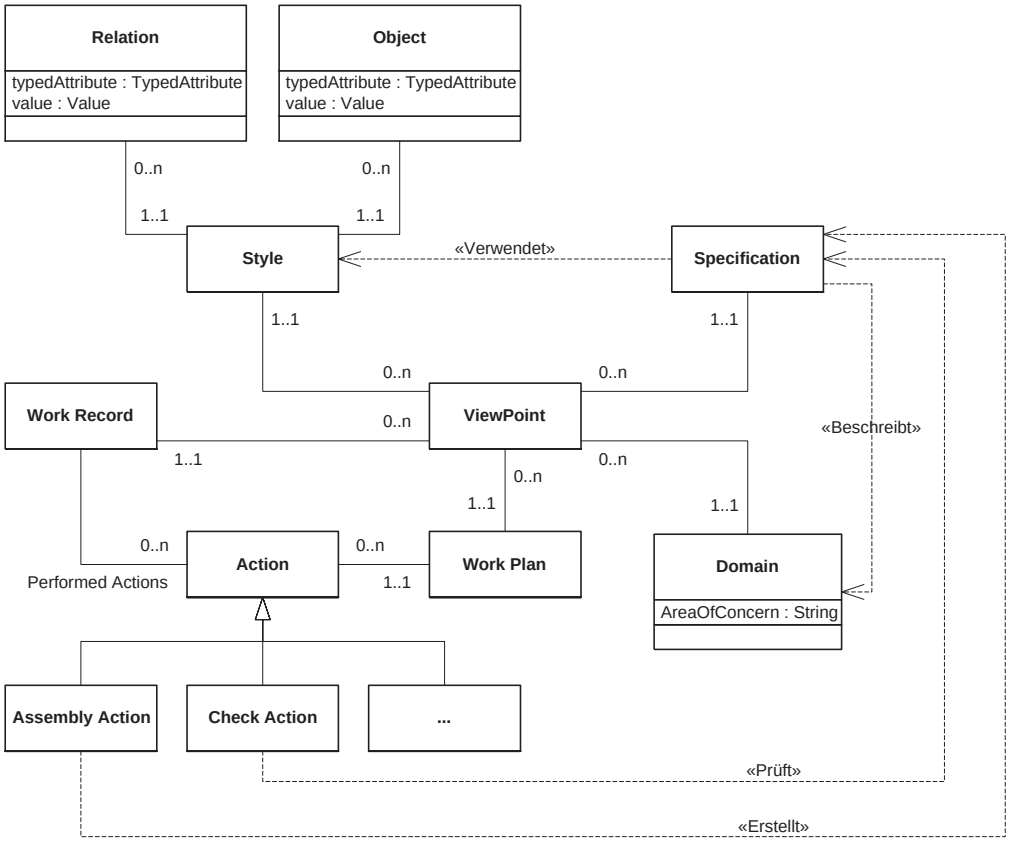



Abbildung 27: Vereinfachtes Metamodell der ViewPoints

## Eigenschaften des Modellierungsprozesses

NUSEIBEH beschreibt die Softwareentwicklung als einen Prozess, in dessen Ergebnis ViewPoints des Softwaresystems entstehen. Aufgabe des Method Engineering ist die Definition von **Templates** für diese ViewPoints. Die Elemente des daraus resultierenden Method Engineering Life Cycle sind (vgl. [Nuse94], S. 100ff.):

1. Erfassung und Strukturierung der Anforderungen der Benutzer und Softwareentwickler: Ergebnisse sind Techniken der Softwareentwicklung und Notationen, die die Zielmethode unterstützen sollen.

2. Konstruktion von ViewPoint Templates: Die Techniken und Notationen werden auf ViewPoint Templates verteilt und entsprechend den Bestandteilen eines ViewPoints zerlegt.

3. Definition der Beziehungen zwischen den Templates zur Erstellung einer Methode: Innerhalb von und zwischen Templates bestehen durch Regeln beschriebene Beziehungen.[92]

4. Benutzung der Methode: Hier erfolgt aus Sicht des ME die Bildung von ViewPoints anhand der Templates. Aus der Anwendung der durch die Templates beschriebenen Methode entstehen ggf. neue Anforderungen an diese.

Der Ablauf wird in Form eines Aktivitätsdiagramms in Abbildung 28 dargestellt. Bei NUSEIBEH selbst finden sich nur informale Beschreibungen.

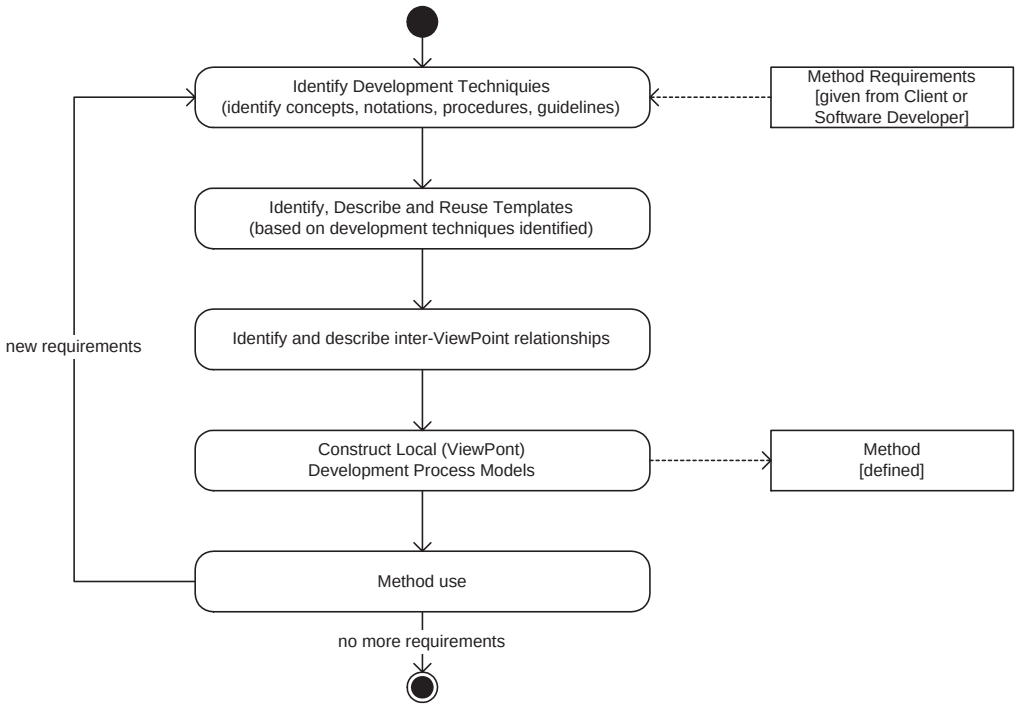



Abbildung 28: Method Engineering Life Cycle (i. A. a. [Nuse94], S. 100)

Als Teilnehmer am Prozess der Softwareentwicklung werden die Rollen **Method Engineer**, **Software Developer** und **Client** identifiziert. Der Method Engineer erstellt Methoden für den Softwareentwickler, indem Anforderungen an Methoden vom Software Developer und dem Client verarbeitet werden. Der Software Developer liefert dem Client auf Basis von dessen Systemanforderungen ein Softwaresystem, welches mit der definierten Methode erstellt wurde.

Weder für ViewPoints noch für deren Templates ist ein Konfigurationsmanagement in die Methode integriert. Die ist insofern von erheblicher Bedeutung für diesen Ansatz, als das NUSEIBEH ein iteratives Vorgehen bei der Template-Definition vorschlägt. Auf der Ebene der In-

[92] Ausführlich werden diese in [Nuse94], S. 110 ff. sowie in [East94], S. 7ff. beschrieben. Die Behandlung der Resultate solcher Konsistenzprüfungen werden in [Nuse94], S. 150 ff. sowie im Überblick in [Fin+93] vorgestellt.

stanzen wird durch die ViewPoints und deren Workrecords eine Konfigurationsbuchführung realisiert.

## 5.4 Vergleich und Motivation

Bei einer zusammenfassenden Betrachtung der existierenden Ansätze aus den vorherigen Abschnitten lässt sich feststellen:

- Die Ansätze des ME beschreiben im Wesentlichen nur die Konzepte und deren Beziehungen und damit nur die produktbezogenen Methodenfragmente.
- Die meisten Beiträge stammen aus dem Anfang und der Mitte der 1990er Jahre. Erst in jüngster Zeit erscheinen wieder verstärkt Beiträge zu diesem Thema, die einen starken Bezug des ME zum Requirements Engineering in der Systementwicklung konstatieren.
- Die wenigen existierenden Werkzeuge zum ME besitzen zumeist einen starken Bezug zum Computer Aided Software Engineering (CASE) und dort insbesondere auf die Entwurfsphase.
- In der Literatur werden mit der *Methode Base* von HARMSEN und dem *Referenz-Metaschema* von WINTER nur zwei Wiederverwendungsansätze beschrieben.
- Der Bezug von Methoden zur Organisation wird in allen Ansätzen weitgehend ignoriert. Es wird lediglich auf Rollen, wie die des Methoden-Ingenieurs, verwiesen.
- Für die Ergebnisse des ME existieren umfassende Vorstellungen über Qualitätsanforderungen und Merkmale von Methoden. Techniken zur Planung, Steuerung und Kontrolle von Qualitätsmerkmalen werden jedoch nicht beschrieben.
- Ein Konfigurationsmanagement wurde in keinem der Ansätze identifiziert, obwohl ein iteratives Vorgehen bei der Methodenentwicklung von fast allen Autoren vorgeschlagen wird.

Die zusammenfassenden Ergebnisse der Methodenbewertung sind der Tabelle 5 zu entnehmen.

| **Kriterium** | Adonis | ViewPoints | MEMO | JKOGGE | MetaEdit+ |
|---|---|---|---|---|---|
| **Reifegrad und Validierung** | | | | | |
| Versionen und Nummerierung | 3 | 0 | 0 | 9 | 9 |
| Anzahl vorhandener Werkzeuge | 9 | 3 | 3 | 3 | 9 |
| Veröffentlichungen | 0 | 9 | 9 | 9 | 9 |
| Interne Validierung | 3 | | 0 | 0 | 9 |
| Externe Validierung | 3 | 9 | 9 | 9 | 9 |
| Dokumentation | 9 | 3 | 3 | 9 | 9 |
| **Eigenschaften der Metamodellsprache** | | | | | |
| Ontologische Klarheit | 9 | 9 | 9 | 9 | 9 |
| Ontologische Vollständigkeit | 3 | 0 | 0 | 0 | 0 |
| Formalität | 3 | 3 | 9 | 9 | 9 |
| Konzepte und Beziehungen | 9 | 0 | 9 | 0 | 9 |
| Darstellungstechniken | 9 | 9 | 0 | 9 | 9 |
| Organisatorische Rahmenbedingungen | 0 | 0 | 0 | 0 | 0 |
| Abstraktionsebenen | 9 | 9 | 9 | 0 | 9 |
| Regeln | 0 | 9 | 9 | 9 | 0 |
| Dokumentation | 0 | 9 | 0 | 0 | 0 |
| Sichten | 9 | 9 | 0 | 0 | 0 |
| Vorgehensmodelle | 0 | 3 | 0 | 0 | 0 |
| **Eigenschaften des Modellierungsprozesses** | | | | | |
| Wiederverwendungsansätze | 0 | 0 | 0 | 3 | 3 |
| Organisationsmodell | 0 | 3 | 0 | 0 | 0 |
| Integration der Modellierungssprache | 0 | 3 | 0 | 0 | 9 |
| Modularität | 0 | 0 | 0 | 0 | 0 |
| Qualitätssicherung | 0 | 0 | 0 | 0 | 0 |
| Vorgehens- und Phasenmodell | 3 | 3 | 3 | 0 | 3 |
| Konfigurationsmanagement | 0 | 0 | 0 | 0 | 0 |

Tabelle 5: Vergleich der Methoden des ME

Die festgestellten Defizite führen zu der Forderung, eine „wirkliche“ Methode für das ME zu definieren. Die Anforderungen an diese Methode wurden nicht empirisch erhoben, vielmehr leiten sie sich aus den vorangestellten Abschnitten ab. Es wird also im Folgenden durch eine Ausgestaltung der Merkmale von Methoden des ME eine Erfüllung der allgemeinen und situationsunabhängigen Anforderungen aus dem Abschnitt 4 angestrebt.

# Teil III
# Die $E^3$-Methode des Method Engineering

Auf Basis der in Teil II erarbeiteten Anforderungen an Methoden und der im Abschnitt 5.4 dargestellten Schwächen bisheriger Ansätze wird in diesem Teil die **E$^3$-Methode des Method Engineering** vorgestellt. Deren Metamodell wird in Kapitel 6 eingeführt. Die verwendeten Konzepte sind Gegenstand der Wiederverwendungsansätze aus Abschnitt 7 und werden in Abschnitt 8 in ein geeignetes Vorgehensmodell integriert. Das Konfigurationsmanagement aus Abschnitt 9 eignet sich als Grundlage des ME und kann gleichzeitig in Projekten Verwendung finden, in denen allgemein eine Informationsmodellierung Anwendung findet. Mit dem Ziel der zusammenfassenden Darstellung der Ergebnisse dieser Arbeit wird der Abschnitt 10 die E$^3$-Methode auf die gleiche Art darstellen und bewerten, wie dies mit den alternativen Ansätzen des ME bereits im Abschnitt 5 des vorigen Teils der Arbeit erfolgte.

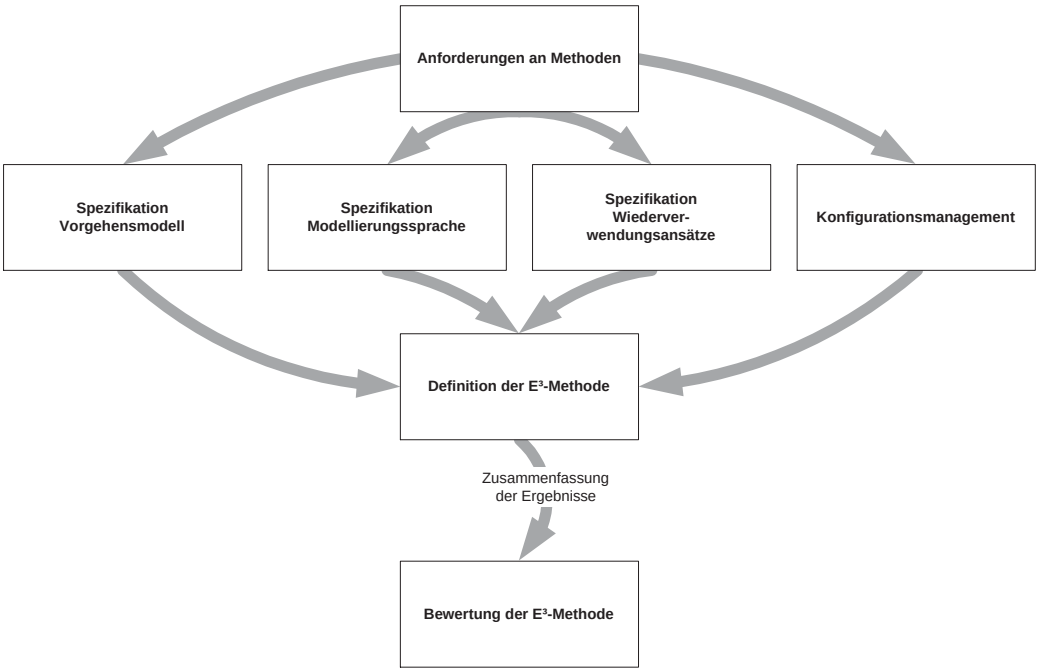



Abbildung 29: Gedankengang in Teil III

# 6 Das E$^3$-Modell

Dieser Abschnitt stellt das E$^3$-Modell vor. Im ersten Teil erfolgt eine Beschreibung der benötigten Elemente zum Method Engineering und im zweiten Teil die Definition einer Sprache zu deren Spezifikation.

## 6.1 Elemente des E$^3$-Modells

Im ersten Teil der Beschreibung wird der Schwerpunkt auf die Elemente des E$^3$-Modells gelegt, erst im Abschnitt 6.2 erfolgt eine Beschreibung der zahlreichen Beziehungen zwischen diesen Elementen. Um auf der einen Seite die Anwendung ausgewählter Techniken des Vorgehensmodells der E$^3$-Methode zu demonstrieren und auf der anderen Seite die Anschaulichkeit der Beschreibung zu erhöhen, werden an einigen Stellen *Ontologien* (vgl. Kapitel 8.3.1.3) verwendet. Die Abschnitte 6.1.1 bis 6.1.3 widmen sich der Abbildung von Modellierungssprachen. Die Abschnitte 6.1.4 bis 6.1.9 führen insbesondere die Konzepte ein, die zur Bildung von Vorgehensmodellen notwendig sind.

### 6.1.1 Objekttypen

Betrachtet man die Bestandteile existierender Metamodelle, so definieren diese stets grundlegende Konzepte, auf welche die Elemente der Objektebene abgebildet werden. Diese werden im E$^3$-Modell als Objekttypen bezeichnet. Den Konzepten lassen sich Eigenschaften zuordnen. Für deren Beschreibung wird das Konzept des Propertytypen definiert. Die Summe aller Objekt- und Propertytypen beschreiben den Kontext eines Metamodells. Beziehungen zwischen den Konzepten werden selbst wieder als Objekttyp abgebildet. Dieser bildet die eigentliche Verknüpfung mit Hilfe seiner Propertytypen ab. Dieser Ansatz entspricht in seinen Grundzügen existierenden ERM-basierten Ansätzen, von denen zahlreiche Varianten zur Metamodellierung eingesetzt wurden.[93]

### 6.1.2 Viewtypen

Die Softwareentwicklung basiert zur Komplexitätsreduktion auf dem Prinzip der Sichtenbildung.[94] Die jeweilige Sicht auf den Gegenstandsbereich wird in einem eigenen Modell abgebildet, die Reduktion der Komplexität beruht demzufolge auf der Modularisierung jeweils interessierender Realitätsausschnitte. Zusätzlich werden zumeist sichtenspezifische Sprachen zur Modellierung eingesetzt, die für die jeweilige Sicht besonders geeignet sind. Demzufolge ergibt

[93] Siehe dazu den Entity-Relationship Modellierungsansatz von STRAHRINGER (vgl. [Stra96], S. 358ff), die Ansätze von SMOLANDER ET AL. (vgl. Abschnitt 5.3.3) und EBERT (vgl. Abschnitt 5.3.2).

[94] Zu den allgemeinen Prinzipien siehe Abschnitt 3.2, zu den Sichten siehe 2.5.2.

sich die Notwendigkeit der Abstimmung bzw. Integration der einzelnen Sichten (vgl. [Stra96], S. 50).

Eine Möglichkeit der Definition von Beziehungen zwischen den Sichten besteht in der Verwendung gemeinsamer Objekttypen, was zu einer Überlappung der Sichten führt, eine andere in der Definition von Beziehungen zwischen Objekttypen unterschiedlicher Sichten oder zwischen Objekttypen und der Sicht selbst.[95] Gerade diese Definition von Beziehungen der Sichten zueinander ist eine der Hauptaufgaben bei der Methodenentwicklung. Ein in diesem Zusammenhang häufig diskutiertes Beispiel ist die Integration von Datenflussdiagrammen (DFD) und Entity-Relationship Modellen (ERM), bei der die Speicher als Kandidaten für Objekttypen des ERM (vgl. [Stra96], S. 51) verstanden werden können oder lediglich die Forderung nach Darstellung der im DFD verwendeten Daten in einem ERM erhoben wird (z. B. bei [StHa97], S. 297 oder [FeSi01], S. 178).

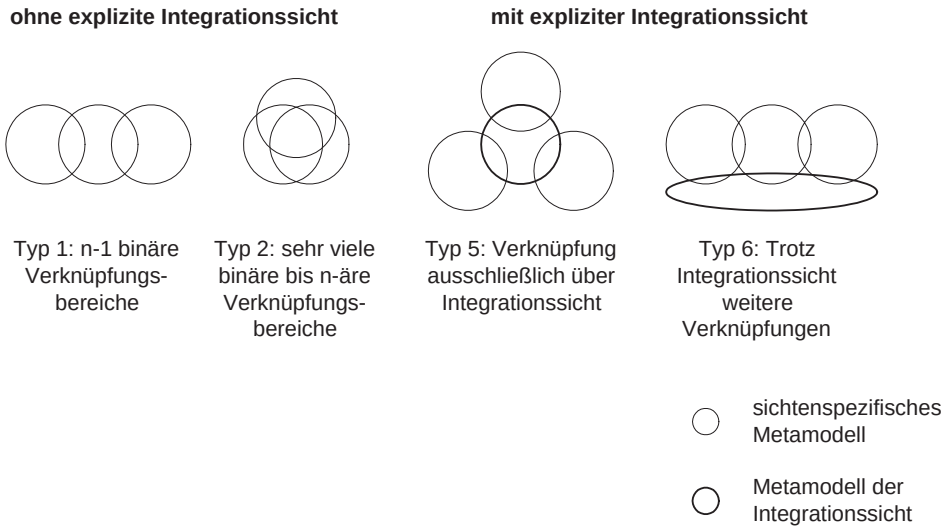



Abbildung 30: Idealisierte Verknüpfungsstrukturen (Auswahl aus [Stra96], S. 53)

STRAHRINGER typologisiert 6 verschiedene Arten der Verknüpfung von Sichten, eine Auswahl wird in Abbildung 30 dargestellt. Grundlage bildet zum einen die Anzahl der Beziehungen zwischen den Sichten und zum anderen deren Pragmatik. So werden beispielsweise von SCHEER die Sichten in der Architektur integrierter Informationssysteme (ARIS) nach dem Kriterium der Ähnlichkeit des semantischen Zusammenhangs gebildet. Zu deren Verknüpfung führt er zusätzlich die integrierende Steuerungssicht ein (vgl. [Sche98], S. 33ff).

Die Unterscheidung der Sichten auf Ebene der Metamodelle wird im E$^3$-Modell durch **Viewtypen** realisiert, wobei die Aufnahme eines Objekttypen in eine Sicht durch die Bildung eines

[95] Ersteres wird z. B. in der UML angewendet, wenn Klassennamen in Klassen- und Paketdiagrammen erscheinen, Letzteres entsteht bei der Verfeinerung von Use Cases in Aktivitätsdiagrammen.

**Viewobjekttypen** vorgenommen wird. Allein auf Ebene der Metamodelle lässt sich jedoch eine Sichtenarchitektur nicht vollständig beschreiben (vgl. [Stra96], S. 53). Es werden zusätzliche Konzepte auf Ebene der Modelle notwendig, da mehrere Sichten gleichen Typs (also mit gleichen Objekttypen) denkbar sind. Auch hier lassen sich Datenflussdiagramme als Beispiel anführen, wenn ein einzelnes DFD bereits als eine Sicht auf die Realität verstanden wird. Weiterhin muss gefordert werden, dass sich Sichten sowohl auf Ebene der Metamodelle als auch auf der Ebene der Modelle hierarchisch anordnen lassen. Die Verfeinerung der Aktivitäten in untergeordneten DFD's dient hier als Beispiel. Auf Ebene der Metamodelle ergibt sich beispielsweise aus der Unterteilung der Daten-, Funktions-, Organisations-, Leistungs- und Steuersicht im ARIS in weitere Ebenen des Fachkonzepts, des Datenverarbeitungskonzepts und der Implementierung die Notwendigkeit von deren Abbildung in einem Metamodell.

### 6.1.3 Präsentationstypen

Gemäß den Ausführungen von STACHOWIAK bzw. SCHÜTTE ergab sich der für diese Arbeit nützliche Begriff des Informationsmodells (siehe dazu Abschnitt 2.4). Dessen Beziehung zur Facette der grafischen Modelle enthält den Hinweis auf die zumeist (jedoch nicht notwendigerweise) gewählte Explikation in Form grafischer Darstellungen. Deren Art und Weise wird als **Präsentationstyp** bezeichnet[96] und somit von den Objekttypen selbst getrennt.[97] Deren Darstellung wird als **Präsentationsobjekttyp** bezeichnet. Präsentations- und Präsentationsobjekttypen beziehen sich stets auf entsprechende Ausschnitte des Metamodells, woraus sich deren jeweilige Zuordnung zu den View- und Viewobjekttypen ergibt.

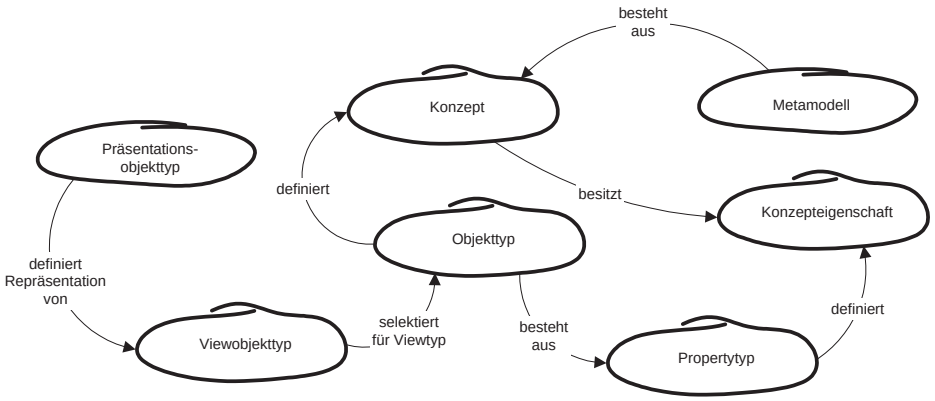



Abbildung 31: Ontologie der Objekt,- Viewobjekt- und Präsentationsobjekttypen im E$^3$-Modell

[96] Beispielsweise werden die Notationen im Klassenstrukturmodell und Sequenzdiagramm der UML als Präsentationstypen abgebildet.

[97] Einen Anhaltspunkt für die Sinnhaftigkeit dieser Vorgehensweise ergibt sich aus der Semiotik, wie sie in Abschnitt 2.1 diskutiert wurde.

### 6.1.4 Zieltyp

In der klassischen Organisationstheorie wird unter einem Ziel „... ein definierter, angestrebter Zustand verstanden, der durch die Erfüllung von Arbeitsaufgaben erreicht werden soll“ ([REFA92], S. 78). Zudem ermöglichen Ziele periodische Soll/Ist-Vergleiche, mit deren Hilfe korrigierende Änderungen im Arbeitsablauf ergriffen werden können (vgl. [LiSu89], S. 52). Ziele stehen untereinander in Beziehung. Dabei kann es sich einerseits um Konfliktbeziehungen handeln oder andererseits um eine Oberziel/Unterziel-Beziehung, die durch ein Verfeinern von Zielen in Teilziele entsteht, was wiederum zu einer Zielhierarchie führt.

Allgemein wird zwischen ökonomischen Zielen und sozialen Zielen unterschieden. Ökonomische Ziele sollen das wirtschaftliche Überleben des Unternehmens sicherstellen, soziale Ziele hingegen „... richten sich auf den Menschen mit seinen individuellen und sozialen Wünschen.“ ([LiSu89], S. 53) Beide Gruppen stehen in engem Zusammenhang, können sich gegenseitig bedingen und fördern, aber auch Zielkonflikte hervorrufen. Dennoch sollen in der vorliegenden Arbeit nur ökonomische Ziele berücksichtigt werden. Bezogen auf Projekte können dies alle denkbaren Teilziele sein, die dem Gesamtziel Projekterfolg zuträglich sind. Ob darunter auch für ein Unternehmen lebenswichtige Ziele wie Kundenzufriedenheit fallen, hängt davon ab, über welchen Zeitraum sich ein Projekt erstreckt. Hier kann es zu Konflikten zwischen Projekt- und Unternehmenszielen kommen. Das Ziel Projekterfolg kann bei einem kurzfristig angelegten Projekt durchaus mit den langfristigen Unternehmenszielen konfligieren, wenn beispielsweise Wartung, Anpassbarkeit und Wiederverwendung außerhalb der Verantwortung des Projektes liegen. Um die Zugehörigkeit zur Typebene deutlich zu machen, werden Ziele im Weiteren als **Zieltypen** bezeichnet.

### 6.1.5 Aufgabentyp

Nach LIEBELT sind Aufgaben „... dauerhaft wirksame Aufforderungen, etwas Bestimmtes zu tun.“ ([LiSu89], S. 18) Entscheidend ist hierbei, dass eine Aufgabe zielführend sein muss. Zweck einer Aufgabe ist es also, ein ihr zugeordnetes Ziel zu erfüllen, und konkrete Maßnahmen zu beschreiben, wie dieses Ziel erreicht werden kann (vgl. [REFA92], S. 78). Der klassische Ansatz der Aufgabenanalyse wurde von KOSIOL geprägt. Er spricht zunächst von einem Aufgabenkomplex, der sich an dem höchsten unternehmerischen Ziel orientiert. Dieser Aufgabenkomplex wird im Zuge der Aufgabenanalyse weiter in Teilaufgaben zergliedert. Dies kann anhand der fünf Kriterien *Verrichtung*, *Objekt*, *Rang*, *Phase* und *Zweckbeziehung* geschehen (vgl. [Kosi62], S. 42ff), wobei den ersten beiden Kriterien die größte praktische Bedeutung beigemessen wird (vgl. [Gait83], S. 17).

Durch die Aufgabenzerlegung entsteht ähnlich der Zielhierarchie eine Aufgabenhierarchie. Ziel- und Aufgabenhierarchie sind zwar über die Zuordnung von Aufgaben zu Zielen miteinan-

der verbunden, müssen jedoch nicht eine identische Baumstruktur aufweisen. So ist es zulässig, Aufgaben in Teilaufgaben zu zerlegen, auch wenn ihr zugeordnetes Ziel nicht verfeinert wurde. KOSIOL unterscheidet drei Phasen der Aufgabenerfüllung: Planung, Realisation und Kontrolle (vgl. [Kosi62], S. 56). In der Kontrollphase werden die Ergebnisse der Aufgabendurchführung mit den vorgegebenen Zielgrößen verglichen und es wird gegebenenfalls korrigierend in den Arbeitsprozess eingegriffen. Der Begriff der Aufgabe abstrahiert von einem konkreten Anwendungsfall. Um dies kenntlich zu machen, wird er im Weiteren **Aufgabentyp** genannt.

Um den inneren Aufbau einer Aufgabe zu beschreiben, führt KOSIOL die fünf Elemente *Gegenstand*, *Verrichtungsvorgang*, *sachliche Hilfsmittel*, *Raum* und *Zeit* an (vgl. [Kosi62], S. 43). LIEBELT bestimmt hierzu lediglich *Aufgabenobjekt* und *Verrichtung*. Aufgabenträger, Sachmittel und Information werden zwar einer Aufgabe zugeordnet, sind aber nicht deren Bestandteil (vgl. [LiSu89], S. 16ff). In den nachfolgenden Abschnitten werden beide Sichtweisen in Einklang gebracht und letztendlich vier Bestandteile einer Aufgabe festgelegt. Der Aufgabentyp stellt den zentralen Baustein von Vorgehensmodellen dar. Durch ihn und die im Folgenden beschriebenen Bestimmungselemente lassen sich die im Laufe eines Projektes durchzuführenden Tätigkeiten sehr genau spezifizieren.

Nach KOSIOL erstreckt sich jede Aufgabe „... auf einen Gegenstand (Objekt), an dem sich die geforderte Tätigkeit vollziehen soll." ([Kosi62], S. 43) Dessen Typ wird als **Aufgabenobjekttyp** bezeichnet. Generell werden materielle und immaterielle Aufgabenobjekttypen unterschieden. Unter materielle Aufgabenobjekttypen fallen beispielsweise Werkstücke oder Dokumente, während jedoch eine in einem Dokument enthaltene Information ein immaterieller Aufgabenobjekttyp ist (vgl. [LiSu89], S. 19).

Unter einem **Verrichtungstyp** wird die Tätigkeit verstanden, die zur Erfüllung des Aufgabentyps an dem Aufgabenobjekttyp durchgeführt wird. Dabei ist nicht entscheidend, ob ein Verrichtungstyp wiederum ein (komplexer) Arbeitsprozess ist, wie es KOSIOL für möglich hält, denn die Granularität der betrachteten Aufgabe hängt vom Zweck der Betrachtung ab.

Die organisatorische Einheit, welche den Aufgabentyp bzw. den Verrichtungstyp am Aufgabenobjekttyp durchführen soll, wird im Folgenden als **Aufgabenträgertyp** bezeichnet. Dabei handelt es sich um eine organisatorische Rolle, wie sie von KIESER und KUBICEK verstanden wird. Demnach handelt es sich bei einer Rolle um eine Position innerhalb der Organisation, welche durch eine Erwartungshaltung an das Verhalten des Positionsinhabers bei der Aufgabenerfüllung gekennzeichnet ist (vgl. [KiKu83], S. 397).[98]

Um einen Aufgabentyp zu erfüllen, bedient sich ein Aufgabenträgertyp diverser **Sachmitteltypen**. Darunter fallen sämtliche Dinge, die nicht zum Aufgabenobjekttyp gehören, aber „... den Menschen bei der Aufgabenerfüllung unterstützen." (vgl. [LiSu89], S. 24). LIEBELT fasst

[98] Sowohl der Aufgabenträgertyp als auch der Aufgabenträger stellen jeweils eine Rolle dar: ein Aufgabenträger ist eine organisatorische Position innerhalb eines konkreten Projektes, ein Aufgabenträgertyp dagegen eine organisatorische Position, die Teil einer von einem konkreten Projekt abstrahierenden Beschreibung ist.

den Begriff jedoch enger, indem er darunter nur materielle Objekte subsumiert, immaterielle Hilfsmittel wie Informationen hingegen separat zuordnet. In der vorliegenden Arbeit wird eine solche Trennung nicht vorgenommen und deshalb grundsätzlich jede Art von Hilfsmittel als Sachmitteltyp verstanden. Der KOSIOL'sche Gedanke des räumlichen Merkmals von Aufgabentypen fließt in die Betrachtungen nur soweit ein, als dass der Ort der Aufgabentyperfüllung als Ressource und damit als unterstützendes Hilfsmittel betrachtet wird.

### 6.1.6 Ereignistyp, Außen- und Innensicht

Der klassische Begriff der Aufgabe, der insbesondere in den Ausführungen von KOSIOL von zentraler Bedeutung ist, wird bei FERSTL und SINZ um den Ereignistypen und um die Differenzierung von Außen- und Innensicht auf einen Aufgabentypen ergänzt. Im Unterschied zur klassischen Organisationstheorie, welche keine Aussage über das eine Aufgabe auslösende Moment trifft, können Aufgaben nach FERSTL und SINZ erst ausgeführt werden, sobald alle **Vorereignisse** der Aufgabe eingetreten sind. Nach Erfüllung der Aufgabe tritt eine Menge von **Nachereignissen** ein, die wiederum auslösende Vorereignisse anderer Aufgaben sein können. Ein Aufgabentyp verfügt demnach über eine Menge von **Vorereignistypen** und eine Menge von **Nachereignistypen**.

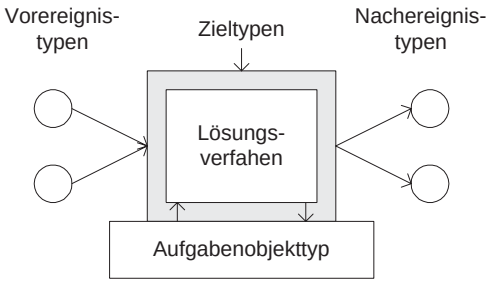



Abbildung 32: Aufgabenstruktur (i. A. a. [FeSi01], S. 90)

FERSTL und SINZ unterscheiden weiterhin zwischen **Außen-** und **Innensicht** eines Aufgabentyps (vgl. [FeSi01], S. 90). Die Außensicht definiert den Aufgabenobjekttyp, die Zieltypen des Aufgabentyps, die Vorereignistypen und die Nachereignistypen (vgl. [FeSi01], S. 93). Die Festlegung des Lösungsverfahrens erfolgt in der Innensicht eines Aufgabentyps. Dabei geht es um die Frage, wie der Aufgabentyp durchgeführt werden soll. „Ein Lösungsverfahren besteht aus einer Menge von Aktionen, die sequentiell oder parallel auf das Aufgabenobjekt einwirken oder Zustände des Aufgabenobjekts erfassen …" ([FeSi01], S. 95) Betrachtet man das Lösungsverfahren vor dem Hintergrund der Aufgabentypzerlegung, so kann man die Schritte des Lösungsverfahrens auch als Teilaufgabentypen des betrachteten Aufgabentyps verstehen. Ein weiterer Aspekt des Lösungsverfahrens ist die Bezugnahme auf einen Aufgabenträgertyp.

Eine Zuordnung von Sachmitteltypen findet in den Ausführungen von FERSTL und SINZ dagegen keine Erwähnung, wenngleich diese ebenfalls zum Lösungsverfahren und damit zur Innensicht eines Aufgabentyps gezählt werden müssen.

### 6.1.7 Artefakttyp

Der Einfachheit halber wird im Kontext der klassischen Organisationstheorie ein Aufgabenobjekttyp lediglich als atomarer Gegenstand behandelt, an dem der Aufgabentyp durchgeführt wird und deswegen nicht weiter aufgeteilt werden braucht. Um Vorgehensmodelle ausreichend detailliert beschreiben zu können, reicht diese grobe Sichtweise jedoch nicht aus, da Aufgabentypen durchaus auch mehrere Gegenstände zur Erfüllung benötigen können. Deshalb soll in der vorliegenden Arbeit der Begriff Aufgabenobjekttyp nur als Container verstanden werden, der die verschiedenen Gegenstände, welche durch den Aufgabentyp erzeugt, geändert oder in einem anderen Gegenstand eingebracht werden, zusammenfasst. Somit setzt sich ein Aufgabenobjekttyp aus mehreren Attributen zusammen, von denen jedes einen dieser Eingangs- oder Ausgangsgegenstände darstellt. In Anlehnung an das Begriffssystem des Rational Unified Process (RUP) werden solche Gegenstände im Folgenden **Artefakttyp** genannt (vgl. [Rati00]).

Ein Artefakttyp kann also in einer beliebigen Anzahl von Aufgabentypen verwendet werden. Wird ein Vorgehensmodell zur Systementwicklung eingesetzt, können Artefakttypen z. B. Modelle oder Quellcode sein. Darüber hinaus können Artefakttypen ebenso Repräsentationen für reale materielle Gegenstände einschließen. Jedes E$^3$-Element kann als Artefakttyp in einem Vorgehensmodell referenziert werden.

### 6.1.8 Bedingungstyp

In Abschnitt 6.1.6 wurden Vor- und Nachereignistypen zum Anstoßen der Durchführung einer Aufgabe vorgestellt. Dieser Ansatz kann als ereignisgetrieben bezeichnet werden und besitzt durch das explizite Anstoßen der Aufgaben aktiven Charakter.[99] Im Gegensatz dazu kann auch ein passiver Ansatz gewählt werden. Mit Hilfe von logischen Regeln lassen sich Anfangs- und Endbedingungen einer Aufgabendurchführung exakt festlegen. Allgemein wird eine solche Regel als **Bedingungstyp** bezeichnet. Jene Regeln, die die Umstände beschreiben, unter denen die Durchführung einer Aufgabe angestoßen werden kann, werden **Vorbedingungstypen** (Pre-Conditions) genannt. Entsprechend werden die Regeln, die bei Abschluss der Aufgabe erfüllt sein müssen, als **Nachbedingungstypen** (Post-Conditions) bezeichnet. Sie stellen ein Kriterium dar, anhand dessen geprüft werden kann, ob eine Aufgabe als erfüllt gelten kann. Die Herangehensweise über Vor- und Nachbedingungstypen wird beispielsweise von NOACK eben-

[99] Dieser wird beispielsweise von FERSTL und SINZ zur Kopplung betrieblicher Objekte (vgl. [FeSi01], S. 188) oder von SCHEER bei den Ereignisgesteuerten Prozessketten (EPK) (vgl. [Sche98], S. 19) beschrieben.

so aufgegriffen (vgl. [NoSc99], S. 169) wie von der Workflow Management Coalition (WFMC) (vgl. [Work98], S. 35f).

### 6.1.9 Gruppierungstyp

In Abschnitt 3.3 wurde bereits allgemein beschrieben, was unter einem Phasenmodell verstanden wird und welche anderen Begriffe damit in Verbindung stehen. Eine Phase ergibt sich aus der Bündelung von Aufgabentypen. Da der Begriff Phase weitgehend mit einer Einteilung nach zeitlichen Kriterien und der zugehörigen Abfolge assoziiert wird, wird in dieser Arbeit statt diesem ein anderer Begriff verwendet. Eine Einteilung nach zeitlichen Gesichtspunkten entspricht der Phase i. e. S. Diese spezialisiert die Phase i. w. S., indem als Ordnungskriterium die Zeit herangezogen wird. In der vorliegenden Arbeit hat diese Unterscheidung jedoch keine besondere Bedeutung, weswegen anstelle des Begriffes Phase der Begriff **Gruppierungstyp** verwendet wird.[100] Ein Gruppierungstyp ist somit eine unter sachlogischen Kriterien gebildete Menge von Aufgabentypen, wobei ein Aufgabentyp mehreren Gruppierungstypen angehören kann.

Wird dennoch nach dem Kriterium der Zeit gruppiert, entstehen herkömmliche Phasen, die durch Meilensteine begrenzt werden (vgl. Abschnitt 3.3). Bei einem Meilenstein handelt es sich um eine zeitliche Zielvorgabe, also um einen Termin, an dem eine Phase abgeschlossen werden soll. Um diesen Zeitpunkt mit Hilfe von Instrumenten der Zeitplanung berechnen zu können, müssen in Attributen der Aufgabentypen zusätzlich ihre zeitlichen Eckdaten festgehalten werden, wie beispielsweise Anfangszeitpunkte, Endzeitpunkte oder die geplante Dauer.

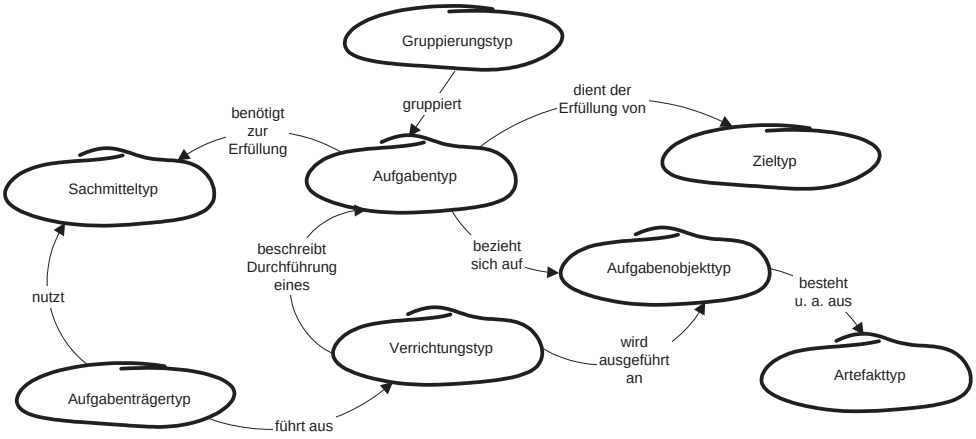



Abbildung 33: Ontologie der wichtigsten Elemente der Vorgangsebene

[100] Damit lässt sich eine größere gedankliche Distanz zur Zeit als Gruppierungskriterium herstellen und die Zugehörigkeit des Begriffes zur projektunabhängigen Typebene unterstreichen.

## 6.2 Spezifikation

### 6.2.1 Auswahl der Spezifikationsform

Für die Spezifikation des $E^3$-Modells können im Wesentlichen eine *Normsprache*, ein *logikbasierter Ansatz* oder ein *netzorientierter Ansatz* angewendet werden.[101] Erstere basiert auf einer **Fachsprache**. Diese weist die Charakteristika menschlicher Sprachen auf, verfügt aber über einen sehr spezialisierten Wortschatz. Für den Einsatz spricht ihre Methodenneutralität[102] und ihre Ausdrucksmächtigkeit, die sich in erster Linie aus ihrem umfangreichen Wortschatz und ihrer Flexibilität ableitet. Gerade aus der Flexibilität und Ausdrucksmächtigkeit rühren jedoch die bekannten Schwachstellen menschlicher Sprache hinsichtlich präziser und eindeutiger Formulierung. An erster Stelle stehen dabei Homonym- und Synonymprobleme, aber auch Kontextabhängigkeit und Vagheit im Ausdruck führen zu Problemen bei der Interpretation der Spezifikation. Um diese Schwachpunkte zu vermeiden, aber dennoch die Vorteile zu nutzen, müssen weitere Einschränkungen vereinbart werden. Dies betrifft in erster Linie den Wortschatz, der auf ein fest definiertes eindeutiges Wörterbuch beschränkt wird. Auch der Grammatik, also den Regeln zum Satzbau, wird ein Teil ihrer Flexibilität genommen. Eine Sprache, die solchen Einschränkungen unterliegt, nennt man **Normsprache** (vgl. [Lehm98], S. 366f). Sie bildet einen Kompromiss zwischen der formalen und der menschlichen Sprache. Damit können Präzision und Eindeutigkeit mit den Vorteilen von Fachsprachen kombiniert werden. Normsprachen werden deshalb seit langem vor allem in frühen Phasen der Systementwicklung und in der Phase der Systemeinführung eingesetzt (vgl. [Jab+97], S. 154). Man verspricht sich davon hauptsächlich verbesserte Kommunikation innerhalb der Organisation bzw. zwischen Organisationen.

Der **logikbasierte Ansatz** wird auf dem Gebiet der Künstlichen Intelligenz zur Spezifikation von Wissensrepräsentationen genutzt (vgl. [Wern95], S. 650). Dabei können verschiedene Wege eingeschlagen werden, unter anderem auch durch prädikatenlogische Ausdrücke.[103] Mit diesen lassen sich beliebige Sachverhalte ausdrücken. Für die Verwendung dieses Ansatzes spricht zunächst dessen Untermauerung in seinen theoretischen Grundlagen (vgl. [Wodt97], S. 35). Ferner erweist sich als vorteilhaft, dass sich natürliche Sprache leicht in Prädikatenlogik umsetzen lässt. Der Hauptvorteil besteht schließlich darin, dass neben „... Ausdrucksmit-

[101] Diese Auswahl wird an dieser Stelle nicht hergeleitet, sondern im Folgenden nur begründet.

[102] Die Spezifikation kann unabhängig von der später verwendeten Methode zur Systementwicklung erstellt werden. Verschiedene Methoden können so eine gemeinsame Spezifikation nutzen. Außerdem können Fachexperten im Fachgebiet etablierte Ausdrücke weiterverwenden und brauchen diese nicht methodenadäquat zu formulieren (vgl. [Jab+97], S. 153).

[103] Diese geht auf unterster Ebene von einer **Logik** aus, die Syntax und Semantik der formalen Sprache festlegt und damit den Rahmen für die Struktur und Bedeutung von Aussagen vorgibt. Darauf aufbauend wird ein **Kalkül** festgelegt, das über Operationen Schlussfolgerungen aus Aussagen ermöglicht. Weiterhin werden Mittel der **Repräsentation** benötigt, die nicht nur in der Lage sind, die Aussagen selbst, sondern auch deren Zusammenhänge auszudrücken. Schließlich werden unter dem Begriff **Steuerung** sämtliche Strategien und Heuristiken subsumiert, die sinnvolle Handlungsempfehlungen zur Anwendung der Kalküle geben (vgl. [Wern95], S. 650).

teln zur Repräsentation Schlussfolgerungsregeln existieren, mit deren Hilfe sich aus der Menge von Voraussetzungen Schlussfolgerungen ableiten lassen.“ ([Wern95], S. 653) Dies kann sich gleichzeitig jedoch als größter Nachteil der Herangehensweise herausstellen, da hohe Ausdrucksmächtigkeit einer Logik mit schwierigerer Beherrschbarkeit ihrer Komplexität einhergeht. Je umfangreicher das abzubildende System, desto schwieriger wird es, das Modell handhabbar und verständlich zu halten. Letzteres liegt auch darin begründet, dass sich prädikatenlogische Ausdrücke nur unzureichend intuitiv visualisieren lassen (vgl. [Wodt97], S. 35). Aus diesen Gründen wird der Weg, Vorgänge mit prädikatenlogischen Ausdrücken zu beschreiben, in der Praxis kaum beschritten.

Das bedeutendste Spezifikationsmittel für Modelle sind zweifellos **netzorientierte Ansätze**. Dies rührt in erster Line daher, dass sich diese sehr leicht visualisieren lassen und damit intuitiv verständlich sind. Solche Netze liegen im Allgemeinen als gerichtete Graphen vor. Ein **Graph** besteht aus einer nichtleeren Menge von Knoten, aus einer Menge von Kanten und aus einer Funktion, die festlegt, zwischen welchen Knoten eine Kante besteht, die Kantenmenge also in das Kreuzprodukt der Knotenmenge abbildet (vgl. [Wern95], S. 613). Als gerichteten Graph bezeichnet man die Art von Graph, bei der die Kanten nur in einer Richtung passiert werden können. So lassen sich eindeutige Abhängigkeiten und z. B. Ablaufreihenfolgen ausdrücken.

Im Rahmen dieser Arbeit wird zur genaueren Beschreibung der Beziehungen der Elemente zueinander eine Darstellung in der zur E$^3$-Methode gehörigen Notation gewählt, ergänzt durch eine normsprachliche Beschreibung. Diese E$^3$-Notation ist der UML sehr ähnlich,[104] sodass zu deren Verständnis ein Lesen des Abschnitts 6.3 nicht zwingend Voraussetzung ist.

### 6.2.2 Überblick

Das E$^3$-Modell ist ein Metamodell, das die einheitliche Definition von Methoden erlaubt. Dabei können deren Modellierungssprache und Vorgehensmodell abgebildet werden. Das E$^3$-Modell ermöglicht zusätzlich die Abbildung der Instanzen dieser sprachbasierten Metamodelle. Aus diesen beiden Ansprüchen ergibt sich eine Zweiteilung des E$^3$-Modells in **Typ-** und **Instanzenebene**. Beide Ebenen werden horizontal und vertikal jeweils in weitere 3 Ebenen unterteilt. Die vertikalen Ebenen dienen der Verfeinerung. Die horizontalen Ebenen trennen den Kontext eines Modells (abgebildet in der **Kontextebene**) von der grafischen Präsentation (abgebildet in der **Präsentationsebene**) über die Bildung von Ausschnitten (in der **Viewebene**). Aus 3x3x2 Ebenen folgen insgesamt 18 Berührungspunkte, die als **Elemente des E$^3$-Modells** oder auch **E$^3$-Elemente** bezeichnet und in Abbildung 34 dargestellt werden. Zu den E$^3$-Elementen werden auch die der **Vorgangsebene** gezählt, auf deren Darstellung in Abbildung 34 verzichtet wurde. Diese dienen der Darstellung von Vorgehensmodellen. Innerhalb der Vorgangsebene wird auf

[104] In Bezug auf die *Darstellung* von Assoziations- und Vererbungsbeziehungen ist sie identisch.

die restlichen $E^3$-Elemente als Artefakttypen Bezug genommen. Die Beschreibung aller $E^3$-Elemente ist Gegenstand der folgenden Abschnitte.

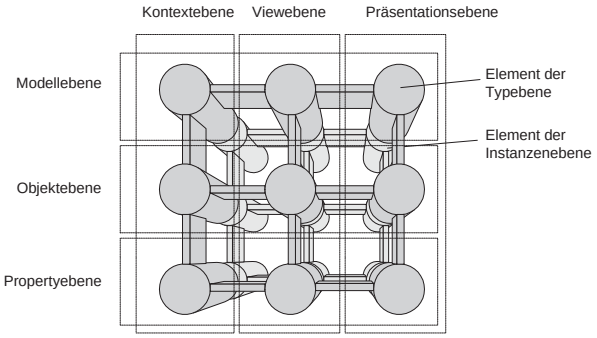



Abbildung 34: Das $E^3$-Modell

Zur Beschreibung der Beziehungen zwischen den Elementen werden diese in direkt und indirekt unterteilt. Eine Beziehung wird **direkt** genannt, wenn sie zwischen zwei Elementen besteht. Eine Beziehung ist **indirekt**, wenn sie aus direkten Beziehungen der beteiligten Elemente folgt, aber nicht direkt ist. Die **Weglänge** zwischen zwei Elementen A und B beschreibt die minimale Anzahl der direkten Beziehungen zwischen Elementen des $E^3$-Modell, die notwendig sind, um die indirekte Beziehung zwischen den Elementen A und B zu beschreiben. Besitzen A und B eine direkte Beziehung ist deren Weglänge 1.

Ein Element A heißt **vorgeordnet** einem Element B (bzw. Element B **nachgeordnet** Element A), wenn A eine direkte Beziehung zu B hat, beide Elemente in der selben horizontalen aber in verschiedenen vertikalen Ebenen liegen und die Weglänge zum $E^3$-Element *Metamodell* für B größer ist als für A.

Ein Element A heißt **übergeordnet** einem Element B (bzw. Element B **untergeordnet** einem Element A), wenn A eine direkte Beziehung zu B hat, beide Elemente in der selben vertikalen Ebene liegen und A Bestandteil der Instanzenebene und B Bestandteil der Typebene ist.

### 6.2.3 Typebene

In der **Typebene** erfolgt die Definition von Metamodellen. Diese werden auf den drei Ebenen **Kontext-**, **View-**, und **Präsentationsebene** beschrieben (horizontale Unterteilung). Auf jeder der Ebenen existieren weitere Elemente, welche gemeinsam die Konzepte, deren Eigenschaften und Repräsentation beschreiben. Diese ergeben sich aus den Schnittpunkten der horizontalen und vertikalen Ebenen. Letztere entstehen aus der mengenmäßigen Zusammenfassung von Konzepten und deren Eigenschaften (vertikale Unterteilung).

Die Abbildung 34 liefert einen Überblick über alle Elemente. Den Bestandteilen der Kontextebene werden Elemente der Viewebene nachgeordnet. Diesen Typen wiederum werden die

Elemente aus der Präsentationsebene nachgeordnet. Zusammenfassend gilt: Die Typisierung eines Metamodells im E$^3$-Modell besteht aus allen Bestandteilen der Typebene, die sich direkt und indirekt einem Metamodell[105] zuordnen lassen.

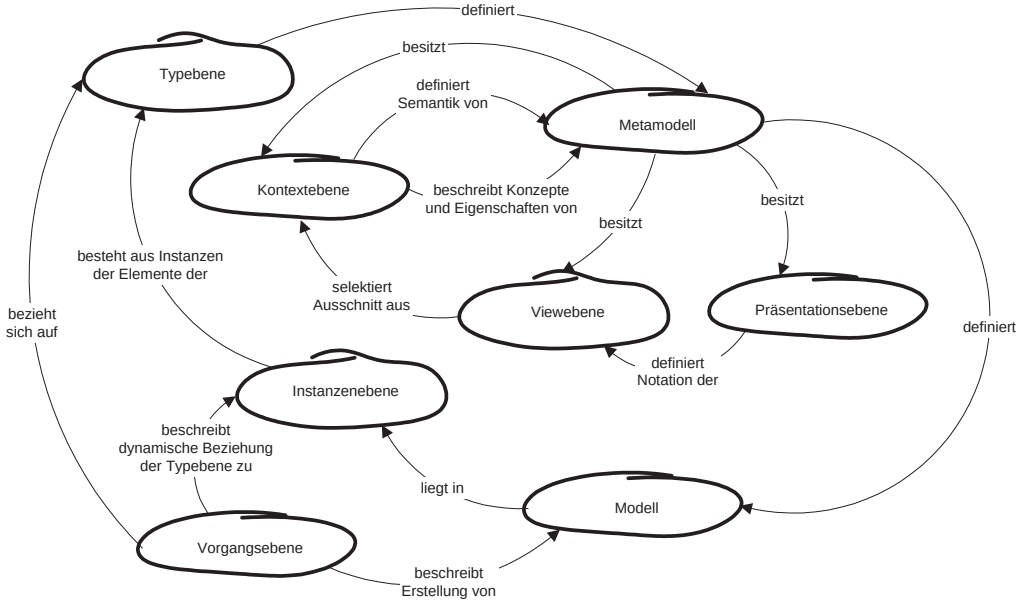



Abbildung 35: Ontologie der Ebenen im E$^3$-Modell

### 6.2.4 Instanzenebene

Jedem Typen der Typebene können Instanzen zugeordnet werden. Die Menge aller Instanzen, die sich der Instanz eines Metamodells zuordnen lassen, bildet ein Modell. Gleichzeitig wird aber auch die Instanz selbst als Modell bezeichnet. Wenn es im Folgenden nicht anders beschrieben wird, ist mit dem Begriff Modell die Instanz eines Metamodells gemeint. Da das E$^3$-Modell Metamodelle und deren Modelle umfasst, werden den Elementen der Typebene jeweils 9 Elemente der Instanzenebene zugeordnet.

### 6.2.5 Kontextebene

Die Kontextebene des E$^3$-Modells beschreibt den Kontext eines Metamodells bzw. davon abgeleiteter Modelle. Sie gliedert sich in **Metamodell**, **Objekttyp** und **Propertytyp**. Wird mit der durch das Metamodell definierten Sprache ein Modell erzeugt, so werden von jedem dieser Typen Instanzen gebildet. Es entstehen **Modelle**, **Objekte** und **Properties**. Jedes Objekt hat pro Propertytyp seines Objekttyps genau ein Property.

[105] als Element des E$^3$-Modells

Für Propertytypen muss zusätzlich ein **Wertebereich**, eine **Multiplizitätsangabe** sowie ein **Strukturtyp** spezifiziert werden. Der Strukturtyp bestimmt, in welcher Form die Ausprägungen des Wertebereichs durch den Propertytypen verwaltet werden. Es stehen Listen, Bäume sowie Mengen zur Verfügung. Die Multiplizität gibt an, wie viele Ausprägungen der zu einem Propertytypen zugeordnete Wertebereich minimal bzw. maximal haben kann. Der *einfache Wertebereich* kann als Zeichenkette (String), Ganzzahl (Integer), Gleitkommazahl (Real), Datum, Zeit und Wahrheitswert angegeben werden. Zusätzlich dazu existiert noch der *Wertebereich* $E^3$, der jedes beliebige Element des $E^3$-Modells umfassen kann. Im Anhang 11.1 befindet sich die Spezifikation der Elemente in der $E^3$-Notation.

### 6.2.6 Viewebene

Eine wesentliche Eigenschaft von Modellen ist die Reduktion der Komplexität des betrachteten Sachverhalts durch Einschränkung der dargestellten Aspekte. Diese Funktionalität wird von der Viewebene übernommen. Die Viewebene definiert einen Ausschnitt aus der Kontextebene. Umfangreiche Methoden machen eine mehrstufige Sichtenbildung notwendig. In diesem Fall wird auf eine Sicht eine weitere Sicht definiert. Abbildung 36 stellt solche teilweise nachgeordneten Views dar. Den Elementen einer Viewebene werden Elemente der Präsentationsebene (bzw. nachgeordnete Elemente der Viewebene) nachgeordnet.

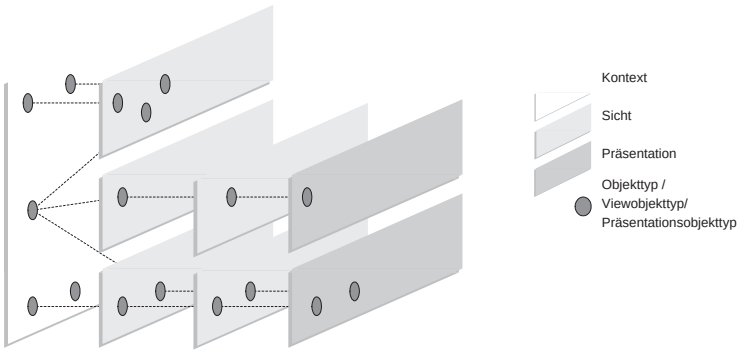



Abbildung 36: Mehrstufige Sichtenbildung

Im Schnittpunkt zwischen Modell- und Viewebene liegt der **Viewtyp**, der einem Metamodell oder einem Viewtypen nachgeordnet ist. Jedem Viewtypen werden **Viewobjekttypen** untergeordnet, welche mit Objekttypen bzw. Viewobjekttypen eines vorgeordneten Viewtypen verknüpft werden. Abhängig vom Viewobjekttypen existieren für ausgewählte Propertytypen (bzw. vorgeordnete Viewpropertytypen) die **Viewpropertytypen**.

Die Instanzen der Viewtypen werden als **View** bezeichnet. Instanzen der Viewobjekttypen werden als **Viewobjekte** und Instanzen der Viewpropertytypen als **Viewproperties** bezeichnet.

Jedes Viewobjekt hat pro Viewpropertytyp seines Viewobjekttyps genau ein Viewproperty.[106] Die Spezifikation der Elemente ist dem Anhang 11.1 zu entnehmen.

### 6.2.7 Präsentationsebene

Die grafischen Darstellungen eines Metamodells werden in der Präsentationsebene beschrieben. Im Schnittpunkt zwischen dieser und der Modellebene steht der **Präsentationstyp**, der einem Viewtypen nachgeordnet ist. Jedem Präsentationstypen werden **Präsentationsobjekttypen** untergeordnet, welche den entsprechenden Viewobjekttypen nachgeordnet werden. Abhängig vom Präsentationsobjekttypen existieren die **Präsentationspropertytypen**, welche zusätzlich mit den entsprechenden Viewpropertytypen verbunden werden. Unterschiedliche Präsentationstypen für einen Viewtypen stellen einen Ausschnitt des Metamodells verschieden dar. Die Präsentationsebene kann keine Filterung realisieren. Informationen der zugeordneten Viewebene müssen vollständig in eine grafische Darstellung überführt werden. Jedem Viewtypen, dem kein weiterer Viewtyp nachgeordnet ist, muss ein Präsentationstyp nachgeordnet werden. Bei allen anderen Viewtypen geschieht dies optional.

Die Instanzen der Präsentationstypen werden als **Präsentationen** bezeichnet, die des Präsentationsobjekttypen als **Präsentationsobjekte**. Deren Form der Darstellung wird im Wesentlichen von ihren Präsentationsobjekttypen bestimmt. Hinzugefügt wird der Ort sowie die Größe der Abbildung innerhalb einer Präsentation.

Instanzen der Präsentationspropertytypen werden als **Präsentationsproperties** bezeichnet. Jedes Präsentationsobjekt hat pro Präsentationspropertytyp seines Präsentationsobjekttyps genau ein Präsentationsproperty.[107] Die Form ihrer Darstellung wird vom Präsentationspropertytypen bestimmt. Die Werte des Property werden durch sie dargestellt. Die genaue Spezifikation der Elemente ist dem Anhang 11.1 zu entnehmen.

### 6.2.8 Pakete

Ein Paket stellt eine Gruppierung von Modellelementen dar.[108] Damit dient das Paket der Komplexitätsreduktion (vgl. Abschnitt 3.2). Wie auch in der UML, wird das Paket hier verwendet, um eine hierarchische Struktur abzubilden (vgl. [Balz99], S. 31ff). Dementsprechend können Pakete anderen Paketen untergeordnet werden.

Jedes Element des E$^3$-Modell ist in genau einem Paket definiert, kann jedoch in mehreren anderen Paketen enthalten sein. Beziehungen zwischen Paketen lassen sich auf Abhängigkeiten der Elemente innerhalb der Pakete zurückführen. Diese würden nur dann nicht existieren, wenn sich das Gesamtsystem verlustfrei in Teilprobleme zerlegen ließe. Da dies i. d. R. nicht möglich

[106] Dies entspricht der Beziehung der Properties zu ihrem Objekt aus Abschnitt 6.2.5.
[107] Dies entspricht der Beziehung der Properties zu ihrem Objekt aus Abschnitt 6.2.5.
[108] in Anlehnung an die Verwendung des Begriffes in der UML (vgl. [Obje00b], S. 3-15ff; S. 2-173)

ist, müssen Möglichkeiten der Darstellung von Abhängigkeiten von Paketelementen verschiedener Pakete geschaffen werden. Hierzu bedient man sich eines Verweises auf ein Element. Dazu wird auf das jeweilige Modellelement mit der Angabe seines Herkunftspaketes referenziert. Ein solches Element wird als *referenziertes $E^3$-Element* bezeichnet.

### 6.2.9 Namensräume

Ein **Namensraum** „. . . is a part of a model that contains a set of ModelElements each of whose names designates a unique element within the namespace“ ([Obje00b], S. 2-40). Innerhalb eines Namensraumes müssen damit die Bezeichnungen aller gleichartigen Modellelemente eindeutig sein, wobei jedes Modellelement gleichzeitig genau einem Namensraum zugeordnet ist. Ohne Namensräume wird die Lesbarkeit von Metamodellen erheblich beeinträchtigt, da sich Objekttypen ggf. nur durch die Wertebereiche ihrer Propertytypen unterscheiden können.

| **$E^3$-Element** | **Namensraum für** |
|---|---|
| Metamodell | Objekttypen, Viewtypen |
| Objekttyp | Propertytypen |
| Viewtyp | Viewobjekttypen, Präsentationstypen |
| Viewobjekttyp | Viewpropertytypen |
| Präsentationstyp | Präsentationsobjekttypen |
| Präsentationsobjekttyp | Präsentationspropertytypen |

Tabelle 6: Namensräume im $E^3$-Modell

Tabelle 6 fasst die im $E^3$-Modell definierten Namensräume zusammen. Dabei wird davon ausgegangen, dass eine Typisierung an Lesbarkeit gewinnt, wenn je Metamodell, Viewtyp und Präsentationstyp ein entsprechender Objekttyp, Viewobjekttyp und Präsentationsobjekttyp eindeutig anhand seines Namens identifiziert werden kann.

Dem bisherigen Vorschlag folgend, würde gelten, dass die Namen für Objekttypen im gesamten Metamodell eindeutig sein müssen. Diese Forderung kann bei Modellen mit einer Vielzahl von Konzepten[109] nicht erfüllt werden, weil vor allem das Problem der Homonyme besonders häufig auftritt. Eine ähnliche Argumentation kann für die Viewtypen geführt werden. Aus diesem Grund werden die Namensbereiche von Objekttypen und Viewtypen auf ein Paket begrenzt.[110] Damit wird die Erweiterbarkeit des jeweiligen Metamodells erheblich erleichtert. Der Fall, dass in einem Paket ein Objekttyp und ein referenzierter Objekttyp gleichen Namens auftritt, wird als Grenzfall zugelassen, da für den referenzierten Objekttypen die Paketangabe eine eindeutige Unterscheidung zulässt.

[109] insbesondere bei Referenzmetamodellen
[110] sofern im Metamodell Pakete definiert werden

### 6.2.10 Vererbungsbeziehungen

Der Nutzen von Vererbung wird in der Softwaretechnik in der Reduktion des Implementierungsaufwandes gesehen (vgl. [Cro$^+$98], S. 572f). Bei der Modellierung ist der Vorteil dieses Konstruktes die Wiederverwendung bereits bestehender Modellteile (vgl. [Över00], S. 194). In beiden Anwendungsfällen wird die Mengenkomplexität verringert. Weiterhin ist es möglich, sich schnell in bestehende Modelle einzuarbeiten,[111] da zunächst die grundlegenden (generalisierten) Konzepte verstanden und schrittweise die Details (Spezialisierungen) erschlossen werden können. Nach LANG kommt der Vererbung weiterhin eine hohe Bedeutung im Zuge des Anpassungsprozesses von Referenzmodellen zu, da mit ihrer Hilfe, durch die Spezialisierung der im Referenzmodell vorhandenen Konstrukte, die Lücke zwischen diesem und den unternehmensindividuellen Modellen geschlossen werden kann (vgl. [Lang97], S. 52).

HARS fügt hinzu, dass durch Vererbung die „... Identität zwischen scheinbar unterschiedlichen Objekten aufgezeigt wird." ([Hars94], S. 76) In der UML wird unter Vererbung die Übernahme von Eigenschaften der Superklasse durch die Subklasse bezeichnet (vgl. z. B. [Oest01], S. 45). Gemeint ist dabei die Definition von Attributen und Methoden für mehrere Subklassen in einer Superklasse. In Anlehnung an dieses Instrument der Strukturierung durch Generalisierung und Spezialisierung wird im E$^3$-Modell unter Vererbung die Weitergabe von Propertytypen eines **Superobjekttyps** an seine durch diese Beziehung untergeordneten **Subobjekttypen** verstanden. Da sich View- und Präsentationsebene stets auf die Vereinbarungen im Kontext beziehen und die dort definierten Konzepte und Beziehungen „nur" selektieren und darstellen, bleibt die Definition einer Vererbungsbeziehung im E$^3$-Modell auf die Objekttypen beschränkt.

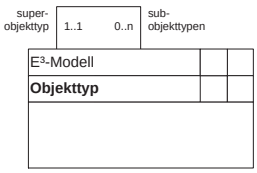



Abbildung 37: Spezifikation der Vererbung

In Analogie zur Softwaretechnik ist es denkbar, Eigenschaften von Superobjekttypen zu überschreiben. Dies entspricht einer Redefinition von Wertebereichen und Strukturtypen der Propertytypen. Um die Klarheit der abgebildeten Metamodelle zu erhöhen, wird aber auf solche Konstrukte verzichtet.[112] An die Stelle einer solchen Redefinition tritt stattdessen die Einführung eines neuen Propertytypen oder die Veränderung der Vererbungshierarchie. Diese Vorgehensweise bezeichnet CRONGORAC als **inkrementelle Vererbung** (vgl. [Cro$^+$98], S. 573). Damit wird mit der hier angewendeten Art der Vererbung eine schrittweise Spezialisierung erreicht,

[111] zum Grundsatz der Klarheit siehe Abschnitt 4.4
[112] Zu den Problemen bei der Redefinition von Methoden siehe z. B. [Cro$^+$98], S. 557.

d. h. ein Subobjekttyp enthält genau so viele oder mehr Propertytypen wie sein Superobjekttyp. Die Wertebereiche, Strukturtypen und Multiplizitäten der Propertytypen des Superobjekttyps entsprechen denen der Propertytypen des Subobjekttyps.

Besitzt eine Klasse der UML zwei oder mehrere Superklassen, so spricht man von **Mehrfachvererbung** (vgl. [Oest01], S. 47). Bei Anwendung dieser entstehen zahlreiche Probleme. In der Abbildung 38 kommt es zu einem Konflikt, falls zwei Objekttypen *OT2* und *OT3* mit je einem Propertytypen *pt1* bestehen, welcher in den Objekttypen bezüglich seines Wertebereichs eine unterschiedliche Ausprägung hat. Existiert nun ein Objekttyp *OT4*, der von *OT2* und *OT3* erbt, ist der Wertebereich von dessen *pt1* nicht eindeutig. Die Abbildung 38 stellt das beschriebene Beispiel dar.

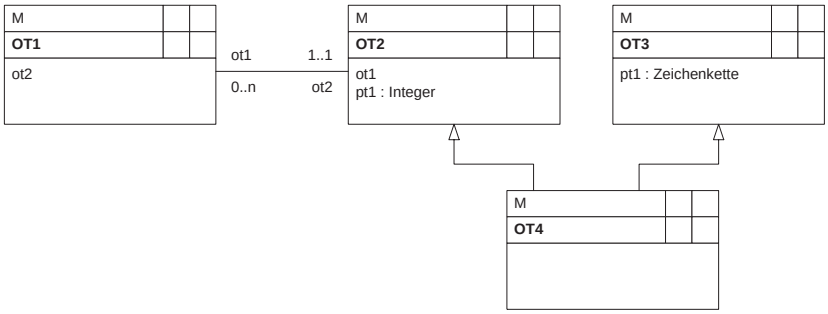



Abbildung 38: Beispiel für eine Mehrfachvererbung

Um dieses und weitere Probleme zu vermeiden, ist die Mehrfachvererbung im $E^3$-Modell ausgeschlossen. Die Fälle, in denen eine solche Mehrfachvererbung notwendig ist, werden durch eine entsprechende Abbildung mittels einer Assoziation substituiert. Beim Auflösen einer Mehrfachvererbung muss eine Entscheidung getroffen werden, welche Vererbungsbeziehungen aufgelöst werden sollen. Die Abbildungen 39 und 40 zeigen beide Möglichkeiten. Gegebenenfalls kann der Superobjekttyp (im Fall der Abbildungen *OT3* bzw. *OT2*) entfallen, zu dem die Vererbungsbeziehung aufgelöst wird. Dies muss aber nicht immer der Fall sein. Soll der Objekttyp bestehen bleiben, so wird er unverändert inklusive aller Assoziationen (ohne Vererbungsbeziehung) mit in die überführte Variante übernommen.

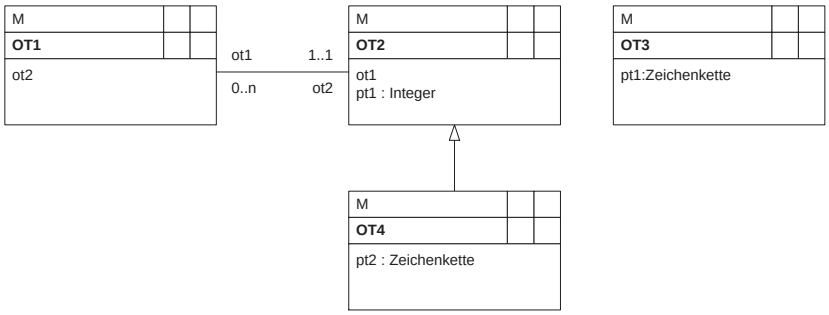



Abbildung 39: Auflösung der Mehrfachvererbung Variante 1

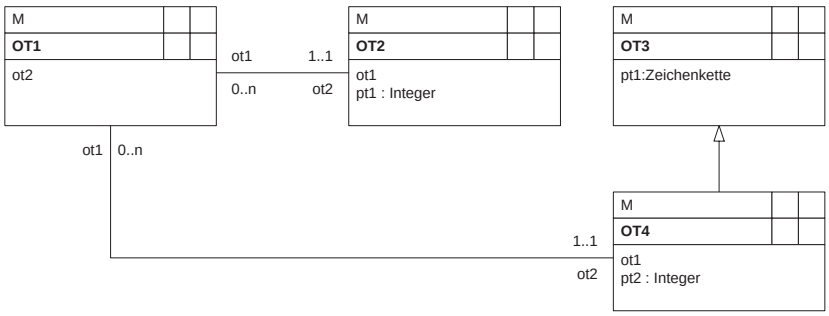



Abbildung 40: Auflösung der Mehrfachvererbung Variante 2

### 6.2.11 Vorgangsebene

In den Kapiteln 6.1.4 bis 6.1.8 wurden mit den Elementen von Vorgehensmodellen diejenigen Konzepte vorgestellt, für die in diesem Abschnitt die Beziehungen untereinander definiert werden. Besonders geeignet erscheint zu diesem Zweck eine Anpassung der Activity Diagrams der UML,[113] nicht zuletzt aufgrund von deren Verbreitungsgrad. In Tabelle 7 wird, soweit dies ohne Erweiterung möglich ist, jedem Konstrukt eines Vorgehensmodells ein entsprechendes Element von Activity Graphs zugeordnet werden. Keine direkte Entsprechung besitzen demnach der Zieltyp, der Aufgabenobjekttyp, der Aufgabenträgertyp und der Sachmitteltyp. Für jedes dieser Elemente wird eine separate Klasse gebildet, die also keine bestehende Klasse des UML-Metamodells für Activity Graphs spezialisiert.

Für den Aufgabentyp und Gruppierungstyp wurden mit dem ActionState bzw. Partition Elemente des Activity Graphs identifiziert, die ihnen von ihrer Bedeutung her entsprechen, jedoch eine Erweiterung um spezielle Attribute benötigen. Die Beziehungen der Elemente der Vorgangsebene untereinander und zu denen der Typebene des E$^3$-Modells sind dem Anhang 11.1 zu entnehmen.

[113] Die vollständige Definition von Syntax (vgl. [Obje00b], S. 3-151ff) und Semantik (zur Semantik der Zustandsautomaten vgl. [Obje00b], S. 2-129ff, zur Semantik der Activity Graphs vgl. [Obje00b], S. 2-160ff) der Activity Diagrams wird als bekannt vorausgesetzt.

| $E^3$-Element der Vorgangsebene | Activity Graph Element |
|---|---|
| Aufgabentyp | ActionState |
| Zieltyp | – |
| Artefakttyp | ClassifierInState |
| Aufgabenobjekttyp | – |
| Verrichtungstyp | Action |
| Aufgabenträgertyp | Swimlane |
| Sachmitteltyp | – |
| Vorbedingungstyp | Guard an eingehender Transition |
| Nachbedingungstyp | Guard an ausgehender Transition |
| Vorgehensmodell | ActivityGraph |
| Gruppierungstyp (Kriterium: organisatorisch) | Partition |

Tabelle 7: $E^3$-Elemente der Vorgangsebene in Activity Graphs

### 6.2.12 Regeln

Das vorgestellte Metamodell des $E^3$-Modells soll in diesem Abschnitt um Regeln erweitert werden, welche die Konsistenz der abgebildeten Metamodelle sichern. Ein Teil der hier vorgestellten Regeln wurde bereits in den vorhergehenden Abschnitten angedeutet. Alle strukturellen Eigenschaften des $E^3$-Modells und Vorgehensweisen bei der Meta-Modellierung können als Regel formuliert werden. Auf diesen Anspruch wird jedoch an dieser Stelle verzichtet, da dieser bereits durch die semiformale Spezifikation in der $E^3$-Notation abgedeckt wird. Vielmehr werden ausgewählte Beziehungen näher beschrieben, von denen angenommen werden muss, dass eine Beschreibung in der $E^3$-Notation nicht zum notwendigen Verständnis führt.

ÖVERGARD stellt fest, dass alle gängigen Modellierungstechniken eine gut spezifizierte grafische Syntax besitzen, die Semantik aber nicht exakt definiert ist. Dadurch können Modelle von verschiedenen Lesern unterschiedlich interpretiert werden (vgl. [Över00], S. 193). Er leitet daraus die Notwendigkeit einer eindeutigen Semantikdefinition ab. Seiner Meinung nach muss die verwendete Sprache solcher Definitionen einfach verständlich und rigoros genug sein, um Mehrdeutigkeiten zu vermeiden (vgl. [Över00], S. 194). Dementsprechend wird die Spezifikation durch eine verbalisierte Form ergänzt. Dies trägt zur Richtigkeit der Sprachanwendung bei und folgt damit dem Grundsatz der Sprachadäquanz.

**R1** Alle $E^3$-Elemente sowie Pakete tragen einen Namen (Benennung des Elements) und haben ggf. Verweise auf andere $E^3$-Elemente.

Die bisherige Spezifikation des $E^3$-Modells birgt Quellen für Inkonsistenzen auf der Objektebene und den View- und Präsentationsebenen. Existiert ein Metamodell *M1* und dazu ein Objekttyp *OT1*, sowie ein Metamodell *M2* mit einem Objekttypen *OT2* und zu *M1* ein Viewtyp *VT1* mit einem Viewobjekttyp *VOT1*, so muss gesichert werden, dass *VOT1* mit *OT1* aus *M1* in Beziehung steht und nicht etwa *OT2* aus *M2* zugeordnet wird. Daraus ergeben sich folgende Regeln:

**R2** *V* sei das einem E$^3$-Element vorgeordnete Element, *Ü* sein übergeordnetes, so muss das dem *Ü* vorgeordnete Element dem *V* übergeordneten Element entsprechen.

Wie bereits in Abschnitt 6.2.10 ausgeführt, kann zwischen Objekttypen eine Vererbungsbeziehung definiert werden. Für diese gelten die folgenden Konsistenzsicherungsregeln. Die erste Regel schließt eine rekursive Vererbungsbeziehung aus:

**R3** Die Menge aller Superobjekttypen *SOT*, die einem Objekttypen *OT* direkt oder indirekt mittels einer Vererbungsbeziehung zugeordnet wurden, darf nicht den *OT* selbst enthalten.

Die folgende Regel definiert die Beziehung vererbter Propertytypen zur Viewebene:

**R4** Existieren zu einem Objekttypen *OT1* ein oder mehrere Propertytypen *PT* und zu *OT1* ein Viewobjekttyp *VOT1*, so sind alle dem *VOT1* zugeordneten Viewpropertytypen *VPT* einem Propertytypen von *OT* bzw. einem Propertytypen eines Superobjekttypen von *OT* zuzuordnen. In diesem Fall wird somit die Menge der zuordenbaren E$^3$-Elemente für einen Viewpropertytypen, welche aus Regel 3 folgen, entsprechend erweitert.

Wenn von einem Objekttypen geerbt wird, so dehnt sich dessen Namensraum auf die für die erbenden Objekttypen definierten Propertytypen aus. Hiermit wird die „Überschreibung“ von Propertytypen durch erbende Objekttypen ausgeschlossen:

**R5** Ein Propertytyp besitzt einen eindeutigen Namen im Namensraum des zugehörigen Objekttypen und dessen direkt oder indirekt durch eine Vererbung zugeordneten Superobjekttypen.

**R6** Jedem Objekttypen kann höchstens ein Superobjekttyp direkt zugeordnet werden.

Eine Präsentationsebene stellt alle Inhalte der ihr vorgeordneten Viewebene dar. Deshalb wird für die Typen und Instanzen der Präsentationsebene die folgende Regel definiert:

**R7** Wird einem Viewtypen ein Präsentationstyp nachgeordnet, müssen alle dem Viewtypen zugeordneten Viewobjekt- und Viewpropertytypen durch dem Präsentationstypen zuzuordnende Präsentationsobjekt- und Präsentationspropertytypen grafisch dargestellt werden.

**R8** Wird einer View eine Präsentation nachgeordnet, müssen alle der View zugeordneten Viewobjekte und Viewproperties durch der Präsentation zuzuordnende Präsentationsobjekte und Präsentationsproperties grafisch dargestellt werden.

Diese Forderungen enthalten nicht, dass jedem Viewobjekttypen (bzw. Viewobjekt) ein Präsentationsobjekttyp (bzw. Präsentationsobjekt) nachzuordnen ist, wenn seinem Viewtypen ein Präsentationstyp nachgeordnet wurde. Diese Forderung entspricht zwar der Regel, ist aber zu streng formuliert. Vielmehr kann ein Viewobjekttyp beispielsweise auch durch einen Präsentationspropertytyp dargestellt werden.

Die letzte Regel bezieht sich auf die Darstellung der in Abschnitt 6.2.8 beschriebenen referenzierten Objekttypen:

**R9** Ein referenzierter Objekttyp enthält zusätzlich die Angabe des Paketes, in welchem er definiert wird.

## 6.3 Notation

### 6.3.1 Elemente des E$^3$-Modells

Zunächst wird eine Abbildung für alle E$^3$-Elemente vorgenommen. Dazu wird die von der UML bekannte Notation einer Klasse verwendet und entsprechend erweitert (vgl. Abbildung 41). Senkrecht werden die vertikal zugeordneten Elemente im E$^3$-Modell beschrieben. An oberster Position finden sich damit die Benennung für Metamodell (Feld 1), Viewtyp (Feld 2) und Präsentationstyp (Feld 3). Darunter werden die Namen des Objekttyps (Feld 4), Viewobjekttyps (Feld 5) sowie Präsentationsobjekttyps (Feld 6) notiert. Den Abschluss bildet das Feld 7. In diesem werden die einem Element zugehörigen Eigenschaften eingetragen (z. B. Propertytypen bei der Darstellung eines Objekttypen oder die Angabe einer grafischen Darstellung bei Präsentationsobjekttypen).

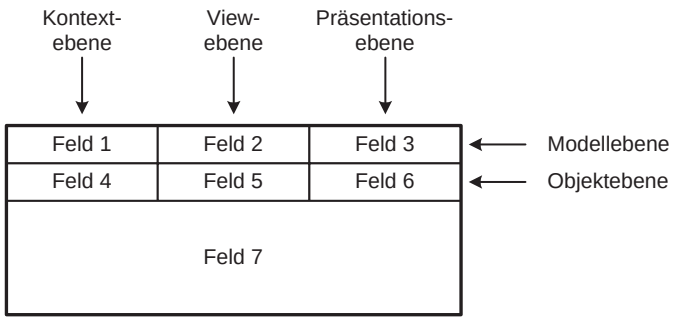



Abbildung 41: Grafische Darstellung eines E$^3$-Elements

Die Bezüge der einzelnen Elemente zueinander werden nur minimal aufgeführt. So werden zu einem Viewobjekttyp lediglich der zugehörige Objekttyp, der Viewtyp sowie die Viewpropertytypen notiert. Auf die Angabe des Metamodells kann verzichtet werden, da diese aus der Definition des dem Viewobjekttypen zugeordneten Objekttypen bzw. Viewtypen hervorgeht. Die Angabe einer grafischen Darstellung für Präsentationsobjekt- und Präsentationspropertytypen erfolgt in Textform. Die aufgeführte Benennung bezieht sich stets auf eine Tabelle, wo alle Namen der Grafiken des Metamodells mit der zugehörigen grafischen Darstellung aufgeführt sind. Beispiele für die Darstellung der Elemente enthält Abbildung 42.

Referenzierte E$^3$-Elemente aus anderen Paketen (vgl. Abschnitt 6.2.8) werden wie ihr Original dargestellt, im Feld 7 wird jedoch der Verweis auf das Paket angegeben, in dem das Element definiert wurde.

Die Propertytypen, Viewpropertytypen und Präsentationspropertytypen nehmen in dieser Notation eine Sonderstellung ein. Sie werden wie Attribute der UML behandelt und können entsprechend zu Objekttypen, Viewobjekttypen und Präsentationsobjekttypen notiert werden. Für Propertytypen muss zusätzlich ihr Wertebereich, eine Multiplizitätsangabe sowie ein Strukturtyp notiert werden. Die Notation der Propertytypen ist somit entsprechend umfangreich. Da sich bei der Arbeit mit dem E$^3$-Modell häufige Typen der Tripel Multiplizität, Strukturtyp und Wertebereich herausgebildet haben, sollen diese verkürzt dargestellt werden, um die Lesbarkeit der Metamodelle zu erhöhen. Es werden zwei Vereinfachungsregeln formuliert, in allen anderen Fällen ist die ausführliche Notation zu verwenden:

1. Diese Verkürzung setzt voraus, dass die Propertytypen einen Wertebereich, bestehend aus anderen E$^3$-Elementen und den Strukturtypen *Menge*, besitzen. Die Multiplizität muss durch die entsprechenden Assoziationsbeziehungen (siehe Abschnitt 6.3.3) spezifiziert werden. Diese Propertytypen müssen nicht in Feld 7 aufgeführt werden.

2. Bei Verwendung des Strukturtypen *Menge* in Kombination mit der Multiplizität *(1,1)* kann auf die Angabe beider Elemente verzichtet werden. Diese Propertytypen werden nur durch ihren Namen sowie den Wertebereich notiert.

Für einen Viewpropertytypen muss zusätzlich zum Viewtypen ebenfalls der zugehörige Propertytyp angegeben werden. Entscheidend ist dabei lediglich die Existenz eines solchen Viewpropertytypen. Notiert wird nur der Name des Propertytypen, der dem Viewpropertytypen zugeordnet wird: Präsentationspropertytypen werden im Feld 7 ihres Präsentationsobjekttypen aufgeführt. Dieses enthält zusätzlich eine Beschreibung der grafischen Darstellungsform. In Abbildung 42 werden Beispiele der bereits diskutierten Elemente dargestellt. Die Abbildung zeigt zusammenfassend die Darstellung aller Elemente des E$^3$-Modells.

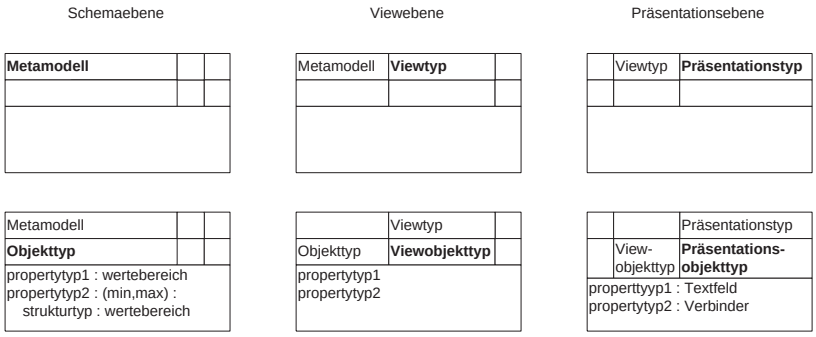



Abbildung 42: Beispiel für die Abbildung von E$^3$-Elementen

### 6.3.2 Pakete

Pakete werden in Anlehnung an die UML-Notation dargestellt. Die Abbildung 43 zeigt ein Metamodell *Metamodell*, das in Paket *paket2* definiert wird. Dieses ist neben dem Objekttypen *Objekttyp* Bestandteil von Paket *paket1*.

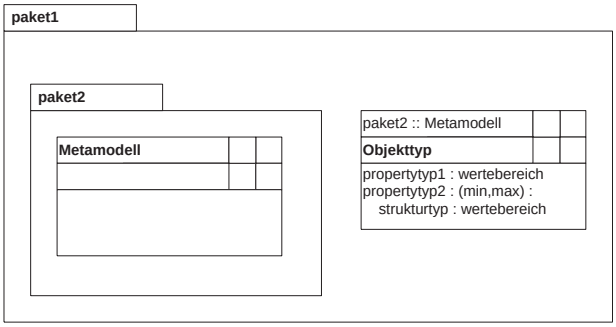



Abbildung 43: Beispiel für Pakete im E$^3$-Modell

### 6.3.3 E$^3$-Wertebereich

Die Darstellung von E$^3$-Wertebereichen[114] erfolgt mit einer an die UML angelehnten Notation. Dort wird eine Beziehung zwischen Objekten als Assoziation bezeichnet. Auf jeder Seite der Assoziation können Rollennamen dazu verwendet werden, genauer zu beschreiben, welche Rolle die jeweiligen Objekte in der Beziehung einnehmen (vgl. [Oest01], S. 263). Im Falle des E$^3$-Wertebereichs repräsentiert eine Rolle entsprechend den Propertytypen, auf den sich die Beziehung bezieht bzw. dessen Wertebereich eingeschränkt werden soll. Für jede angegebene Rolle wird darüber die entsprechende Multiplizität notiert. Beide Enden der Kante müssen mit einem gültigen E$^3$-Element verbunden sein.

[114] Muster für diese werden z. B. mit dem Abschnitt 7.1.1 als Assoziation im E$^3$-Modell beschrieben.

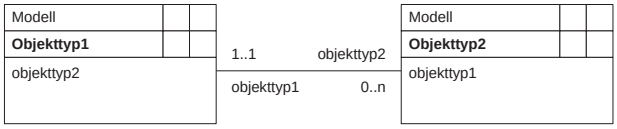



Abbildung 44: Notation eines E$^3$-Wertebereichs

Die Abbildung 44 demonstriert den Umgang mit Rollen und Multiplizitäten. Dabei ist abgebildet, dass ein Property des Propertytypen *Objekttyp1* genau einem Property des Propertytypen *Objekttyp2* zugeordnet werden muss.

Da diese Notation einen E$^3$-Wertebereich beschreibt, ist an einer solchen grafischen Beziehung mindestens ein Propertytyp beteiligt. Deshalb muss mindestens eine Rolle der Kante belegt sein. An diesem Ende ist ebenfalls eine Multiplizität einzutragen. Ist ein Ende einer Assoziation nicht durch eine Rolle belegt, so erfolgt auch keine Angabe der Multiplizität. Für den Fall, dass an einem Ende keine Rolle eingetragen wird, liegt demnach ein Aggregations- oder Detailmuster (siehe dazu die Abschnitte 7.1.3 bzw. 7.1.4) vor.

### 6.3.4 Vererbungsbeziehungen

In Anlehnung an die Darstellung in der UML werden der Superobjekttyp und die von ihm erbenden Subobjekttypen mittels einer Kante mit offener Pfeilspitze visualisiert. Diese führt vom Subobjekttypen zum Superobjekttypen.

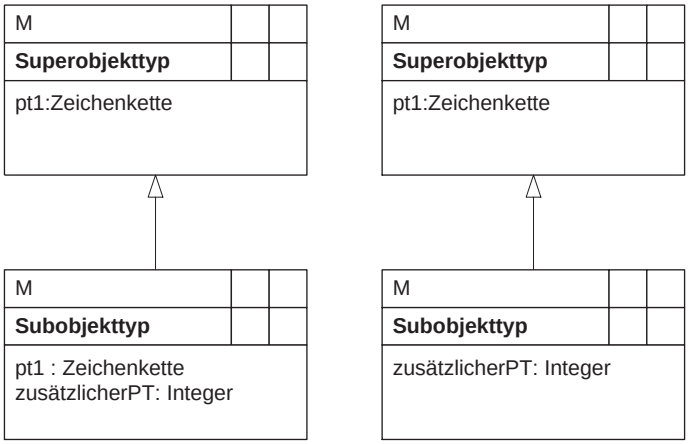



Abbildung 45: Beispiel einer Vererbungsbeziehung

Die vom Subobjekttypen geerbten Propertytypen können im Subobjekttypen dargestellt werden. Dies dient auf der einen Seite der Klarheit, kann aber bei zahlreichen geerbten Propertytypen zu unübersichtlichen Grafiken führen. Die Abbildung 45 stellt beide Alternativen dar.

### 6.3.5 Vorgangsebene

Die Darstellung der Vorgangsebene orientiert sich weitgehend an den Activity Diagrams der UML (vgl. [Obje00b], S. 3-151). Wesentliche Bestandteile der Graphen sind die Aufgabentypen, die in der ausführlichen Darstellung gemeinsam mit dem Sachmitteltypen, dem Zieltypen und dem Aufgabenträgertypen in einem Rechteck mit abgerundeten Ecken dargestellt werden. In der verkürzten Form wird in diesem nur der Name aufgeführt. Die Abbildung 46 zeigt dies beispielhaft.

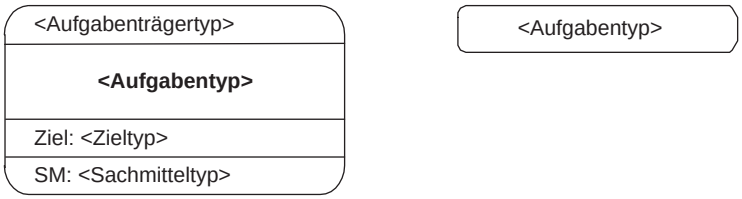



Abbildung 46: Aufgabentyp

Die zweite Knotenart stellen die Artefakttypen in einem bestimmten Zustand dar, welche in Anlehnung an die Darstellung von Objekten in einem bestimmten Zustand (ObjectFlowState) innerhalb eines Rechtecks unter Angabe des Zustands abgebildet werden. Die Abbildung 47 gibt ein Beispiel.


Abbildung 47: Zustandsbehafteter Artefakttyp

Verbunden werden die Aufgabentypen und zustandsbehafteten Artefakttypen mittels Transitionen. Dabei stehen in Anlehnung an klassische Datenflussansätze entweder Objekt- oder Kontrollflüsse zur Verfügung. Eine Aufspaltung des Flusses durch eine Entscheidung oder das parallele Auftreten einer Transition führen zu nebenläufigen, von einander unabhängigen *Regionen*. Solche Verzweigungen können nur durch eine Zusammenführung wieder synchronisiert werden. Bei Verzweigungen aufgrund von Entscheidungen ist als Symbol ein um 45 Grad gedrehtes Quadrat zu wählen und die Transitionen sind mit den entsprechenden Bedingungen zu beschriften. Alternativ werden die beschrifteten Transitionen direkt an den erzeugenden Aufgabentypen angetragen. Die Parallelisierung und Synchronisierung von Abläufen erfolgt über *Gabeln*. Abbildung 48 stellt alle Elemente dar.

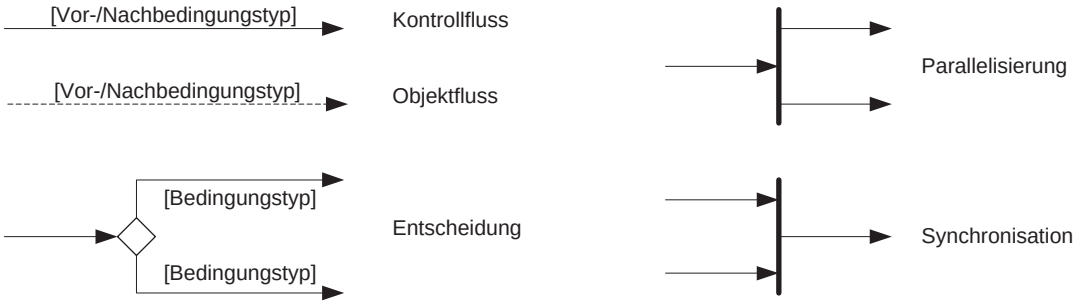



Abbildung 48: Verbindungen von Aufgabentypen

Eine Möglichkeit zur Gruppierung von Knoten innerhalb der Grafen bietet eine Partition: „A partition is a mechanism for dividing the states of an activity graph into groups.“ ([Obje00b], S. 2-163) Dabei wird ausdrücklich auf die häufige Entsprechung von Partitionen und organisatorischen Einheiten innerhalb eines Unternehmens hingewiesen. Zur grafischen Darstellung werden *Swimlanes* verwendet. Dabei handelt es sich um parallel verlaufende Bahnen, in die die zugehörigen Zustandsknoten platziert werden. Abbildung 49 zeigt anhand eines Anwendungsbeispiels die Verwendung von Swimlanes.

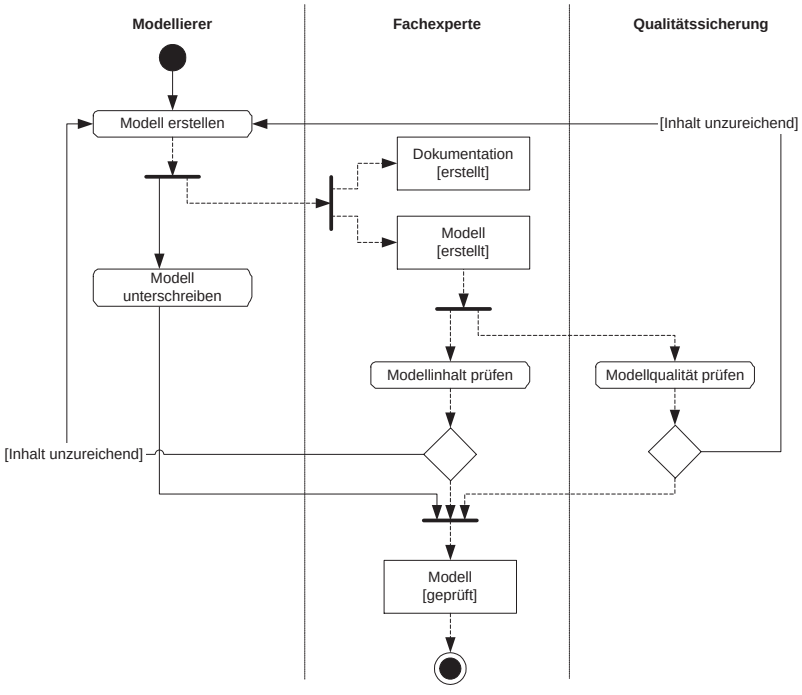



Abbildung 49: Beispiel für Partitionen

# 7 Wiederverwendungsansätze

Bei der Entwicklung von Softwaresystemen ist das Ziel der Wiederverwendungsansätze die Vermeidung von Doppelarbeiten durch das Erkennen von Gemeinsamkeiten bei ähnlichen Aufgaben (vgl. [Diet02], S. 14). Die $E^3$-Methode wird in diesem Abschnitt mit einer ähnlichen Zielstellung um Typisierungsmuster und ein Referenzmetamodell ergänzt. Erstere zielen insbesondere auf die Generalisierung wiederkehrender Strukturen innerhalb der Darstellungstechniken von Methoden ab. Letzteres dient vor allem als Ausgangspunkt bei ihrer situationsangepassten Gestaltung.

## 7.1 Muster

Allgemein wird unter einem **Muster** eine wiederkehrende Problemlösung verstanden. Um diese zu ermöglichen, ist es notwendig, zunächst von den konkreten Problemlösungen zu abstrahieren und die gemeinsamen Faktoren, also die Essenz der Lösung, zu extrahieren. „Stets werden abstrakte Lösungen sich wiederholenden Problemen gegenübergestellt." ([Diet02], S. 93) Dieses Denken in Problemlösungen ist weit verbreitet in der Architektur, Volkswirtschaft und auch der Softwareentwicklung (vgl. [Bus+96], S. 3). Mit Hilfe von Mustern ist es möglich, dass existierende Wissen und die Erfahrungen effizient zu dokumentieren und gute Lösungen einer breiten Öffentlichkeit zugänglich zu machen (vgl. [Hein98], S. 22f). Neben diesen qualitativen Aussagen ist eine Definition für den eindeutigen Gebrauch des Begriffs des Musters unerlässlich. Sehr allgemein definiert ALEXANDER Muster als eine dreiteilige Regel, welche eine Beziehung zwischen einem bestimmten Kontext, einem Problem und einer Lösung ausdrückt (vgl. [Alex79], S. 247). Aus ökonomischen Gründen lohnt die Erstellung eines Musters nur, wenn es mehrfach eingesetzt und somit die Analyse- und Designphase erheblich verkürzt werden kann.

Muster werden in der Systementwicklung insbesondere während des Designs und der Implementierung eingesetzt, wobei sie vereinzelt auch eine Unterstützung für die Analyse und den Test der Systeme bieten. Entsprechend sind auch deren Ziele zu differenzieren. Ein sehr wichtiges ist während der gesamten Systementwicklung und insbesondere in der Modellierung die Bereitstellung eines einheitlichen Sprachraumes. Durch die in den Beschreibungen der Muster verwendeten Begriffe werden Missverständnisse zwischen den Projektmitarbeitern vermieden und somit die Kommunikation unter den Beteiligten effektiviert. Durch die einheitliche Bedeutung der Begriffe werden langwierige Erläuterungen zum Verständnis der gewählten Lösungen überflüssig. Des Weiteren erlauben Muster die Komplexitätsreduktion großer und heterogener Systeme durch den Einsatz als „mental - building blocks".[115]

[115] In der SE werden dabei die großen Architekturen zunächst aus groben Architekturmustern zusammengesetzt. Diese benötigen zu ihrer Implementierung wiederum kleinere Muster. Durch die rekursive Anwendung der Muster wird die Komplexität der Systemumsetzung schrittweise unter Wahrung des Überblicks reduziert.

Muster dienen häufig als Wissensspeicher für existierende, erprobte Designlösungen. In Verbindung mit ihrer Veröffentlichung ist durch die mehrfache Anwendung der Muster durch verschiedene Entwickler sichergestellt, dass Fehler im Muster schnell erkannt und die vorgegebenen Lösungsstrukturen optimiert werden. Durch die Vielzahl der Entwickler ist es aber auch wahrscheinlich, dass bei Verfügbarkeit neuer Technologien oder Paradigmen gänzlich neue Muster mit wesentlich verbesserten Problemlösungsstrukturen entwickelt werden. Damit ist im Allgemeinen eine hohe Qualität der öffentlich verfügbaren Muster sichergestellt.

Eine wichtige Anforderung ist die Wiederverwendbarkeit der Modelle. Diese erfordert aber oft Abstraktionen und Strukturen, die sich im Rahmen der Systemanalyse nicht oder nur schwer erkennen lassen. Die notwendige Flexibilität wird nur durch mehrfache Iterationen von Analyse und Design erreicht. Muster können den Iterationsbedarf entscheidend verringern, da sie ja bereits erprobte Strukturen repräsentieren. Dabei ist nicht unbedingt eine bessere als die eigene Lösung garantiert, zumindest wird jedoch ein Bewertungsmaßstab bei der Evaluierung vorgegeben.

Durch den Einsatz von Mustern im Design vereinfacht sich aber auch die notwendige Dokumentation besonders bei komplexen Systemen. Muster können ebenfalls als Zusatz für existierende Methoden verstanden werden, die z. B. aufzeigen wie einfache Elemente[116] zu verwenden sind. Es lassen sich aber auch die Gründe für ein bestimmtes Design ablegen, sodass die Entwicklung zunächst nachvollziehbar und anschließend auf ein Problem anwendbar wird. Ein weiteres Ziel ist die Unterstützung der Umformung der Ergebnisse einer Analyse in eine konkrete Implementierung unter Sicherstellung einer möglichst hohen Wiederverwendbarkeit der Ergebnisse. Für die vollständige Unterstützung einer Methode werden aber z. B. zusätzlich Analysepatterns benötigt. In der Programmierung liegt mit der Schulung ein weiteres Einsatzgebiet vor. Anfängern werden bestimmte Muster für oft vorkommende Programmierprobleme vorgegeben, um häufig vorkommende Fehler zu vermeiden und das Erlernen der Syntax zu vereinfachen.

Fasst man die wesentlichen Ziele zusammen, so dienen Muster der:

- Bereitstellung eines einheitlichen Sprachraumes,
- Wiederverwendbarkeit erstellter Ergebnisse,
- Vereinfachung der Dokumentation der Ergebnisse und der
- Umformung der Analyse in eine konkrete Implementierung.

Um die gestellten Ziele zu erreichen und insbesondere eine breite Verwendbarkeit der Muster zu gewährleisten, sind gewisse Mindestanforderungen an den Inhalt der Musterdokumentationen

[116] Beim Umgang mit dem $E^3$-Modell sind dies z. B. Objekt- und Propertytypen.

zu stellen. Aufgrund der unterschiedlichen Schwerpunkte bei der Definition von Mustern finden sich in der Literatur auch unterschiedliche Varianten des Aufbaus von Mustern. BUSCHMANN fordert beispielsweise für die Beschreibung eines Musters:

- *Problembeschreibung*: Diese beinhaltet nach BUSCHMANN: die Essenz des Problems, die zu beachtenden Regeln (constraints), die Lösungsanforderungen und gegebenenfalls wünschenswerte Eigenschaften der Lösung. Zusätzlich sollte die Problembeschreibung eine Menge von Zwängen (forces) enthalten, die das Problem aus mehreren Sichten darstellen und somit das Verständnis der Details erleichtern.

- *Anwendungskontext*: Der Anwendungskontext enthält eine meist allgemein gehaltene Beschreibung der Situationen, in denen das zu lösende Problem auftritt. Eine detaillierte Darstellung des Kontextes ist zumeist schwierig, ein pragmatischer Ansatz könnte deshalb die Auflistung aller Situationen fordern, in denen das Problem bis jetzt auftrat (vgl. [Bus$^+$96], S. 9).

- *Lösungsstruktur*: Die Strukturbeschreibung enthält sowohl statische als auch dynamische Aspekte.

Strukturen, die bei bereits erfolgten Typisierungen mit dem $E^3$-Modell immer wieder auftreten, werden als *Typisierungsmuster* bezeichnet. Aufgrund der Impulse durch den Einsatz von Mustern in der Modellierung, lässt sich ein ähnlich positiver Effekt durch den Einsatz von Mustern der Metamodellierung vermuten. Demnach lässt sich die Qualität der Typisierungen erhöhen und gleichzeitig der Zeitaufwand für deren Erstellung und Erweiterung verringern. Die dafür notwendigen qualitätssichernden Maßnahmen bei der Erzeugung weiterführender Muster können mit Hilfe einer von QUIBELDY-CIRKEL beschriebenen Autorenwerkstatt realisiert werden. Innerhalb von moderierten Gruppensitzungen werden zur Fehlervermeidung neue Muster vorgestellt und diskutiert (vgl. [Quib99], S. 67).

Folgende Bestandteile werden zu einer Musterbeschreibung in dieser Arbeit in den folgenden Kapiteln verwendet:

- ein *aussagekräftiger Name*, der das Muster eindeutig identifiziert
- eine *Problembeschreibung*, die ein wiederkehrendes Problem beschreibt
- ein *Kontext*, in welchem das Muster eingesetzt wird
- eine *Beschränkung*, die angibt wann das Muster nicht verwendet werden soll
- die *Lösung* definiert die Elemente des Musters und deren Beziehungen zueinander
- die *Konsequenzen* aus der Anwendung des Musters

- ein *Beispiel* aus den bisher vorgenommen Typisierungen
- die *verwandten Muster*, die entweder einen ähnlichen Problembezug besitzen oder eine vergleichbare Lösungsstruktur aufzeigen

### 7.1.1 Das Typisierungsmuster „Assoziation“

#### Problem:

Dieses Muster verbindet zwei Objekttypen miteinander. Die Beziehung soll im Kontext eines Metamodells beschrieben werden.

#### Kontext:

Das Muster wird immer dann eingesetzt, wenn zwei Objekttypen grafisch und inhaltlich in Beziehung gesetzt werden.

#### Beschränkung und Konsequenzen:

Das Muster kann nicht eingesetzt werden, wenn die Beziehung mit weiteren Eigenschaften näher beschrieben werden soll.

#### Lösung:

Die Beziehung zwischen den beiden Objekttypen wird über jeweils einen Propertytypen realisiert. Diese Propertytypen besitzen einen Wertebereich, der den jeweils anderen Propertytypen enthält (siehe dazu Abbildung 50).

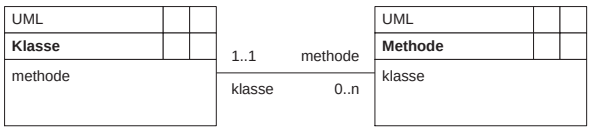



Abbildung 50: Das Assoziationsmuster

#### Beispiel:

Das Muster ist eines der am häufigsten eingesetzten, da es u. a. ein Bestandteil des Kantenmusters ist. Im Beispiel aus Abbildung 50 werden die Objekttypen *Klasse* und *Methode* mit diesem Muster verbunden. Deren Instanzen können durch die im Beispiel gesetzten Kardinalitäten folgende wechselseitige Beziehung eingehen: Jede Methode wird genau einer Klasse zugeordnet und jede Klasse kann keine oder beliebig viele Methoden besitzen. Im Anhang 12.2 werden

als weiteres Beispiel die Objekttypen *Organisationseinheit* und *Aufgabenzuordnung* im Paket *System* auf diese Weise miteinander verknüpft.

#### verwandte Muster:

Assoziationspool, Aggregation, Kante

### 7.1.2 Das Typisierungsmuster „Assoziationspool“

#### Problem:

Das Muster verbindet einen Objekttypen mit mehreren anderen. Dabei wird nicht, wie im Assoziationsmuster gefordert, für jede dieser Beziehungen ein Propertytyp definiert, sondern für nur einen Propertytypen ein Wertebereich definiert, der sich über mehrere Objekttypen erstreckt.

#### Kontext:

Das Muster wird immer dann angewendet,wenn ein Objekttyp grafisch und inhaltlich mit mehreren anderen in Beziehung gesetzt werden soll.

#### Beschränkungen und Konsequenzen:

Es gelten die Einschränkungen des Assoziationsmusters. Zusätzlich kann aufgrund der Tatsache, dass im Kontext nur ein Propertytyp den Pool definiert, jede grafische Präsentation des Property nur den gesamten Pool abbilden.

#### Lösung:

**Alternative 1:** Zur Abbildung des Pools wird ein „Hilfsobjekttyp“ definiert. Dieser bildet den Superobjekttypen für alle Objekttypen im Pool. Er wird über das Assoziationsmuster mit dem Objekttypen verbunden, mit dem alle Objekttypen im Pool in Beziehung stehen sollen (siehe dazu Abbildung 51). Diese Alternative ist der Alternative 2 grundsätzlich vorzuziehen, da sie den Pool direkter verdeutlicht. Ihre Verwendung ist jedoch nicht in jedem Fall möglich.

**Alternative 2:** Wenn für die Objekttypen im Pool bereits andere Superobjekttypen definiert wurden, werden zur Vermeidung einer Mehrfachvererbungsbeziehung (siehe Abschnitt 6.2.10) die Objekttypen direkt zugeordnet. Für den Assoziationspool des einen Objekttypen wird ein Propertytyp definiert, dessen Wertebereich die Propertytypen der anderen Objekttypen umspannt. Diese definieren einen Wertebereich, der den Assoziationspool enthält (siehe dazu Abbildung 52).

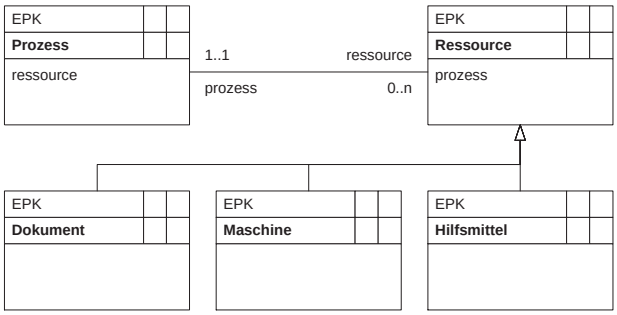



Abbildung 51: Das Assoziationspoolmuster, Alternative 1

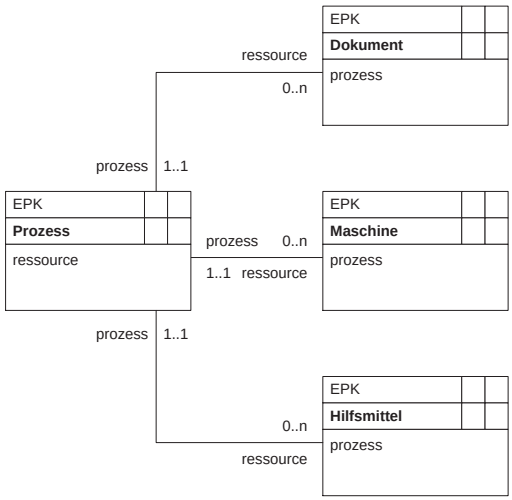



Abbildung 52: Das Assoziationspoolmuster, Alternative 2

### Beispiel:

Einem Objekttypen *Prozess* werden in beiden Alternativen des Musters zu dessen Durchführung die Objekttypen *Dokument*, *Maschine* und *Hilfsmittel* als Ressourcen zugewiesen. Letztere beziehen sich in den Beispielen aus den Abbildungen 51 und 52 immer auf genau einen Prozess. Ein weiteres Beispiel zur Alternative 1 des Musters bildet die Gliederung einer Aufgabe nach *Ausführungs-*, *Leitungs-*, *Planungs-*, *Kontroll-* und *Verwaltungsaufgabe* im Paket *Aufgabengliederungsparadigma* (siehe dazu Anhang 12.2.2.1).

### verwandte Muster:

Assoziation

### 7.1.3 Das Typisierungsmuster „Aggregation“

#### Problem:

Dieses Muster verbindet zwei Objekttypen miteinander. Die Beziehung soll im Kontext eines Modells beschrieben werden. Der als *Aggregat* bezeichnete Objekttyp kann zu seiner näheren Beschreibung auf andere Objekttypen verweisen, die beschreibenden Objekttypen enthalten keine Eigenschaft, die auf ihr Aggregat verweist.

#### Kontext:

Das Muster wird immer dann angewendet, wenn ein Objekttyp grafisch und inhaltlich mit mehreren anderen in Beziehung gesetzt werden soll.

#### Beschränkung und Konsequenzen:

Aufgrund der Zuordnung in nur einer Richtung kann das Aggregat zu einem beschreibenden Objekttypen nur durch eine Abfrageoperation auf alle potentiellen Aggregate ermittelt werden.

#### Lösung:

Das Aggregat wird als Objekttyp abgebildet. Diesem wird ein Propertytyp zugewiesen, dessen Wertebereich die zu aggregierenden Objekttypen umfasst. In umgekehrter Richtung enthalten die zu aggregierenden Objekttypen keine Eigenschaft, die auf die Beziehung hinweist (siehe dazu Abbildung 53).

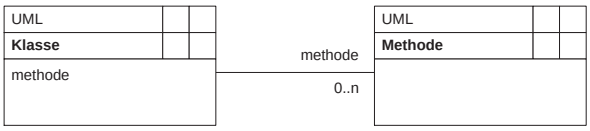



Abbildung 53: Das Aggregationsmuster

#### Beispiel:

Innerhalb der UML können einer Klasse mehrere Methoden zugeordnet werden. Sowohl das Konzept der Methode als auch das Konzept der Klasse werden als Objekttypen abgebildet. Für die Klasse wird ein Propertytyp definiert, dessen Wertebereich den Objekttypen für die Methode enthält (siehe dazu Abbildung 53). Im Anhang 12.2.4.2 wird das Muster zur Verbindung der Objekttypen *Wiederholung* mit *Bedingung* und *Verzweigung* mit *Alternative* verwendet.

**verwandte Muster:**

Detail

### 7.1.4 Das Typisierungsmuster „Detail“

**Problem:**

Einem Objekttypen wird eine Ebene der Beschreibung zugeordnet.

**Kontext:**

In vielen Metamodellen wird die Komplexität einer Beschreibung durch das Prinzip der Hierarchisierung (siehe hierzu Abschnitt 3.2) bewältigt. Einem Objekttypen werden weitere Sichten oder Darstellungen zugeordnet. Präsentations- und Viewtypen lassen sich dabei mehreren Objekttypen unterordnen.

**Beschränkungen und Konsequenzen:**

Da sowohl Viewtypen als auch Präsentationstypen keine Referenzen auf Objekttypen ablegen können, ist die Beziehung nur einer Richtung navigierbar. Ähnlich wie im Aggregationsmuster lassen sich die Objekte, welche einer Präsentation bzw. einer View zugeordnet wurden nur über Abfrageoperationen ermitteln. Dass ein View- oder ein Präsentationstyp nur einem Objekttypen zugeordnet werden kann, lässt sich über Wertebereiche beschreiben. Dass deren Instanzen (also eine View oder eine Präsentation) sich nur einem Objekt zuordnen lassen, muss bei Bedarf über Regeln abgebildet werden.

Die Beziehung zwischen dem Objekttypen und seiner detaillierten Beschreibung kann (wie beim Assoziationsmuster) nicht näher mit Eigenschaften versehen werden.

**Lösung:**

Das detailliert zu beschreibende Konzept wird als Objekttyp abgebildet. Diesem wird ein Propertytyp zugewiesen, dessen Wertebereich den entsprechenden View- oder Präsentationstyp umfasst (siehe dazu Abbildung 54).

**Beispiel:**

Für die Strukturierte Analyse kann die Zerlegung der Prozesse in Datenflussdiagrammen mit diesem Muster abgebildet werden. Daneben wird das Muster u. a. dazu verwendet, die detaillierte Spezifikation von Anwendungsfällen in Anwendungsfalldiagrammen zu ermöglichen (siehe Paket *Anwendungsfalldiagramm* des Datenflussparadigmas im Anhang 12.2.4.1).

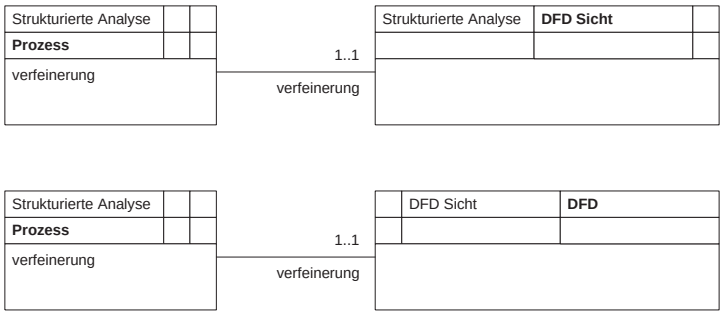



Abbildung 54: Das Detailmuster

### verwandte Muster:

Aggregation

## 7.1.5 Das Typisierungsmuster „Kante“

### Problem:

Das Kantenmuster bildet die Verbindung zwischen zwei Objekttypen über einen dritten, im Folgenden als Kante bezeichneten Objekttypen ab. Die Verbindung wird im Kontext des Metamodells beschrieben.

### Kontext:

Das Muster wird vor allem dann angewendet, wenn der verbindende Objekttyp grafisch als Kante dargestellt wird.

### Beschränkung und Konsequenzen:

Dieses Muster behebt die Einschränkungen des Assoziationsmusters, dass eine Beziehung nicht mit weiteren Eigenschaften beschrieben werden kann. Da die Beziehung selbst ein Objekttyp ist, können für diese weitere Propertytypen definiert werden. Aufgrund der Realisierung der Verbindung über einen Assoziationspool gelten zusätzlich die für dieses Muster beschriebenen Einschränkungen.

### Lösung:

**Alternative 1:** Die Knoten werden dem Objekttypen für die Kante als Assoziationspool zugeordnet. Die Kante erhält einen Propertytypen, der die Verbindung zum „Hilfsobjekttypen“ *Knoten* realisiert (siehe Abbildung 55). Diese Alternative ist der Alternative 2 vorzuziehen. In

Analogie zum Muster Assoziationspool gilt jedoch auch hier, dass sie zur Vermeidung einer Mehrfachvererbung nicht immer eingesetzt werden kann.

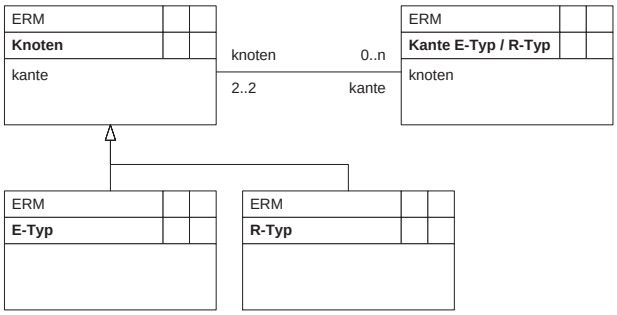



Abbildung 55: Das Kantenmuster, Alternative 1

**Alternative 2:** Unter Anwendung der Alternative 2 des Musters *Assoziationspool* erhält die Kante einen Propertytypen, der die Verbindung zu den anderen Objekttypen realisiert (siehe Abbildung 56).

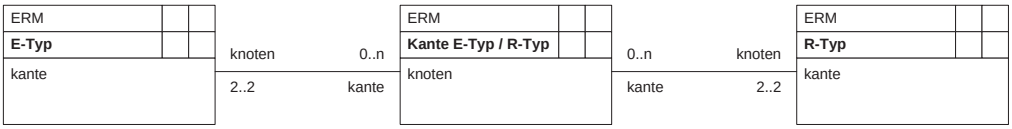



Abbildung 56: Das Kantenmuster, Alternative 2

### Beispiel:

Das Muster findet immer Anwendung, wenn die Richtung der Kante (grafisch z. B. durch einen Pfeil symbolisiert) unerheblich ist. Entsprechend kann die Kante im Entity-Relationship Model (ERM), die einen Entity- mit einem Relationship-Typen verbindet, mit dem Kantenmuster abgebildet werden.

### verwandte Muster:

Assoziation

## 7.1.6 Das Typisierungsmuster „Gerichtete Kante“

### Problem:

Das Kantenmuster bildet die Verbindung zwischen zwei Objekttypen über einen dritten Objekttypen (der als Kante bezeichnet wird) ab. Die Verbindung wird im Kontext des Metamodells beschrieben. Die Kante enthält zusätzliche Informationen über die Richtung der Verbindung.

### Kontext:

Das Muster wird vor allem dann angewendet, wenn der verbindende Objekttyp grafisch als Kante dargestellt wird. Nicht immer muss die Richtung mit Pfeilen zum Ausdruck gebracht werden (wie bei einigen Kanten im SERM), vielmehr kann der Kontext bereits die Abbildung einer Richtung implizieren.

### Beschränkung und Konsequenzen:

Dieses Muster behebt die Beschränkung des Kantenmusters, die Richtung nicht abbilden zu können.

### Lösung:

Wie beim Kantenmuster wird für die Verbindung ein Objekttyp definiert, der als Kante bezeichnet wird. Dieser erhält für jeden Objekttypen, mit dem er in Beziehung treten kann, einen Propertytypen mit entsprechendem Wertebereich (siehe dazu Abbildung 57).

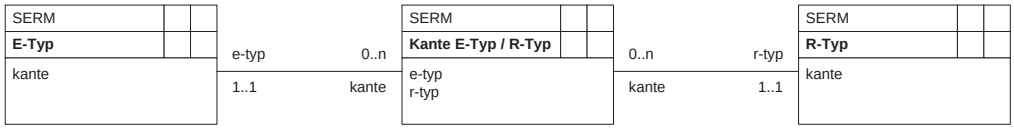



Abbildung 57: Das Muster „Gerichtete Kante“

### Beispiel:

Anders als im ERM werden im Strukturierten Entity-Relationship Modell (SERM) die Entity- und Relationship-Typen über gerichtete Kanten miteinander verbunden. Grafisch wird dem vor allem durch die Anordnung der Typen von links nach rechts zur Verdeutlichung der Abhängigkeitsbeziehungen Rechnung getragen. Der Objekttyp zur Abbildung der Kante besitzt zwei Propertytypen, um die Beziehungen zum Entity- und Relationship-Typen abzubilden. Im Anhang 12.2.3.1 wurde das Muster zudem bei der Verbindung der Objekttypen *Generalisierbare Objektklasse* und *Generalisierungsknoten* über den Objekttypen *Spezialisierungslinie* verwendet.

### verwandte Muster:

Kante

### 7.1.7 Zusammenfassung

Die fünf in diesem Abschnitt vorgestellten Muster haben ihre Wirksamkeit insbesondere bei der Erstellung des im nächsten Abschnitt beschriebenen Referenzmetamodells unter Beweis gestellt. Die Tabelle 8 fasst die Typisierungsmuster dieses Abschnitts unter Nennung ihres Namens und einer kurzen Aufführung von Kontext und Lösung zusammen.

| Name | Kontext | Lösung |
|---|---|---|
| Assoziation | Verbindung zweier Objekttypen | Propertytypen mit „wechselseitigen“ Wertebereichen |
| Assoziationspool | Verbindung eines Objekttypen mit mehreren anderen | Propertytyp mit Propertytypen als Wertebereich |
| Aggregation | Verbindung zweier Objekttypen | Propertytyp mit Objekttyp als Wertebereich |
| Detail | Verbindung eines Objekttypen mit einer Detailbeschreibung | Propertytyp mit View- oder Präsentationstyp als Wertebereich |
| Kante | Verbindung von Objekttypen untereinander | Definition eines Objekttypen *Kante* mit einem Propertytypen nach dem Assoziationsmuster |
| Gerichtete Kante | Gerichtete Verbindung von Objekttypen untereinander | Definition eines Objekttypen *Kante* mit zwei Propertytypen nach dem Assoziationsmuster |

Tabelle 8: Zusammenfassung der Muster

## 7.2 Referenzmetamodell

SCHÜTTE und ROSEMANN klassifizieren die in Abschnitt 2.5.3 vorgestellten Referenzmodelle unter anderem nach dem Grad der Aussagestufe der Modellsprache in Referenzobjektmodelle und in Referenzmetamodelle (vgl. [Schü98], S. 71; [Rose96], S. 22). Ihrer Meinung nach kann der Gegenstandsbereich eines Referenzmodells sowohl ein Objektmodell als auch ein Metamodell betreffen. Gemäß den Ausführungen des Abschnitts 2.5.5 bzw. 2.5.4 befindet sich ein Referenzmetamodell auf der gleichen Stufe wie ein Metamodell und damit eine Stufe über Referenzmodellen.

Ähnlich, wie dies bereits im Abschnitt 3.1 bei der Einführung des Methodenbegriffes für Metamodell erläutert wurde, existiert bei WINTER die Auffassung, ein Metamodell umfasse ebenfalls eine Beschreibung des Vorgehens zur Erstellung und Anwendung von Referenzmodellen. Zugunsten einer einheitlichen Begriffswelt wird dem in dieser Arbeit nicht gefolgt.

### 7.2.1 Vorgehen bei der Erstellung eines Referenzmetamodells

ROSEMANN und SCHÜTTE schlagen ein zyklisches fünfphasiges Vorgehen zur Konstruktion von Referenzmodellen vor (vgl. [RoSc99], S. 26ff). An diesem wurde sich bei der Erstellung

eines Referenzmetamodells für die $E^3$-Methode orientiert. Für Referenzmetamodelle existiert ein eigenständiger Lebenszyklus, der unabhängig von den Methoden, die möglicherweise Bestandteile dessen enthalten, mit folgenden Aufgabentypen beschrieben werden kann:

1. Die Definition des zu modellierenden Problems dient in der Analysephase insbesondere der Eingrenzung des Problemraums, indem zum einen grob die Inhalte des Modells beschrieben werden und zum anderen dessen Grenzen.

2. Im ersten Teil des Entwurfs erfolgt die Erstellung eines Referenzmodellrahmens zur besseren Übersicht über das Referenzmodell.

3. Dieser wird im Folgenden inhaltlich ausgestaltet.

4. Die definierten Beziehungen innerhalb des Modells bzw. die Beziehungen zu anderen Modellen werden überprüft. ROSEMANN und SCHÜTTE bezeichnen dies als Überprüfung der Intra- und Intermodellkonsistenz.

5. Mit der Beschreibung der Anwendung schließt sich der Kreis, da mit jeder Anwendung neu entstandene Probleme definiert und im Referenzmodell berücksichtigt werden. Damit wird dieses ständig verbessert und weiterentwickelt.

Bei der Entwicklung des Referenzmetamodells wurden die Phasen drei und vier gemeinsam behandelt. Die notwendigen Beziehungen zu anderen Modellteilen werden sofort bei der Modellierung mit aufgenommen. Es wird an den jeweiligen Stellen auf bereits bestehende Referenz- bzw. andere Teilmetaschemata verwiesen. Damit reduziert sich das Vorgehensmodell auf insgesamt vier Phasen (vgl. Abbildung 58). Im Folgenden werden die einzelnen Phasen näher vorgestellt, wobei eine Beschreibung der Anwendung in Kapitel 8.3 erfolgt.

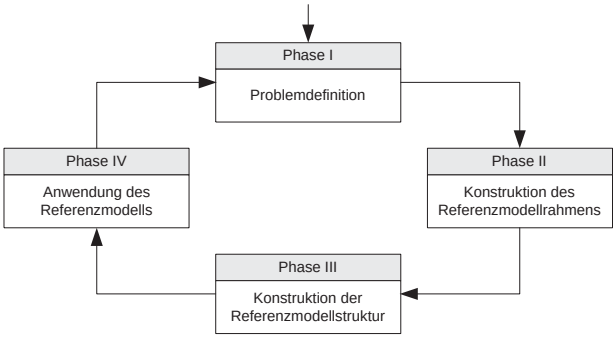



Abbildung 58: Lebenszyklus eines Referenzmetamodells

#### 7.2.1.1 Problemdefinition

In dieser Phase „... werden neben der grundsätzlichen Abgrenzung des Gegenstandsbereiches des zu erstellenden Referenzmodells auch grob die adressierten Perspektiven festgelegt“ ([RoSc99], S. 28). Das entstehende Referenzmetamodell beinhaltet die ausgewählten Metamodelle, deren Konzepte und Darstellungsmittel. Aus der Vielzahl dieser Beschreibungsmittel werden im Folgenden die wichtigsten ausgewählt und modelliert. Eine flexible Architektur ermöglicht eine spätere Erweiterung. Das Modell wird mittels $E^3$-Modell gebildet. Zusätzlichen wird eine Normierung der Benennung vorgenommen. Dies bewirkt eine schnellere Einarbeitung in abgebildete Metamodelle und unterstützt die Erweiterbarkeit. Um eine zusätzliche Vergleichbarkeit der Konstrukte zu erreichen, werden die Typisierungsmuster aus dem Abschnitt 7.1 verwendet. Mit der Normierung erfolgt zum einen eine Standardisierung von Typisierungen unterschiedlicher Metamodelle und zum anderen wird auf Grund der Vielzahl der erfassten Konzepte die Erstellung neuer Metamodelle beschleunigt.

#### 7.2.1.2 Referenzmetamodellrahmen

Der **Referenzmetamodellrahmen** bildet „... die oberste Abstraktionsebene der Referenzmodelle ...“ ([Rose96], S. 32). Er dient dazu, einen anschaulichen Zugang zum Referenzmodell zu erhalten. Durch das Prinzip der Abstraktion werden Details derart ausgeblendet, dass eine einfache Einarbeitung ermöglicht wird. Als Prinzip wird an dieser Stelle die Hierarchiebildung eingesetzt. Das Modell wird in vier Abstraktionsebenen gruppiert, wobei Elemente der niedrigeren Abstraktionsebenen Spezialisierungen von Elementen höherer Ebenen sind. Gleichzeitig dienen sie der Klassifikation der Beschreibungsmittel:

1. Die erste Abstraktionsebene beinhaltet den höchsten Oberbegriff, d. h. das zu modellierende System.

2. Daran schließt sich die zweite Abstraktionsebene der Sichten an. Bei diesen Sichten stellen die ihnen zugeordneten Beschreibungsmittel einen besonders betonten Systemaspekt in den Vordergrund (vgl. [Wint00], S. 32).

3. Die Sichten werden weiter in Paradigmen unterteilt (in Anlehnung an [Wint00], S. 34).[117] Innerhalb eines Paradigmas folgen die Beschreibungsmittel einem einheitlichen Beschreibungsmuster.

4. Die letzte Ebene stellen die Beschreibungsmittel selbst dar. In der Praxis hat sich jedoch herausgestellt, dass sich einige Beschreibungsmittel auch direkt unter eine Sicht klas-

[117] Die Abgrenzung des Paradigmas von der Sicht auf ein System erfolgt in Abschnitt 2.5.2.

sifizieren lassen (deduktiver Aspekt der Referenzmodellerstellung), sodass eine solche Klassifikation ebenfalls als zulässig erachtet wird.

Weiterhin ist die Verwendung eines mono- bzw. polyhierarchischen Abstraktionssystems zu entscheiden (vgl. [Deut80], S. 4). Bei einem monohierarchischen Abstraktionssystem muss ein Gliederungskriterium Anwendung finden, welches eine eindeutige Zuordnung von Begriffen zu genau einem Oberbegriff zulässt. Da aber Beschreibungsmittel gegenseitig in Beziehung stehen, kann für die vorliegende Arbeit ein solches ausschließendes Kriterium nicht angegeben werden. Da für polyhierarchische Abstraktionssysteme Baumdarstellungen jedoch ungeeignet, Flächendiagramme aber nur schwer lesbar sind, findet sowohl die eine als auch die andere Notation Anwendung. Die Baumdarstellung wird genutzt, um eine Gesamtübersicht zu definieren. Dabei wird von den Querverbindungen zwischen Blättern unterschiedlicher Baumäste abgesehen. Dem angeführten Problem des hohen Platzbedarfs wird begegnet, indem alle Beschreibungsmittel einheitlich in der Winkeldarstellung abgebildet werden. Alle Abhängigkeiten zwischen den Elementen werden im Flächendiagramm des Anhangs 12.1 dargestellt.

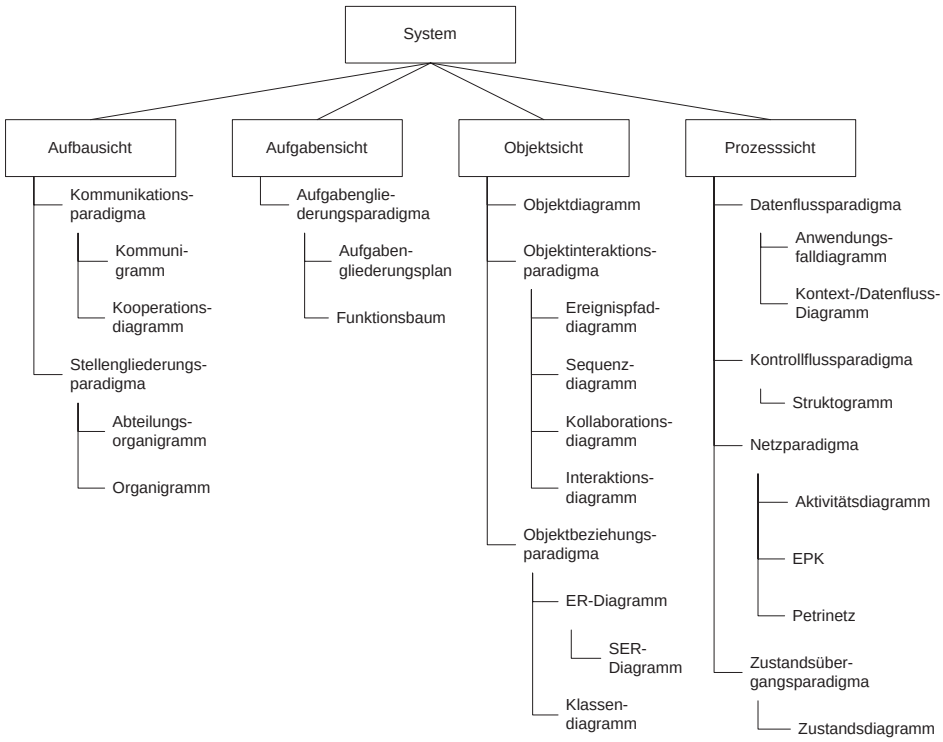



Abbildung 59: Baumdarstellung des Referenzmetamodellrahmens

Im Paket *System* werden die wesentlichen Zusammenhänge der Konzepte *Organisationseinheit*, *Aufgabe*, *Prozess* und *Objekt* hergestellt. Da diese aber die Grundkonzepte der Aufbau-, Aufgaben-, Objekt- bzw. Prozesssicht sind, d. h. diese dort definiert werden, können hier nur re-

ferenzierte Objekttypen aufgeführt werden. Dieses Paket dient damit ausschließlich dem Zweck der Erleichterung der Einarbeitung in das Referenzmetamodell. Alle modellierten Beziehungen zwischen den Elementen finden sich in der Modellierung der Sichten wieder.

Die *Aufbausicht* fasst Beschreibungsmittel zusammen, die das Ziel haben, die Struktur von Organisationen zu beschreiben sowie die Kommunikationswege der Organisationseinheiten abzubilden. Die Beschreibungsmittel des *Stellengliederungsparadigmas* bilden die Gliederung von Organisationseinheiten ab. Diesem können somit Organigramme (vgl. [Leh+91], S. 266f) sowie Abteilungsorganigramme (vgl. [Wint00], S. 47) untergeordnet werden. Die Darstellungsmittel des *Kommunikationsparadigmas* betonen die Kommunikation zwischen diesen Organisationseinheiten. Beschreibungsmittel sind das Kommunigramm (vgl. [Wint00], S. 52f) sowie das Kooperationsdiagramm (vgl. [Flo+97], S. 17).

Die *Aufgabensicht* orientiert sich ebenfalls an den Vorschlägen von WINTER (vgl. [Wint00], S. 162). Die *Aufgabe* und deren Zerlegung bilden die zentralen Elemente der Sicht. Abweichend von den Vorschlägen WINTERS wird der Objekttyp Aufgabenzerlegung im *Aufgabengliederungsparadigma* weiter verfeinert. Die Beschreibungsmittel der Aufgabensicht, Aufgabengliederungsplan und Funktionsbaum, fügen keine neuen Konzepte mehr hinzu.

Die *Objektsicht* basiert auf den zahlreichen existierenden Metamodellen zu den Methoden des objektorientierten Paradigmas. Stellvertretend wird hier Bezug auf die Aussagen von WINTER, OESTEREICH und BOOCH (vgl. [Wint00], S. 203ff; [Oest01], S. 37ff; [Booc91], S. 25ff) genommen. Eine *Instanz* wird im *Objektbeziehungsparadigma* einer *Klasse* zugeordnet. Diese kann wiederum *Attribute* enthalten. Eine solche Klasse wird zu *Objekt-* und *Beziehungsklassen* verfeinert. Da nicht alle dem Paradigma zugeordneten Beschreibungsmittel eine Generalisierung unterstützen,[118] wird zusätzlich eine generalisierbare Objektklasse spezialisiert. Die Beschreibungsmittel des Objektbeziehungsparadigmas ERM und SERM sind im Wesentlichen direkt auf dieses zurückzuführen.

WINTER schlägt für das *Objektinteraktionsparadigma* eine Zusammenfassung mit dem Kommunikationsparadigma der Aufgabensicht vor (vgl. [Wint00], S. 206). Diese ist jedoch nicht zulässig, da in letzterem u. a. die Kommunikation mit Geschäftspartnern abgebildet wird. Das Objektinteraktionsparadigma hingegen beschreibt ausschließlich Kommunikationsbeziehungen zwischen betrieblichen Objekten. Neben Kollaborations- und Interaktionsdiagrammen werden Ereignispfad- und Sequenzdiagramme der UML direkt diesem Paradigma zugeordnet.

Der *Prozess* als Kernkonzept in der *Prozesssicht* ermöglicht die Beschreibung der logischen als auch zeitlichen Reihenfolge der Aufgabenbearbeitung. Dementsprechend werden die Paradigmen zum *Daten-* und *Kontrollfluss* sowie das *Netz-* und *Zustandsübergangsparadigma* hier eingeordnet. Die Zuordnung der Beschreibungsmittel ist dem Anhang 12.1 bzw. der Abbildung

[118] Dies trifft z. B. auf den Relationship-Typen im SERM zu, da von diesem keine ausgehenden Beziehungen zulässig sind (vgl. [FeSi01], S. 155).

59 zu entnehmen.

#### 7.2.1.3 Erweiterung des Referenzmetamodells

Bei der Einordnung des Beschreibungsmittels in den Referenzmetamodellrahmen ist ein Top-Down-Vorgehen bei der Einordnung der Konzepte notwendig. Die Abbildung 60 liefert einen Überblick über die Vorgehensweise. Begonnen wird das Verfahren mit dem zentralen Konzept (z. B. *Objekt* für ein objektorientiertes Beschreibungsmittel). Dieses ist einer Sicht zuzuordnen. Ist dies nicht möglich, ist eine neue zu erzeugen und das Beschreibungsmittel als neues Paket direkt unterzuordnen. Im anderen Fall ist eine entsprechende Einordnung in untergeordnete Paradigmen oder Beschreibungsmittel zu prüfen. Für jedes Konzept, also auch für die restlichen nicht zentralen Elemente, ist im übergeordneten Paket nach möglichen Superobjekttypen zu suchen. Bei deren Identifikation in anderen Paketen muss man sich der Gefahr komplexer paketübergreifender Beziehungen bewusst sein. Diese erschweren eine spätere Wartung und dabei insbesondere Veränderung von Superobjekttypen, da in deren Paket nicht unbedingt alle Konsequenzen einer Veränderung absehbar sind.

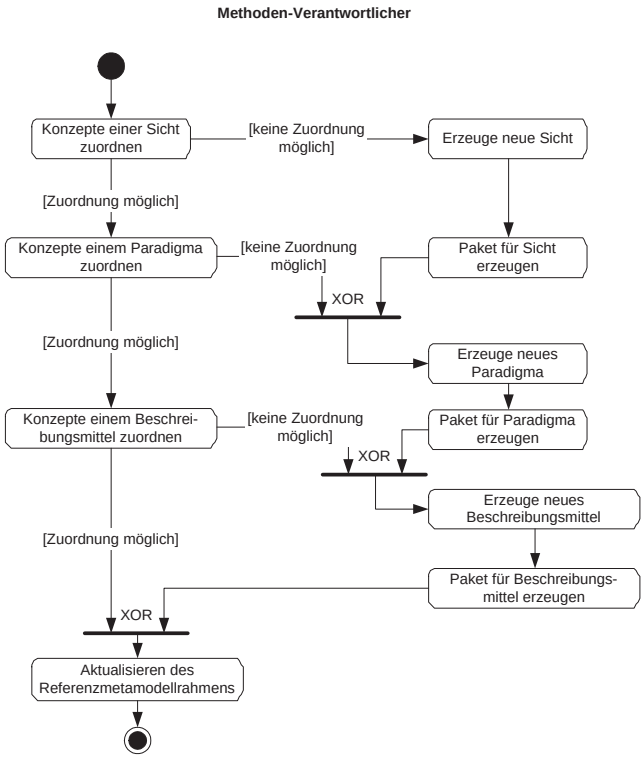



Abbildung 60: Erweiterung des Referenzmetamodells

### 7.2.2 Ergänzende Regeln

Zu den grundlegenden Regeln der Konsistenzsicherung aus Abschnitt 6.2.12 werden an dieser Stelle zusätzliche Modellierungsregeln definiert, welche die Transparenz der im $E^3$-Modell erstellten Metamodelle erhöhen. Sie beziehen sich sowohl auf die Modellierung der einzelnen Pakete als auch auf die grafische Anordnung der Elemente in den Paketen. Die Modellierungsregeln tragen damit dem Grundsatz der Klarheit Rechnung. Problematisch erscheint in diesem Zusammenhang vor allem der Umgang mit referenzierten Objekttypen. Es bedarf einer Festlegung der Informationsmenge, die im Paket des referenzierten Objekttypen abgebildet wird. Zur Formulierung der Regeln gelte beispielhaft folgender Sachverhalt: Es existieren die Pakete *Paket 1* und *Paket 2*. Der Objekttyp *OT* ist in *Paket 1* definiert und der Objekttyp *rOT* in *Paket 2*. Dann gilt:

**ER1** Sind *OT* und *rOT* Bestandteil eines Assoziationsmusters, so erfolgt die vollständige Modellierung des Assoziationsmusters in *Paket 1* und *Paket 2* (siehe Abbildung 61).

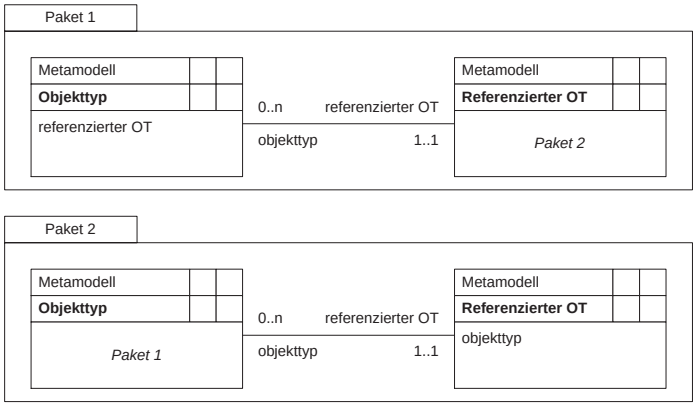



Abbildung 61: Assoziationsmuster mit Objekten aus mehreren Paketen

**ER2** Ist der *rOT* abhängiger Bestandteil eines Aggregationsmusters, so erfolgt die Modellierung des Zusammenhangs nur in *Paket 1* (siehe Abbildung 62).

In einer Präsentation eines Beschreibungsmittels wird nur der für dieses Beschreibungsmittel relevante Kontext, die zugehörigen Sichten und die jeweiligen Präsentationen modelliert. Unzulässig ist beispielsweise die Modellierung einer Sicht, die sich nicht auf den in einem Paket modellierten Kontext bezieht. Definiert ein Beschreibungsmittel keinen zusätzlichen Kontext, so werden die entsprechenden *Objekttypen* lediglich referenziert. Somit stellt die Präsentation

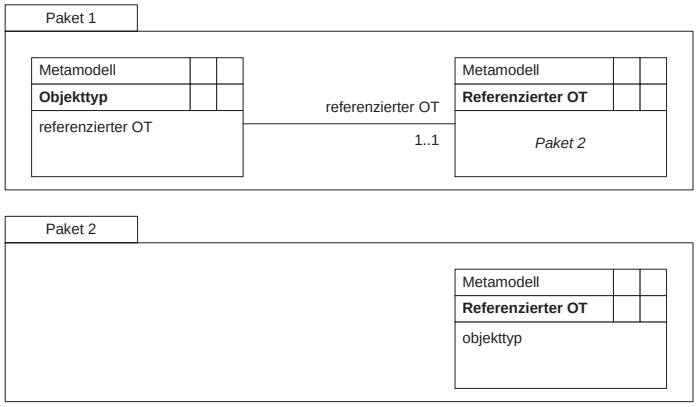



Abbildung 62: Aggregationsmuster mit Objekten aus mehreren Paketen

eines Beschreibungsmittels immer eine vollständige Typisierung dar. Damit wird die Transformation eines Paketes in die Modellebene erleichtert (vgl. Kapitel 8.3.3). Für die Modellierung der Sichten und Paradigmen wird die Abbildung von View- und Präsentationsebene nicht empfohlen und nur der entsprechende Kontext modelliert. Das Wurzelpaket System enthält die Modellierung der Zusammenhänge der vier Sichten. Es werden nur *referenzierte Objekttypen* aufgenommen. Zusätzlich ist das *Modell* enthalten. Komprimiert ergibt sich:

**ER3** Das Paket System enthält bis auf das *Modell* und *referenzierte Objekttypen* keine Elemente des $E^3$-Modells.

**ER4** In den Paketen der Sichten und Paradigmen werden nur $E^3$-Elemente der Kontextebene abgebildet.

**ER5** Auf der Ebene der Beschreibungsmittel wird eine komplette Modellebene, bestehend aus Kontext (ggf. durch *referenzierte Objekttypen*), Viewebene und Präsentationsebene, modelliert.

**ER6** Die Spezifikation der Elemente eines Pakets muss in einer Präsentation erfolgen. Der Paketname und der Name der Präsentation müssen dabei übereinstimmen.

Abschließend werden noch Empfehlungen zur grafischen Anordnung der Elemente gegeben. Generell sollen von oben nach unten in der Reihenfolge Kontext, Views und Präsentationen modelliert werden. Wird in einem Paket mehr als ein *Viewtyp* definiert, so kann die Anordnung ebenfalls in der Reihenfolge Kontext, Viewebene1, Präsentationsebene1, Viewebene2, Präsentationsebene2 usw. erfolgen. Bei der Modellierung eines Detailmusters wird das jeweilige Element der View- oder Präsentationsebene in grafischer Nähe zum Kontext modelliert. Die Modellierung von Generalisierungs- bzw. Spezialisierungsstrukturen erfolgt von oben nach unten.

Damit befinden sich im oberen Teil stets die *Superobjekttypen* und weiter unten die *Subobjekttypen*. Für Assoziationen wird keine Festlegung getroffen. Zusammengefasst ergibt sich:

**ER7** In einer Präsentation werden von oben nach unten der Kontext, die Viewebene und die Präsentationsebene dargestellt. Elemente der beiden letztgenannten Ebenen, die Teil eines Detailmusters sind, werden in der Nähe des Kontextes angeordnet. Bei der Modellierung von mehr als einem *Viewtypen* kann der entsprechende *Präsentationstyp* mit den zugeordneten *Präsentationsobjekttypen* und *Präsentationspropertytypen* direkt nach dem jeweiligen *Viewtypen* angeordnet werden.

**ER8** Den Abschluss einer Präsentation bilden ggf. untergeordnete Pakete.

**ER9** Vererbungsstrukturen werden stets von oben nach unten modelliert.

Die Einhaltung der Modellierungsregeln obliegt dem Modellierer. Da durch ihre Verletzung keine Verstöße gegen die Konsistenz des Metamodells entstehen, kann deren Einhaltung im Zuge einer nachgelagerten Qualitätssicherung geprüft werden.

# 8 Vorgehensmodell der E$^3$-Methode

Der Abschnitt 3 hat bereits die Vorteile eines ingenieurmäßigen Vorgehens bei der Entwicklung von Software erläutert. Das in diesem Abschnitt vorgestellte Vorgehensmodell muss die Spezifikation von Methoden ermöglichen, d. h. es müssen sowohl anpassbare Modellierungssprachen[119] als auch flexible Vorgehensmodelle entstehen, die in Summe den Anforderungen einer Methode aus Abschnitt 4.8 entsprechen. Um ein Vorgehen so zu beschreiben, dass es sich wirtschaftlich für eine Anwendung in Projekten eignet, muss die eigentliche Produktentwicklung so beschrieben werden, dass sie sich problemlos in folgende Grundkomponenten einer Projektorganisation einordnet (vgl. [Mayr01], S. 39ff):

- Planung: Ziel ist die Entwicklung von Vorgaben zur Methodenentwicklung in Bezug auf Aufgaben, Aufwand, Termine, Ressourcen, Kosten, Finanzen und begleitende Maßnahmen[120]. Für die Methodenentwicklung leitet sich daraus die Forderung nach Planbarkeit ab, d. h. sie muss sich klar strukturieren lassen und Arbeitspakete so definieren, dass sie auch Aufgabenträgern zugeordnet werden können.

- Durchführung: Ziel der Durchführung ist die eigentliche Methodenentwicklung, deren Vorgehen in diesem Abschnitt beschrieben wird. Inhalt ist die Abarbeitung der in der Planung vorgesehenen Arbeitsschritte. Die Aufgaben der Durchführung müssen so beschrieben werden, dass eine Reihenfolgeplanung und Priorisierung einzelner Schritte für das Projekt möglich sind. Dieser Abschnitt wird entsprechende Aufgabentypen für die Methodenentwicklung beschreiben.

- Überprüfung: Ziel ist die Erkennung von Abweichungen von der Planung während der Durchführung und die Einleitung entsprechend korrigierender Maßnahmen. Sowohl die Anforderungen an die Methode als auch deren Eigenschaften müssen demnach entsprechend messbar beschrieben werden.

Erstellt werden mit diesem Abschnitt Spezifikationen entsprechend dem Abschnitt 6.3.5, welche geeignete Aufgabentypen, Aufgabenträgertypen, Sachmitteltypen, Zieltypen und Artefakttypen enthalten. Der Schwerpunkt der Ausführungen liegt aufgrund der Zielstellungen dieser Arbeit bei den Aufgabentypen, die sich mit der Analyse, dem Entwurf und der Evaluierung von Methoden befassen. Im Rahmen des von HARMSEN ET AL. beschriebenen Situational ME werden in Projekten diese gemeinsam mit den Aufgabentypen bearbeitet, die als Bestandteil der Methode erst definiert werden. Daneben lassen sich anhand der Ausführungen in diesem Abschnitt auch „reine“ ME Projekte ableiten, die Vorteile einer systematischen Methodenentwicklung treten jedoch erst bei Anwendung des von HARMSEN ET AL. beschriebenen maximalen

[119] TOLVANEN spricht hier von einer domänenspezifischen Modellierungssprache (vgl. [Tolv01], S. 17ff).
[120] MAYR nennt hier eine Sicherungs- oder Notfallplanung (vgl. [Mayr01], S. 40).

Freiheitsgrades im Umgang mit der Methode zu Tage.[121] Auch im Zusammenhang mit der Beherrschung und Entwicklung agiler Methoden, wie sie in Abschnitt 3.5 beschrieben wurden, ist eine integrierte Betrachtung notwendig, wie sie von HARMSEN ET AL. für die Entwicklung situationsangepasster Methoden vorgeschlagen wird. Das von ihnen beschriebene Vorgehen wird in Abbildung 63 dargestellt.

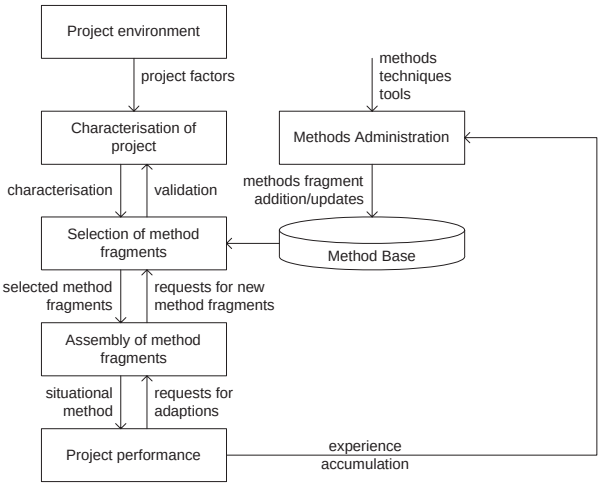



Abbildung 63: Entwicklung situativer Methoden ([Har$^+$94], S. 172)

Stellvertretend für zahlreiche Autoren schlagen GUPTA und PRAKASH einen Prozess des Method Engineering vor, der sich an den klassischen Phasen der Systementwicklung orientiert und entsprechend die Phasen Requirements Engineering, Design, Konstruktion und Implementierung enthält (vgl. [GuPr01], S. 136). Diese Einteilung erscheint nahe liegend, weil auf anderen Gebieten bewährt, trägt aber weder der Dienstleistungsfunktion des ME für das Projektmanagement noch der starken Kopplung der einzelnen Aufgabentypen des ME mit den Aufgabentypen der beschriebenen Methode Rechnung. Das in diesem Abschnitt vorgestellte Vorgehen wird diesen Punkten in besonderem Maße Beachtung schenken. Auf eine nähere Beschreibung der Tätigkeiten zur Definition bzw. Organisation des Projektes, in welchem das Vorgehen zum ME eingebettet ist, wird an dieser Stelle jedoch verzichtet. Stellvertretend wurde in Abbildung 64 die Tätigkeit *Projektorganisation* dem Ablauf vorangestellt, wohl wissend, dass damit projektphasenübergreifende Tätigkeiten nicht beschrieben werden.[122]

[121] Bei der Entwicklung von Informationssystemen unterscheiden HARMSEN ET AL. in Bezug auf das Method Engineering mehrere Freiheitsgrade (vgl. [Har$^+$94], S. 182ff). Die Spanne reicht von der Benutzung *starrer Methoden* (bei dieser Art des Vorgehens existieren für das Projekt Rahmenbedingungen, die nur die Verwendung einer vorgegebenen Methode zulassen; zusätzlich sind keine Anpassungen der Techniken oder des Vorgehens gestattet) bis zur *modularen und situationsangepassten Konstruktion von Methoden* (in diesem Fall werden die in Abschnitt 3.1 vorgestellten Methodenfragmente zu der im Projekt anzuwendenden Methode zusammengesetzt).

[122] Als weiterführende Literatur zu diesem Thema wird stellvertretend auf [Mayr01] verwiesen.

Aus Sicht des ME werden für Projekte, in denen die $E^3$-Methode angewendet wird, in Abschnitt 8.1 einige Rollen definiert. Auf diese wird bei der Zuordnung der Aufgabentypen zu Aufgabenträgern in den Folgeabschnitten Bezug genommen. In Abschnitt 8.2 werden Anforderungen an die Methode definiert und entsprechende Techniken bei der Bestimmung von deren Inhalt und Grenzen vorgestellt. In der Phase des Entwurfs wird in Abschnitt 8.3 mit Hilfe der Wiederverwendungsansätze aus dem Abschnitt 7 und ausgewählter Techniken ein Verfahren zur Spezifikation der Methoden vorgestellt. Auf die eigentliche Anwendung und projektbegleitende Evaluierung der Methode wird in Abschnitt 8.4 nur insofern Bezug genommen, als dass sie den Input für deren Evaluierung liefern. Die Nachbereitung des Projektes verändert gegebenenfalls projektübergreifend das Vorgehensmodell der Methode und passt das Referenzmetamodell an. Die Beschreibung des Prozesses erfolgt in Abschnitt 8.5.

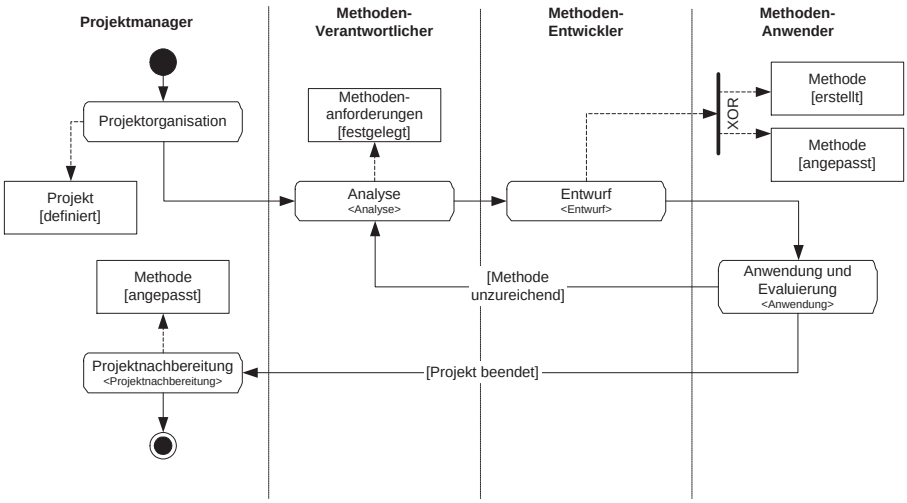



Abbildung 64: Vorgehen bei der Methodenentwicklung

Die Zuordnung der Rollen beschreibt die Hauptverantwortlichkeit für den jeweiligen Aufgabentypen in der Abbildung 64. Diese wird in der jeweiligen Detaildarstellung der Aktivitäten durch die folgenden Abschnitte verfeinert.

## 8.1 Organisationale Spezifikation

Die Verwendung von Rollen bei der Definition des Vorgehens ermöglicht die Unabhängigkeit von einer konkreten Organisationsgestaltung. Der Begriff wird hier im Sinne einer organisatorischen Rolle angewendet, wie ihn KIESER und KUBICEK definieren. Sie entspricht einer Position innerhalb der Organisation und beinhaltet „… die Ansprüche und Erwartungen der Organisation an das Verhalten der Positionsinhaber bei der Erfüllung ihrer Aufgaben …“ ([KiKu83], S. 397). Mit ihnen verknüpfte Ziele und Aufgaben bilden folglich die Basis der notwendigen

Strukturen und Aufgabentypen. In der Umsetzung dienen sie weiterhin der Vermeidung von Interessenkonflikten. Beispielsweise sollten die Entwickler einer Methode keinen Einfluss auf die Terminierung des eigentlichen Entwicklungsprozesses nehmen können.

Grundlage des hier verwendeten Rollenmodells bilden die Erfahrungen des Lehrstuhls für Systementwicklung und der semture GmbH. Infolgedessen sind die folgenden Rollen relevant:

- Projektmanager,
- Methoden-Verantwortlicher,
- Methoden-Entwickler,
- Qualitätsbeauftragter,
- Tester und
- Methoden-Anwender.

Das Hauptaugenmerk des **Projektmanagers** liegt auf der Sicherstellung des Projekterfolgs innerhalb von Budget und Zeitrahmen. Er definiert sich über das Projekt, in welches das ME eingebettet ist. Er benötigt Informationen über den gegenwärtigen Projektstand und den Status notwendiger Änderungen. Mit ihrer Hilfe ist es ihm möglich, den Entwicklungsablauf zu kontrollieren und auftretenden Engpässen bzw. Schwierigkeiten mit geeigneten Maßnahmen zu begegnen. Ihm obliegt im Speziellen die Erstellung des Konfigurationsmanagementplans, wie er in Abschnitt 9.2.1.3 beschrieben wird.

Aufgabe der **Methoden-Entwickler** ist die Realisierung des Teilprojektziels *Methodenentwicklung*. Wie bereits in Kapitel 3 dargelegt wurde, kann der Ablauf des Entwicklungsprozesses kaum vorab bestimmt werden, sondern hängt von der Psyche, Gesamtheit der Kenntnisse und dem Wertesystem der betreffenden Individuen ab. Ziel des hier beschriebenen Vorgehens muss es also auch sein, die individuellen Vorgehensweisen der einzelnen Entwickler zuzulassen bzw. geeignet zu unterstützen, gleichzeitig jedoch auch deren Bezug zu den an die Methode gestellten Anforderungen herzustellen. Für den Methoden-Entwickler sind identifizierte Mängel bzw. Fehler der Methode zu erfassen und ihre Behebung sicherzustellen.

Während durch den **Test** die Sicherstellung der Produktqualität auf Grundlage der Erfüllung funktionaler Anforderungen erfolgt,[123] dient das **Qualitätsmanagement** (QM) insbesondere der Überprüfung der Prozessqualität und der nichtfunktionalen Anforderungen. Letztere beinhalten möglicherweise die Einhaltung allgemeiner Standards, deren Anwendung mit dem Auftraggeber vereinbart wurde (siehe dazu z. B. die GoM aus Abschnitt 4.4). Im Rahmen der

[123] Diese Überprüfung kann letztlich nur durch die Anwendung der Methode erfolgen.

Prozessqualitätskontrolle sind speziell die durch die Methode definierten Aufgabentypen hinsichtlich ihrer Wirksamkeit zu evaluieren. Es ist zusätzlich sicherzustellen, dass die relevanten Verfahrensbestandteile (Methoden, Rollen, Berechtigungen und Struktur) und ihre Änderungen dokumentiert werden, um eine Basis für spätere Überprüfungen zu schaffen.

Die Umsetzung und Anpassung der Methode in den Projekten obliegt dem **Methoden-Verantwortlichen**. Seine Aufgabe ist es, die Funktionsfähigkeit der Methode im Unternehmen bzw. Projekt herzustellen und zu erhalten. Dabei können diese Aufgaben aufgrund potentieller Interessenkonflikte nicht durch das Projektmanagement (PM) übernommen werden. Dessen Ziel ist die Durchführung des Projektes im Zeit- und Budgetrahmen. Hierbei wirkt sich eine etablierte Methode jedoch eventuell negativ auf den Projekterfolg aus, da sie die Entwicklungsarbeiten beispielsweise aufgrund erhöhter Anforderungen an Dokumentation und Konsistenzsicherung beeinflussen kann. Es gehört gleichermaßen zu den Aufgaben des Methoden-Verantwortlichen, zur Verfolgung der Änderungen an einer Methode notwendige Berichte zu erstellen. Dies sollte zu großen Teilen automatisiert erfolgen können.

Der **Methoden-Anwender** ist ebenfalls im Rollenmodell zu berücksichtigen. Seine Mitarbeit konzentriert sich auf die Festlegung der Methoden-Anforderungen zu Beginn und auf die Evaluierung der Ergebnisse des ME. Im Rahmen der Anwendung von Methoden werden deren Mängel und Verbesserungsmöglichkeiten deutlich. Um ihre Behebung bzw. Umsetzung überprüfbar zu gestalten, muss der Methoden-Anwender mittels eines definierten Vorgehens Änderungsanträge in das Projekt einbringen können. Dies ermöglicht es gleichzeitig, Entscheidungen aufgrund von Änderungswünschen bzw. deren Revisionen zurückzuverfolgen.

Da das ME durch die Möglichkeit der Gestaltung von Abläufen und Strukturen innerhalb einer Methode den Charakter eines Projektmanagementwerkzeugs erhält, ist die Integration zusätzlicher Rollen zu prüfen. BENDECK ET AL. schlagen eine Unterscheidung der Funktionsbereiche in managementorientierte und technisch orientierte Aktivitäten vor (vgl. [Ben+99], S. 1). Bezüglich der managementorientierten Rollen lassen sich zusätzlich Projekt- und Messgrößenplaner sowie Qualitäts- und Produktbeauftragte unterscheiden (vgl. [Ben+99], S. 2f). Auf diese Erweiterung wird verzichtet, vielmehr werden für die Anpassung der Konzeption Möglichkeiten zur Erweiterung des Rollenmodells, z. B. durch Spezialisierung vorhandener Rollen, vorgesehen. So ist es in komplexen Projekten beispielsweise notwendig, Teilprojektleiter zu installieren, um Gruppen von Entwicklern führen zu können.

Ein weiteres bestimmendes Kriterium zur Ausgestaltung des Rollenmodells ist der weitgehende Ausschluss der bereits oben aufgeführten Interessenkonflikte. Für die Realisierung dieses Anspruchs existieren zwei Ansatzpunkte, zum einen durch eine entsprechende Hierarchisierung, und zum anderen durch die Definition der Überführungsregeln zwischen Rollen und davon abgeleiteten Stellen. Eine dritte Möglichkeit bietet die geeignete Besetzung der Stellen, also beispielsweise die entsprechende Gestaltung des Stellenprofils bzw. der Bewerberauswahl.

Diese Optionen sind jedoch stark vom konkreten Projekt und seinen Anforderungen abhängig und daher nicht Bestandteil dieser Spezifikation. Entsprechende Hinweise können z. B. *Organisationspattern* entnommen werden, wie sie u. a. durch COPLIEN ET AL. zusammengestellt wurden (vgl. [Cop$^+$00]).

Zur Vermeidung von Konflikten bezüglich kurz- und langfristiger Interessen im Projekt bzw. Unternehmen dürfen zwischen den Rollen der Projektleitung, der Methoden-Verantwortlichen und der Qualitätssicherer keine Über- bzw. Unterordnungsbeziehungen bestehen. Nur so kann erreicht werden, dass eine übergeordnete Rolle nicht die übrigen dominiert, um damit primär ihre eigenen Ziele durchzusetzen. Fehlen jedoch Hierarchiebeziehungen, so sind zusätzliche Koordinationsmaßnahmen zur Abstimmung der Tätigkeiten im Hinblick auf das Projektziel einzuführen.

Obwohl durch die gleichberechtigte Stellung der entscheidenden Rollen prinzipielle Interessenkonflikte ausgeschlossen wurden, können sie durch eine ungeeignete Stellenbildung wieder auftreten (z. B. Vereinigung der Rollen Projektmanager und Methoden-Verantwortlicher in einer Stelle). Allerdings lassen sich im Rahmen dieser Arbeit keine konkreten Stellenbildungsregeln angeben, da sie im Zusammenhang mit dem bearbeiteten Projekt zu definieren sind. Zu empfehlen ist jedoch, dass für alle Rollen zumindest eine Stelle gebildet wird, um Interessenskonflikte durch Personalunion auszuschließen. Wird für eine Rolle mehr als eine Stelle gebildet, so sind zwischen diesen geeignete Koordinations- und Führungsprinzipien zu formulieren und zu dokumentieren. Insbesondere in kleinen Projekten ist es jedoch unvermeidlich, dass mehrere Rollen durch eine Stelle wahrgenommen werden. In diesem Fall sind geeignete Kontrollmaßnahmen zu implementieren, um den Missbrauch von Berechtigungen ermitteln zu können.

Die Abbildung 65 weist zusammenfassend die im Folgenden zu verfeinernden Aufgabentypen der E$^3$-Methode den hier vorgestellten Aufgabenträgertypen zu. Auf eine Darstellung der phasenübergreifenden Tätigkeiten des Testers und des Qualitätsbeauftragten wurden aus Gründen der Übersicht verzichtet. Die Aufgabentypen wurden zusätzlich auf Grund ihres jeweiligen Schwerpunkts dem ME oder der Umsetzung der durch das ME erstellten Methode zugeordnet. So wird zum Beispiel deutlich, dass die Anwendung der Methode nur begrenzt in der Verantwortung des ME liegt.

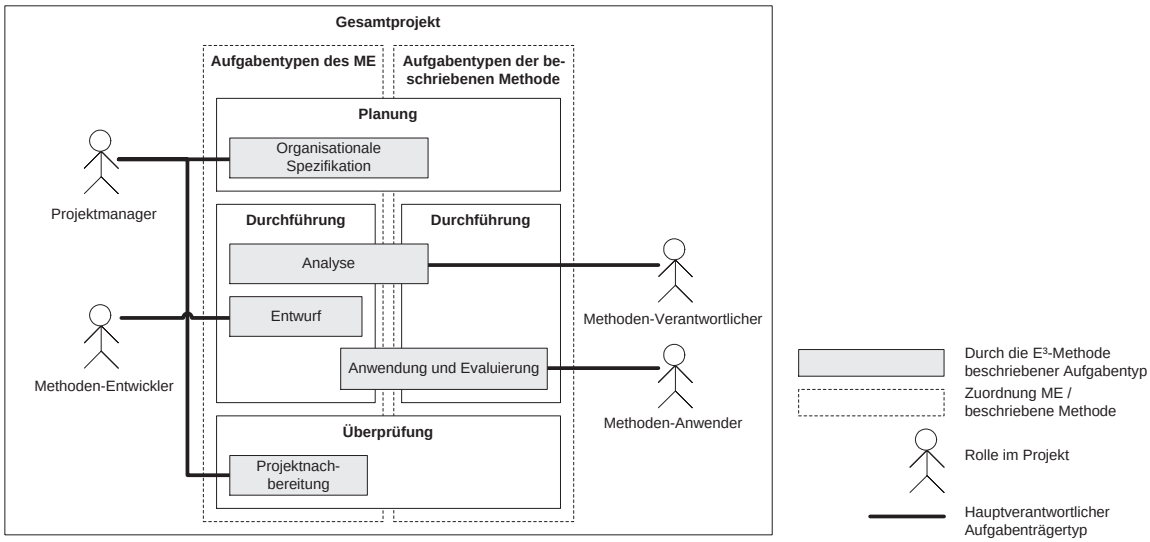



Abbildung 65: Aufgabenträgertypen der $E^3$-Methode

## 8.2 Analyse

Nach BALZERT sind Methoden, bei deren Anwendung man in der Systementwicklung von vagen Anforderungen an ein System zu definierten und präzisen Spezifikationen gelangt, nur ansatzweise vorhanden (vgl. [Balz92], S. 102). Mit der Zielstellung der Schaffung von Instrumenten zur Präzisierung von Problemstellungen entstand für die Systementwicklung das *Requirements Engineering* als Teildisziplin der Informatik. Die wichtigsten Teilaufgaben sind nach PARTSCH (vgl. [Part91], S. 27f):

- Problemanalyse: Untersuchen eines Problembereiches mit dem Ziel der Erstellung eines Kataloges von Anforderungen an das künftige System
- Anforderungsdefinition: Erklärung und Präzisierung des Anforderungskataloges als Vorgabe für eine zukünftige Konzeption
- Schrittweises Prüfen: Prüfen der Ergebnisse aus diesen Schritten anhand der Vorgaben.[124]

Die wichtigste Teilaufgabe des Requirements Engineering ist die Anforderungsdefinition. Sie dient der präzisen Beschreibung des Funktions- und Leistungsumfangs des Systems zur Problemlösung (vgl. [Part91], S. 29). Die Folgekosten bei qualitativ minderwertigen Ergebnissen dieser Phase sind so früh wie möglich zu erkennen und zu beheben (vgl. [Balz92], S. 99).

Im Abschnitt 4 wurde bereits mit dem QFD ein Prüfinstrument für die Produktentwicklung vorgestellt. Die Analysephase des hier beschriebenen Vorgehens konzentriert sich auf die Erstellung eines entsprechenden HoQ für die Methode, die im Projekt angewendet werden soll.

[124] Darin ist nicht die Überprüfung der Einhaltung der Anforderungen eingeschlossen. Diese wird den späteren Phasen des Testens oder der Wartung zugeordnet (vgl. [StHa97], S. 313).

Im Ergebnis der Tätigkeiten entstehen definierte Anforderungen an die Methode. Diese bilden die Grundlage der Aktivität „Methodenanforderungen überprüfen“ im Entwurf und werden in der Evaluierung der Methode eventuell angepasst.

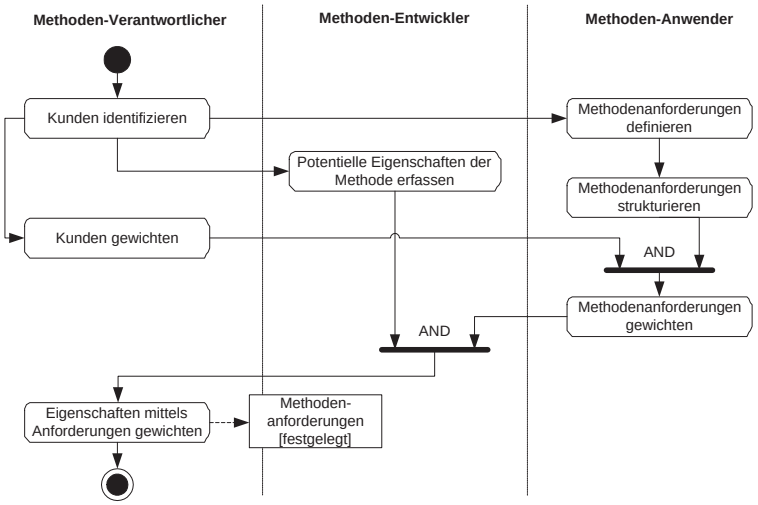



Abbildung 66: Vorgehen bei der Analyse

Die Aufgabentypen zur Strukturierung und Gewichtung der Anforderungen wurden dem Methoden-Anwender zugeordnet, da dieser den Input für die Tätigkeiten liefert. HERZWURM fordert für deren Durchführung zusätzlich die Moderation und Anleitung durch einen QFD-Experten (vgl. [Herz00], S. 234). Diese Rolle wird im ME durch den Qualitätsbeauftragten ausgefüllt. Durch ihn werden die Vorgaben erfasst und aufbereitet. Diese Form der Kooperation kann auch für die anderen Aufgabentypen von großer Bedeutung sein, wenn das Team sich selbst noch in der Lernphase beim Umgang mit dem QFD befindet.

Zur Verdeutlichung des Vorgehens dient an dieser Stelle ein Projekt der semture GmbH, in welchem für eine Schweizer Versicherungsgesellschaft die Analysephase zur Methodenspezifikation durchgeführt wurde. In diesem Projekt wurde aufgrund der geringen Anzahl an Kunden der Methode für das QFD auf deren Gruppierung verzichtet. Insgesamt waren an der Anforderungsanalyse 8 Mitarbeiter der Gesellschaft beteiligt. Die Anforderungen wurde abgefragt und anschließend in einem Workshop strukturiert.

Die Tabelle 9 zeigt beispielhaft die von einem der Teilnehmer im ersten Schritt erhobenen Aussagen zu den Anforderungen. Aus diesen wurden vom Moderator insgesamt 62 Anforderungen abstrahiert und jeweils die entsprechenden Aussagen der Teilnehmer zugeordnet. Diese Anforderungen wurden ebenfalls im Vorfeld grob in die Kategorien *Eigenschaften der Methodenprodukte*, *Eigenschaften des Prozesses*, *Eigenschaften der Gesamtmethode* und *Betriebswirtschaftliche Forderungen* untergliedert. 21 Aussagen der Kunden betrafen schon konkrete

Forderungen nach Methodenbestandteilen. Diese wurden unterteilt in *Zusätzliche Methodenbestandteile (neben Produkten und Prozessen)* und *Sprachumfang des Metamodells*. Die Teilnehmer hatten bis zu dem folgenden ersten Workshop keinerlei Kenntnis über die Vorgehensweise beim QFD. Es erfolgte nur eine kurze mündliche Einweisung für die Beantwortung der Fragen aus Tabelle 9.

| Nr. | Anforderung | Wozu | Warum | Wann | Wie |
|---|---|---|---|---|---|
| 1 | Die Methode ist in sich konsistent und widerspruchsfrei | Gewährleistet, die anvisierten Ziele auf direktem Weg zu erreichen | vermeidet Erzeugung von Ergebnissen, welche wenig (bzw. nichts) zur Zielerreichung beitragen | Methodenentwicklung, Methodenanwendung | vollständig, zwingend |
| 2 | Methode ist standardisiert (so wenig Unternehmensspezifika wie möglich) | Wiederverwendung, Erlernbarkeit | einfache Handhabung, verfügbares Know How (Markt), es gibt keine zwingenden Gründe, unternehmensspezifische Artefakte zu definieren | Methodenentwicklung | teilweise, bedingt |
| 3 | Verständlich und knapp - lieber eine einfache, lückenhafte aber anwendbare Methode als eine vollständige, welche im Schrank, im Netz oder in Köpfen von Experten verstaubt | the rubber has to hit the ground | Methode ist nicht Selbstzweck, sondern hilft, strukturiert ein bestimmtes Geschäftsziel (wiederholbar) zu erzielen | Methodenentwicklung | teilweise, zwingend |
| 4 | Stabilität | projektübergreifende, vergleichbare Ergebnisse erzielen (Architektur) | Vergleichbarkeit, Lesbarkeit zwischen Projekten erhöhen, Synergien nutzen, langfristiger Aufbau von Know How (Erreichen von Geschäftszielen) | Methodenentwicklung, Methodenanwendung | teilweise, bedingt |

Tabelle 9: Beispiel zur Erhebung von Kundenaussagen zur Methode

Es folgte ein erster Workshop, auf dem im theoretischen Teil allen Teilnehmern der Nutzen und Inhalt des QFD verdeutlicht wurde. Eine generelle Vorgehensweise für das ME existierte bereits im Unternehmen, sodass nur noch eine Einigung über die Begriffe auf diesem Gebiet (z. B. Vorgehens- und Metamodell) erfolgen musste. Im Anschluss wurden im praktischen Teil nacheinander die folgenden Schritte vollzogen:

- Definition von Teilprodukten der Methode unter Zuordnung der Aussagen der Kategorien *Zusätzliche Methodenbestandteile* und *Sprachumfang des Metamodells*
- Vorstellung der vom Moderator ermittelten Anforderungen unter Nennung beispielhafter Aussagen aus denen diese gewonnen wurden

- Konsolidierung der Aussagen durch Hinzufügen neuer, Entfernen, Zusammenlegung oder Umgruppierung ermittelter Anforderungen
- Gewichtung der Anforderungen durch paarweisen Vergleich

Die Kategorie des Methodennutzens ist an der Grenze der betriebswirtschaftlichen Forderungen. Diese hätten unter unternehmerischen Aspekten alle anderen Anforderungen dominiert und wurden deshalb nicht ausführlicher betrachtet. Die Einhaltung solcher Forderungen ist nicht Aufgabe der Qualitätssicherung eines Projektes.

Aus den anfänglich 40 Forderungen wurden im Workshop die 26 in Abbildung 67 dargestellten herausgearbeitet. Aufgrund der Tatsache, dass zur Gewichtung der Anforderungen 484 paarweise Vergleiche notwendig waren, wurden diese zwar im Workshop begonnen, dann jedoch individuell von jedem Teilnehmer fortgesetzt, zusammengeführt und das Ergebnis von allen in einer kurzen Sitzung konsolidiert. Die wichtigsten fünf Anforderungen sind für dieses Beispiel *Akzeptanz, Klarheit / Verständlichkeit und Überschaubarkeit* und *Kosten / Nutzen Verhältnis*.

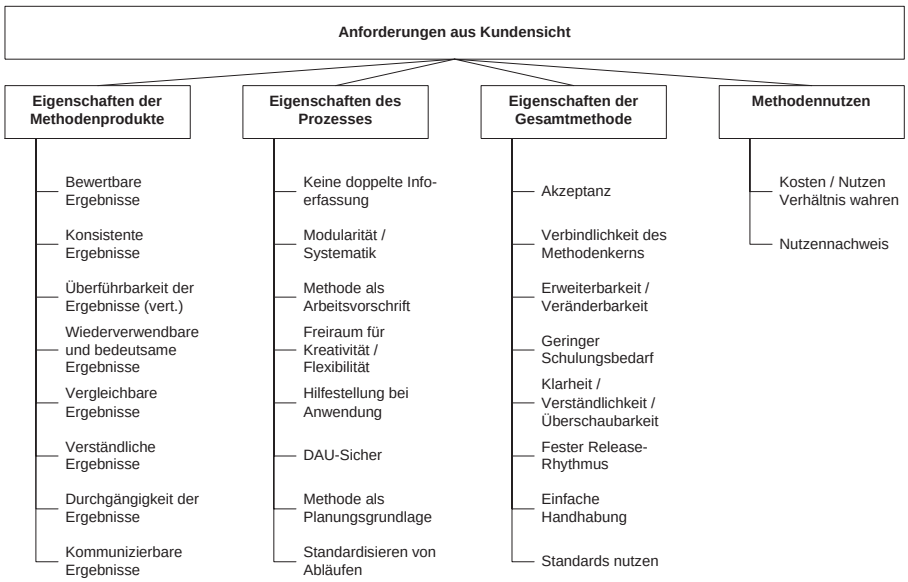



Abbildung 67: Im Workshop konsolidierte Kundenforderungen

Auf Basis der im Workshop erarbeiteten Teilprodukte der Methode erfolgt in den nächsten Schritten eine Festlegung der Eigenschaften der Methode. Die Eigenschaften der Methode werden gemeinsam mit den Methoden-Entwicklern zusammengestellt und die entsprechenden Zuordnungen zu den Anforderungen in einem Workshop erarbeitet. Zur Verdeutlichung der dabei anzuwendenden Technik dient hier bereits die Darstellung der Interdependenzmatrix im Anhang 13.5.

## 8.3 Entwurf

Der Entwurf eines Metamodells dient der Umsetzung der Anforderungen aus der Analysephase. Entsprechend der Abbildung 68 ist im ersten Schritt ein Fachbegriffsmodell und eine Ontologie zu erstellen. Das dabei anzuwendende Vorgehen wird in Abschnitt 8.3.1 beschrieben. Mit den Ergebnissen kann zur Erstellung eines geeigneten Metamodells entweder eine Auswahl aus dem Referenzmetamodell (beschrieben in Abschnitt 8.3.3) oder, wenn dies nicht möglich oder sinnvoll ist, eine direkte Umsetzung des Fachbegriffsmodells in ein Metamodell, wie es in Abschnitt 8.3.4 dargelegt wird, erfolgen.

Nach der Anpassung des Metamodells erfolgt aufgrund der Konsistenzsicherung möglicherweise eine Überarbeitung des Vorgehens, woraus eine Überarbeitung des Projektplans resultiert. Das Vorgehensmodell der Methode wird jedoch erst überarbeitet, wenn dieses am Ende des Projekts auch wirklich zum Erfolg geführt hat. Zum Abschluss der Änderungen am Metamodell ist dieses anhand der Methodenanforderungen im Rahmen des Testens zu validieren, was evtl. zu einem weiteren Durchlauf führt.

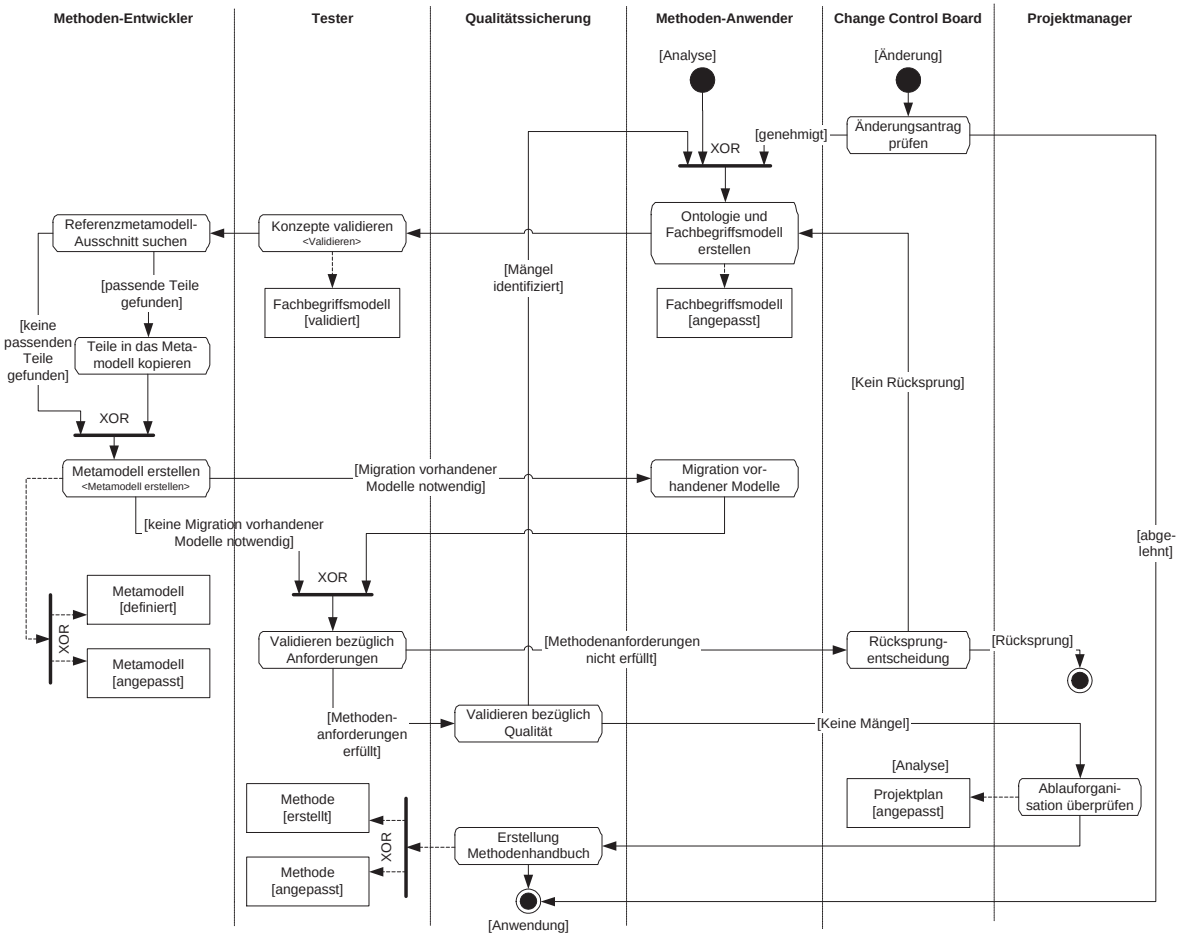



Abbildung 68: Vorgehen beim Entwurf

Während des Testens der Methode anhand der an sie mit der Analysephase gestellten Anforderungen kann ein Rücksprung in diese notwendig werden. Bestehende Eigenschaften der Methode werden in diesem Fall neu bewertet und entsprechend nur indirekt durch eine erneute Entwurfsphase beeinflusst. Ein solcher Rücksprung ist vor allem dann notwendig, wenn die (zwar intersubjektiv notwendige und durch das Change Control Board[125] bestätigte) Änderung einer Methodeneigenschaft zu einer anschließenden schlechteren Bewertung der Methode führt.

### 8.3.1 Ontologie und Fachbegriffsmodell erstellen

Als geeignete Hilfsmittel werden in diesem Abschnitt das Mind-Mapping in Abschnitt 8.3.1.1 und die Erstellung von Ontologien in Abschnitt 8.3.1.3 vorgestellt. Im Ergebnis des ersten Schrittes entsteht ein Fachwörterbuch (vgl. Abschnitt 8.3.1.4). Bei diesem Übergang vom kreativen Mind-Mapping zu einem Begriffssystem sind Konzepte, deren Eigenschaften und Beziehungen zu identifizieren. Dieses Problem wird in der Literatur vielfach dadurch verdrängt, dass man auf eine intuitive Wahrnehmbarkeit verweist. Gerade von der Objektorientierung hat man sich versprochen, dass man durch einen „offensichtlichen“ Realitätsbezug der Objekte diese nicht mehr identifizieren sondern nur noch in sein Modell integrieren muss (vgl. u. a. [Gra$^+$97], S. 2; [Oest01], S. 26ff). Trotzdem beschäftigen sich zahlreiche Veröffentlichung der letzten Jahre immer wieder mit dem Problem, wie Objekte bzw. Klassen identifiziert werden können. Die dort gefundenen Problemlösungen lassen sich teilweise auf das Identifizieren von Konzepten und deren Eigenschaften und Beziehungen bei der Typisierung übertragen. Einen vielversprechenden Ansatz stellen Heuristiken dar. Diese sind „... Tips und Tricks von Experten, wie man durch die Beobachtung der realen Welt zu Teilen des Modells kommt.“ ([Hrus98], S. 40)

In diesem Sinne werden die Heuristiken von HRUSCHKA (vgl. [Hrus98], S. 40ff) für den Entwurf der Methode teilweise übernommen und auf die Identifizierung von Konzepten, Konzepteigenschaften und Beziehungen zwischen Konzepten übertragen. In darauf folgenden Abschnitten wird die Erstellung von Ontologien und zugehörigen Fachbegriffsmodellen vorgestellt. Auf dieser Basis kann mit Hilfe des Referenzmetamodells die Arbeit abgeschlossen werden.

Um eine eindeutige Unterscheidung der Begriffe in den Meta-Ebenen vornehmen zu können, werden in diesem Abschnitt zusätzliche Begriffe vereinbart. Als **Konzept** wird ein Element eines Metamodells bezeichnet. Im Falle des E$^3$-Modells sind dies insbesondere die Objekttypen. Konzepte können Ausprägungen in der untergeordneten Ebene[126] besitzen. Diese werden an dieser Stelle als **Typen** bezeichnet. Konzepte besitzen zur näheren Erläuterung **Konzepteigenschaften**. Diesen werden in der untergeordnete Ebene **Typeigenschaften** zugeordnet. In der untersten Ebene werden den Typen **Objekte** und den Typeigenschaften entsprechend **Objektei-**

[125] Integrierendes Gremium aller Projektmitglieder, die genaue Beschreibung erfolgt in Abschnitt 9.3
[126] also der Ebene, die die Modelle zum Metamodell enthalten

**genschaften** zugeordnet. Objekteigenschaften können verkürzt als **Werte** bezeichnet werden. Die Abbildung 69 verdeutlicht diese Zusammenhänge. Zur näheren Erläuterung das folgende Beispiel:

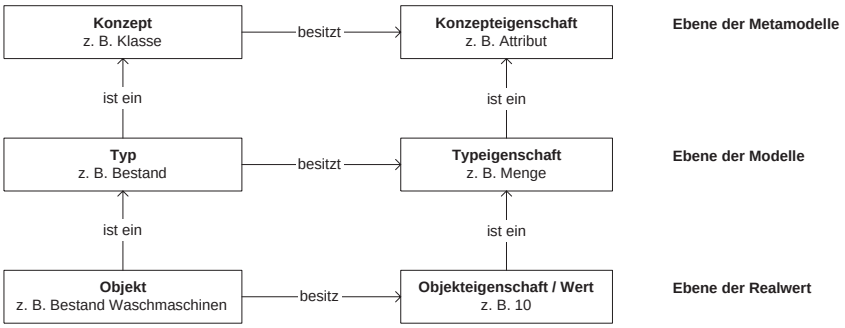



Abbildung 69: Begriffe in diesem Abschnitt

**Beispiel 1:** Für eine objektorientierte Entwurfsmethode wird als Konzept die *Klasse* definiert. Eine mögliche Konzepteigenschaft ist das *Attribut*. Gültige Typen in der Anwendung der Konzepte einer Modellierung einer Lagerhaltung wären *Bestand* und *Artikel*, Typeigenschaften entsprechend für *Bestand* z. B. die *Menge* oder für *Artikel* der *Name*. Wobei *Bestand* und *Name* jeweils als Ausprägungen von *Attribut* angesehen werden können. Eine Ebene darunter sind Objekte zum Typen *Bestand* vielleicht *Bestand an Waschmaschinen*, Werte zu *Menge* entsprechend *10*.

#### 8.3.1.1 Mind-Mapping

Zur ersten Begriffsbildung im Entwurf können Kreativtechniken eingesetzt werden. Diese basieren auf der Annahme, dass die Nutzung kreativer Potentiale unseres Denkens (z. B. Phantasie oder Assoziationsvermögen) bei der Modellierung die gleiche Berechtigung besitzen wie rationale Denkprozesse (z. B. bei der Strukturierung oder Präzisierung). Je besser es gelingt, beide Aspekte zu mobilisieren, desto stärker wird das Denkvermögen insgesamt angeregt.

Kreative Techniken basieren auf einer Unterbrechung automatisierter Denkprozesse. OESTEREICH schlägt beispielsweise vor, Blätter einfach quer zu legen, um zu vermeiden, dass man die linke obere Ecke des Blattes fixiert und anschließend nur in Zeilen denkt. Im Querformat hingegen wird der Blattmittelpunkt fixiert und das Denken in Zeilen fällt schwerer. Das ungewöhnlich liegende Blatt stimuliert die Kreativität (vgl. [Oest01], S. 117).

Das **Mind-Mapping** versieht Schlüsselworte unseres Denkens mit optischen Reizen (z. B. Farben, Symbolen oder Illustrationen wie in Abbildung 70). Die entstehenden Skizzen sollen nur Assoziationen erfassen, deren nachträgliche Perfektionierung muss vermieden werden. Nach der Erstellung des Mind-Maps erzeugt man zu diesem Zweck vielmehr eine Ontologie (siehe dazu Abschnitt 8.3.1.3).

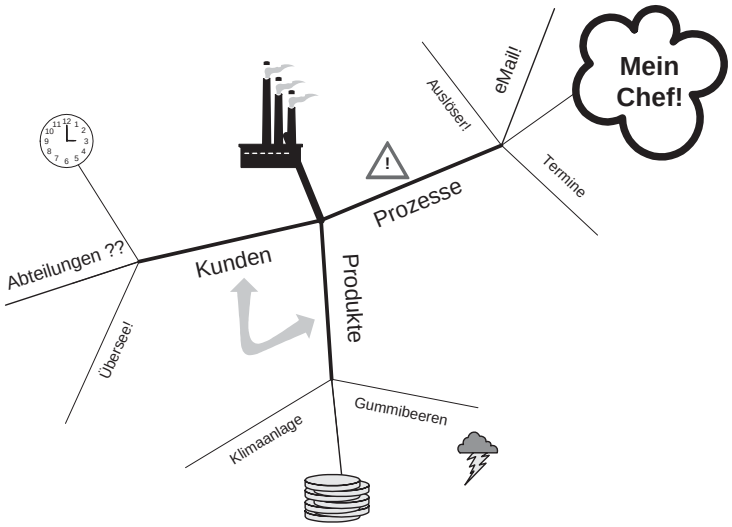



Abbildung 70: Beispiel zu einem Mind-Map

#### 8.3.1.2 Heuristiken

Die **Heuristiken** können auf verschiedene Quellen angewendet werden, so z. B. auf Ergebnisse von Interviews, Ergebnisse des Mind-Mappings oder auch auf vorhandene Dokumentation aus dem Gegenstandsbereich. Die Heuristiken implizieren keine Reihenfolge ihrer Anwendung, vielmehr werden bei den Überlegungen zur Konzeptfindung und -beschreibung bestimmte Schalter ausgelöst (WENN-Abschnitt der Heuristik), die mit einer bestimmten Wahrscheinlichkeit zur Anwendung des DANN-Abschnittes der Heuristik führen. Die einfachste Heuristik versucht, Konzepte innerhalb der Beschreibungen zu identifizieren:

**H1: Konzepte finden**

WENN man in einer Beschreibung ein echtes Substantiv hört oder liest,

DANN handelt es sich (leider nur mit einer sehr geringen Wahrscheinlichkeit) um ein Konzept.

Jedes Substantiv muss im Kontext mindestens zweimal instanziiert werden können, damit es als Konzept dienen kann.

**Beispiel 2:** Für den Begriff *Mensch* lassen sich die Ausprägungen *Mann* und *Frau* bilden. Nur wenn konkrete Männer und Frauen in späteren Modellen Gegenstand der Abbildung sein sollen, ist das Konzept *Mensch* sinnvoll.

**H2: Konzepte finden**

WENN man einem Substantiv eine sinnvolle Ausprägung und dieser wiederum eine sinnvolle Ausprägung von Bedeutung für den Gegenstandsbereich zuordnen kann,

DANN handelt es sich (mit hoher Wahrscheinlichkeit) um ein Konzept.

Umgekehrt kann man von Begriffen, denen sich nur eine Abstraktionsebene unterordnen lässt, weiter abstrahieren und so zu Konzepten gelangen, die Bestandteil eines Metamodells sind. Mit der folgenden Heuristik kann die Wahrscheinlichkeit erhöht werden, Konzepte von ihren Konzepteigenschaften, die zwangsläufig in der gleichen Abstraktionsebene liegen müssen, zu trennen:

**H3: Eigenschaften finden**

WENN dem Substantiv eine Ausprägung und dieser wiederum ein Wert zugewiesen werden kann,

DANN handelt es sich (mit sehr hoher Wahrscheinlichkeit) um eine Konzepteigenschaft und nicht um ein Konzept.

Die Trennung der Begriffe *Typ* und *Konzepteigenschaft* ist von hoher Bedeutung. Eine Entscheidung über die Wahl der richtigen Alternative kann nur durch die Problembeschreibung zur Typisierung getroffen werden.

**Beispiel 3:** Betrachtet man das Konzept *Mensch* und bildet den Typen *Mitteleuropäer*, so kann dieser Sachverhalt auch dadurch zum Ausdruck gebracht werden, dass man dem Konzept *Mensch* die Konzepteigenschaft *Wohnort* zuordnet und der Instanz von *Mensch* als *Wohnort* die Ortsangabe „Mitteleuropa“.

Ein weiteres Kriterium bei der Identifikation von Eigenschaften und insbesondere bei deren Zuordnung zu Konzepten, ist die Prüfung, ob sich den Konzepteigenschaften im gleichen Maß Ausprägungen zuordnen lassen, wie ihren Konzepten.

**Beispiel 4:** Die Zuordnung der Konzepteigenschaft *Haarfarbe* zum Konzept *Mensch* führt dann zu Problemen, wenn man als Typeigenschaften *grün* und *orange* bereits den Typen *Mann* und *Frau* zuordnen muss. Gefordert ist aber noch eine weitere untergeordnete Abstraktionsebene, die sowohl Ausprägungen von *Mann*, *Frau* und entsprechend auch *grün* und *orange* enthalten müsste. Eine Lösung des Konflikts bietet für dieses Beispiel die Einführung der Konzepteigenschaft *menschliche Eigenschaft*, die im nächsten Schritt für die Ausprägungen *Mann* und *Frau* jeweils zur Typeigenschaft *Haarfarbe* instanziiert werden kann. Umgekehrt kann dieses Problem auch jederzeit zu einem Überdenken des Konzepts *Mensch* führen.

Zur weiteren Annährung an korrekt gebildete Konzepte können folgende vier Fragen beantwortet werden, wobei jede positive Antwort die Wahrscheinlichkeit erhöht, dass es sich um ein „echtes“ Konzept handelt (in Anlehnung an [Hrus98], S. 41):

- Hat das Konzept neben der identifizierten Konzepteigenschaft mindestens noch eine weitere?
- Kann eine sinnvolle Mehrzahl vom Konzept im Kontext gebildet werden?
- Kann man das Konzept im Kontext eindeutig identifizieren (siehe dazu auch die *Methode der Typologisierung* in Abschnitt 8.3.2)?
- Wenn zwei Typen zum Konzept existieren, sind diese unterscheidbar?

Zwischen Konzepten können Beziehungen existieren. Diese können selbst wieder als Konzepte beschrieben werden. Dies sollte aber solange vermieden werden, bis zur Beziehung wiederum Beziehungen zu anderen Konzepten identifiziert werden.

**H4: Beziehungen finden**

WENN man in einem Satz ein Verb oder eine Verbform hört oder liest, die im Zusammenhang mit zwei oder mehreren Hauptwörtern erwähnt wird, die bereits als Kandidaten für Konzepte erkannt wurden,

DANN kann aus dem Verb eine Beziehung zwischen den Konzepten abgeleitet werden.

**Beispiel 5:** Zwischen unterschiedlichen Ausprägungen des Konzepts *Mensch* kann die Beziehung *Verwandtschaft* definiert werden. Erst wenn die Abbildung einer Beziehung zwischen *Verwandtschaft* und einem Konzept wie z. B. *Erbschaft* abgebildet werden soll, wird *Verwandtschaft* zum Konzept und es müssen Beziehungen zwischen *Verwandtschaft* und *Mensch* definiert werden.

Da im E$^3$-Modell zwischen Konzepten eine Generalisierungsbeziehung bestehen kann, muss auch diese identifiziert werden. Zu diesem Zweck existieren die folgenden Heuristiken:

**H5: Generalisierungen finden**

WENN man ein zusammengesetztes Substantiv hört oder liest, dessen zweiten Teil man bereits als Kandidaten für ein Konzept identifiziert hat,

DANN ist das zusammengesetzte Substantiv mit hoher Wahrscheinlichkeit eine Spezialisierung des ersten Wortteils.

**H6: Generalisierung finden**

WENN man ein Substantiv, das man bereits als Kandidaten für ein Konzept identifiziert hat, mit einem vorangestellten Adjektiv hört oder liest,

DANN ist die Kombination aus Adjektiv und Substantiv mit hoher Wahrscheinlichkeit eine Spezialisierung des Substantivs.

Auch in diesem Fall ist eine genaue Analyse der Zusammenhänge notwendig. Man muss zwischen spezialisiertem Konzept und Typ unterscheiden.

**Beispiel 6:** Bildet man zum Konzept *Mensch* das spezialisierte Konzept *Kleiner Mensch*, kann Letzteres sowohl in der selben Abstraktionsebene liegen, als auch als Bestandteil einer untergeordneten Ebene (und somit als Ausprägung von *Mensch* und entsprechend als Gegenstand der Modellierung des Objektsystems betrachtet) werden. *Kleiner Mensch* sollte nur dann als Konzept installiert werden, wenn die Kriterien der anderen Heuristiken für die Spezialisierung greifen.

#### 8.3.1.3 Bildung von Ontologien

Ein wichtiges Hilfsmittel bei der Bildung von **Ontologien** ist deren grafische Darstellung. Im Gegensatz zu den Mind-Maps aus Abschnitt 8.3.1.1 dienen diese Abbildungen der Darstellungen von Konzepten und deren Beziehungen. Während der Entwicklung der Ontologie können aber durchaus auch Konzepteigenschaften Bestandteil sein, da die Ontologien erst aus der Anwendung der Heuristiken auf Grundlage der Mind-Maps entwickelt werden. Parallel zu dieser Abbildung muss das Fachbegriffsmodell aus Abschnitt 8.3.1.4 entwickelt werden. Die Abbildung 71 stellt eine solche Ontologie dar. Die Beziehung innerhalb der Begriffe in der Ontologie können unterteilt werden in:

- Ist-Eigenschaft-von: zur Abgrenzung von Konzepten und Konzepteigenschaften.
- Ist-Ein: Darstellung der Generalisierungsbeziehung zwischen Konzepten.

Kontextabhängig sollten weitere Beziehungsarten definiert werden.

#### 8.3.1.4 Fachbegriffsmodell

Jeder in der Ontologie enthaltene Begriff wird im **Fachbegriffsmodell** aufgeführt und mindestens mit einem eindeutigen Namen, einer Beschreibung, welche die inhaltliche Bedeutung im fachlichen Kontext festlegt, und seinen Eigenschaften definiert. Ein wichtiger Bestandteil von Fachbegriffsmodellen ist der Hinweis auf Sprachdefekte (z. B. Synonyme), sofern diese sich nicht vermeiden lassen. Für die Qualitätssicherung können zu jedem Begriff dessen Quelle,

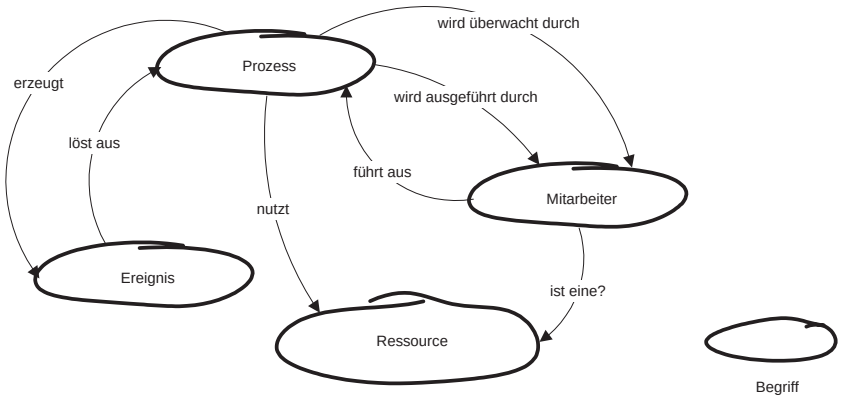



Abbildung 71: Beispiel für eine Ontologie

der Erfasser und Prüfstatus festgehalten werden. Wenn aus dem Fachbegriffsmodell im späteren Ablauf Konzepte des Beschreibungsmittels entstehen, sollten diese in einer Spalte vermerkt werden. Zur Konsolidierung der Begriffe schlägt OESTEREICH u. a. vor (vgl. [Oest01], S. 128):

- Verwendung aktiver statt passiver Formulierungen[127]
- Vermeidung von Synonymen, Homonymen und Tautologien
- Verwendung von Verben anstelle von Substantiven, die keine Fachbegriffe darstellen[128]
- Verwendung des Plural nur in begründeten Fällen[129]

Fachbegriffsmodell und Ontologie müssen ständig abgeglichen werden. Bei einer Werkzeugunterstützung lässt sich das Begriffsmodell evtl. automatisiert warten. Bei der Entwicklung des Fachbegriffsmodells werden spätestens bei der Beschreibung eines Begriffes neue Begriffe für die Ontologie definiert.

| **Begriff** | **Beschreibung** | **Eigenschaften** |
|---|---|---|
| Prozess | beschreibt einen Ablauf in unserem Unternehmen, bei dem ein Produkt erzeugt wird | Name, besitzt Ressourcen |
| Mitarbeiter | reale Person, die eine Stelle besetzt | Name, verantwortliche Prozesse |
| … | | |

Tabelle 10: Beispiel für ein minimales Fachbegriffsmodell

[127] Anstelle von „Der Prozess wird angestoßen“ ist „Ereignisse stoßen Prozesse an“ genauer.
[128] Anstelle von „Der Anstoß von Prozessen erfolgt durch Ereignisse“ ist „Ereignisse stoßen Prozesse an“ besser.
[129] Anstelle von „Ereignisse stossen Prozesse an“ ist „Ein Ereignis kann mehrere Prozesse anstoßen“ eindeutiger.

### 8.3.2 Konzepte validieren

Während beim **Verifizieren** überprüft wird, ob ein System richtig erstellt wurde (Überprüfung der internen Konsistenz), beschreibt die **Validation**, ob das richtige System erstellt wurde (Übereinstimmung mit den Anforderungen) (vgl. [Hofm00], S. 68 und [Part91], S. 47f). Ersteres wird durch die Gesamtmethode sichergestellt und Letzteres wird im Rahmen des QFD für die Methode in Abschnitt 8.3.6 beschrieben. Auf der Ebene der Konzepte aus dem Fachbegriffsmodell kann jedoch schon zu einem früheren Zeitpunkt eine sinnvolle Validierung erfolgen. Hierfür wird die *Methode der Typologisierung* verwendet. Aus der Gruppe der Begriffe im Fachbegriffsmodell werden induktiv mit Hilfe der wahrgenommenen Eigenschaften Typen gebildet. Deduktiv wird die Zielrichtung vorgegeben, diese zu einem späteren Zeitpunkt einem Typen im $E^3$-Modell zuzuordnen. Die Typen müssen nach KNOBLICH folgenden Anforderungen genügen (vgl. [Knob74], S. 141ff):

- Überschneidungsfreiheit: Die entstehenden Typen sollen eine hohe Trennschärfe aufweisen, d. h. die Überschneidungen zwischen den Typen müssen vermieden werden.
- Vollständigkeit: Die Typen sollen alle in der Realität beobachtbaren Objekte des Untersuchungsfeldes abdecken.
- Gleiche Abstraktionsebene: Die Typen sollen sich auf der gleichen Abstraktionsebene befinden.
- Allgemeingültigkeit: Die Typen sollen unabhängig von einzelfallspezifischen Gegebenheiten sein.
- Zweckbezogenheit: Die Typen sollen auf den verfolgten Zweck ausgerichtet sein.

Für die Validierung des Metamodells des Beschreibungsmittels liefert die Abbildung 72 ein geeignetes Vorgehen.

### 8.3.3 Anwendung des Referenzmetamodells

Als generelle Vorgehensweise bei der Anpassung von Referenzmodellen werden von SCHÜTTE zwei relevante Fälle unterschieden (vgl. [Schü98], S. 218). Demnach ist es denkbar, dass zunächst die relevanten Anpassungsmaßnahmen durchgeführt werden und dann eine Vervielfältigung des Referenzmodells eingeleitet wird. Dieser Prozess ist immer dann von Vorteil, wenn gleichartige Änderungen an den Kopien des Referenzmodells vorgenommen werden sollen. Die Alternative besteht darin, das Referenzmodell zunächst zu vervielfachen und anschließend zu konfigurieren. Dabei sind durch eine entsprechende Projektorganisation und Kommunikationsinfrastruktur redundante Anpassungsvorgänge zu verhindern (vgl. [Schü98], S. 219). SCHÜTTE

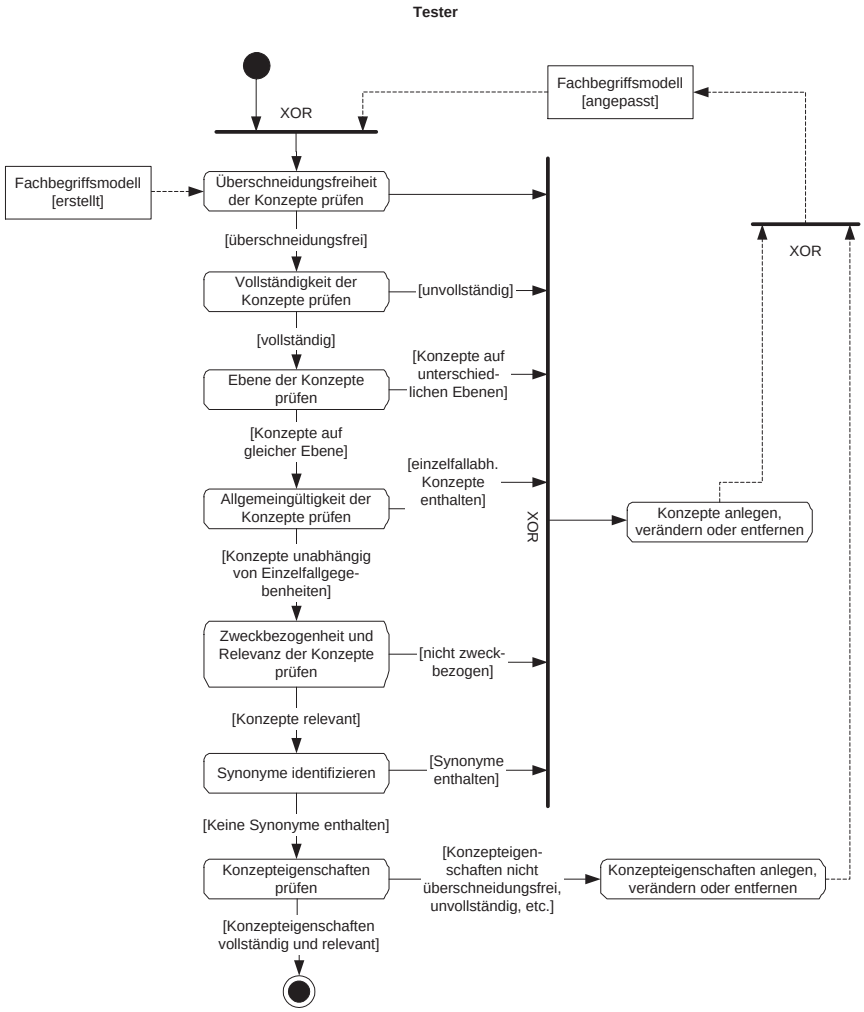



Abbildung 72: Validieren der Konzepte

gibt der zweiten Variante den Vorrang, da so die jeweiligen Anpassungen analysiert und daraus Rückschlüsse auf das Referenzmodell gezogen werden können, wie dies in Abschnitt 7.2.1 gefordert wurde.

Der relevante Teil des Referenzmetamodells wird ausgewählt, in einen separaten Arbeitsbereich kopiert (Spezifikation siehe Anhang 12.3) und anschließend durch dessen Nutzer angepasst. Im Anschluss werden die relevanten Ausschnitte des Modells markiert (Spezifikation im Anhang 12.3) und in die Typebene des E$^3$-Modells transformiert. Die Auswahl und Anpassung des Referenzmetamodells können nicht vollständig automatisiert werden, wohingegen die formale Beschreibung des Kopierens und Transformierens eine Werkzeugunterstützung ermöglicht.

Mit diesem Verfahren wird implizit eine möglicherweise vorhandene Vererbungshierarchie der Objekttypen aufgelöst, da es zu Fällen kommen kann, in denen der Objekttyp nicht mar-

kiert ist, aber einige seiner Propertytypen. Zusätzlich soll der Anpasser auch noch die Möglichkeit haben, die Elemente manuell auszuwählen, die von dem halbautomatischen Prozess noch nicht berücksichtigt wurden. Insbesondere vor der Transformation in die Typebene muss damit überprüft werden, inwieweit die Markierung konsistent gegenüber der im $E^3$-Modell geltenden Regeln ist.[130]

### 8.3.4 Metamodell erstellen

Der Aufgabentyp *Metamodell erstellen* setzt ein definiertes Fachbegriffsmodell voraus. Er ist notwendig, da bei der Suche nach passenden Teilen im Referenzmetamodell der Fall, dass keine Ergebnisse erzielt wurden, nicht unwahrscheinlich ist. Gerade spätere Durchläufe, die zu einer Anpassung vorhandener Metamodelle führen, werden entweder nur geringe oder sehr projektspezifische Anpassungen an der Methode erfordern, die den Einsatz des Referenzmetamodells ausschließen. Bei der derartigen Erstellung bzw. Anpassung von Metamodellen wird folgendes Vorgehen empfohlen:

1. Definition von Objekt- und Propertytypen: Die Heuristiken aus Abschnitt 8.3.1.2 liefern Konzepte und Konzepteigenschaften. Diese werden in Objekt- und Propertytypen überführt. Beziehungen zwischen den Konzepten werden anhand der Typisierungsmuster aus Abschnitt 7.1 abgebildet.

2. Definition von Darstellungstechniken: Für die Methode werden Präsentations-, Präsentationsobjekt- und Präsentationspropertytypen definiert.[131]

3. Definition der Viewebene: Sichten werden in erster Linie danach gebildet, welche Ausschnitte aus der Kontextebene in der jeweiligen Darstellungstechnik verwendet werden. Diese Sichten können zusätzlich als Sichten auf Sichten nach inhaltlichen Kriterien definiert werden, wie sie in Abschnitt 2.5.2 vorgestellt wurde. Dieser Schritt wird i. d. R. nicht zu einer Anpassung der Kontext- oder Präsentationsebene führen, da hier nur eine Zuordnung der Elemente beider Ebenen erfolgt. Es ist jedoch möglich, dass in der Kontextebene evtl. überflüssige (weil nicht dargestellte) Elemente entfernt oder fehlende Elemente ergänzt werden.

Der vollständige Ablauf zur Erstellung bzw. Anpassung des Metamodells ist der Abbildung 73 zu entnehmen.

[130] Es muss beispielsweise ausgeschlossen werden, dass ein Viewpropertytyp ausgewählt wird, obwohl sein zugehöriger Propertytyp nicht für die Transformation vorgesehen wurde. Dies kann durch eine entsprechende Werkzeugunterstützung sichergestellt werden.

[131] Erfahrungen des Autors in durchgeführten Projekten haben gezeigt, dass dieser Schritt regelmäßig zu einer Überarbeitung der Beziehungen in der Kontextebene führt, weil mit der grafischen Darstellung die Konzepte „greifbarer“ werden.

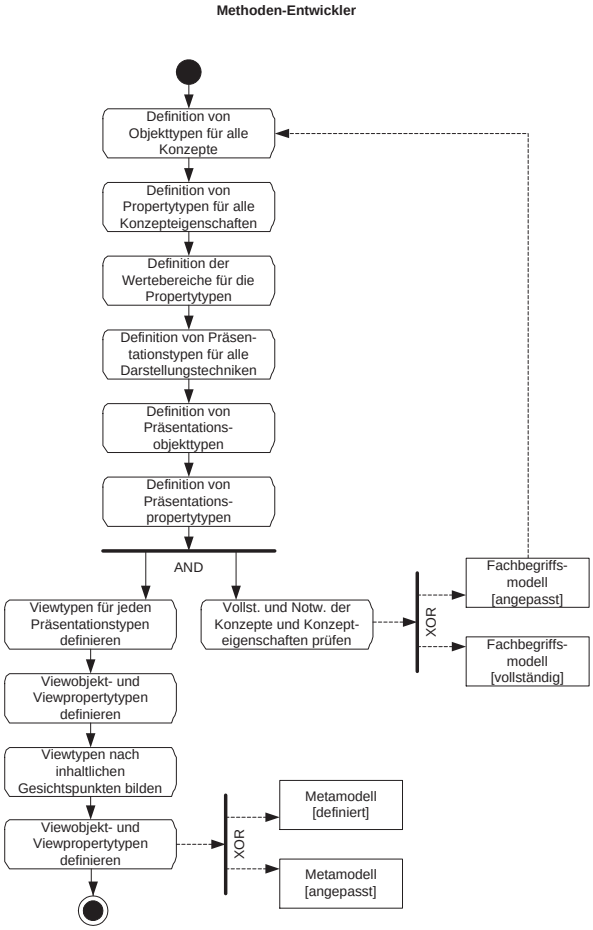



Abbildung 73: Entwurf bzw. Anpassung von Metamodellen

### 8.3.5 Transformation der Metamodell-Instanzen

Bei der Veränderung von Metamodellen entstehen möglicherweise Inkonsistenzen in den Modellen, für die die Metamodelle die Strukturen und Regeln definieren. Um die Qualität der Modelle im Sinne des Grundsatzes der Sprachadäquanz der GoM sicherzustellen und zudem den Prozess der Modellerstellung wirtschaftlich sinnvoll zu gestalten, ist es notwendig die Modelle nach Veränderungen an deren Metamodellen zu transformieren. Überlegungen zu diesem Problembereich wurden in der Vergangenheit vor allem auf dem Gebiet der Schematransformation für Datenbank Management Systemen (DBMS) angestellt.[132] Diese Ansätze sind vor allem durch ihre strenge Abhängigkeit von den verwendeten Metamodellen gekennzeichnet.

In den letzten Jahren entstanden daneben Ansätze, die eine Überführung unabhängig vom verwendeten Metamodell ermöglichen. FAHRNER beschreibt z. B. eine allgemeine Vorgehens-

[132] Diese beschäftigt sich insbesondere mit der Überführung von Datenbankschemata hierarchischer Modelle bzw. von Netzwerkmodellen in relationale Modelle und umgekehrt. Siehe dazu z. B. die Arbeiten von SAKAI (vgl. [Saka80]), IOSSIPHIDIS (vgl. [Ioss80]) oder WONG ET AL. (vgl. [WoKa80]).

weise auf Basis bestimmter modellabhängiger Basistransformationen, welche bestimmte Teilstrukturen eines Modells überführen (vgl. [Fahr97], S. 120ff). SU ET AL. schlagen eine regelbasierte Transformation von Datenmodellen vor und unterscheiden dabei *direkte* und *indirekte* Ansätze (vgl. [Su+92]). Bei einer direkten Überführung wird das Ausgangsmodell direkt in das Zielmodell übersetzt. Indirekte Ansätze überführen das Ausgangsmodell zunächst in ein Zwischenmodell mit festgelegtem und projektunabhängigem Metamodell und von dort in das Zielmodell. Bei einer indirekten Überführung werden entsprechend weniger Überführungsvorschriften benötigt. Nachteilig kann sich beim indirekten Vorgehen die Wahl der Modellierungssprache des Zwischenmodells auswirken. „The expressive power of the intermediate model has to be as strong as the sum of all data models to be handled ...“ ([Su+92], S. 7).

Im Folgenden wird ein Ansatz zur direkten Überführung beschrieben. Den Vorteilen einer indirekten Überführung steht vor allem die Auswahl eines geeigneten Metamodells entgegen. Im Umfeld der situationsangepassten und projektabhängigen Entwicklung von Methoden ist es zudem fraglich, ob ein Pool von Überführungsregeln stark voneinander abweichender Metamodelle erarbeitet werden muss, da die Änderungen am Metamodell einen eher evolutionären Charakter tragen.

#### 8.3.5.1 Anforderungen an die Transformation

Eine wesentliche Forderung an die Migration eines Modells besteht in der **Verlustfreiheit** der Transformation. Ausgangs- und Zielmodelle sollen die gleichen Informationen enthalten (vgl. [AtTo95], S. 2). Es verändert sich nur die zur Darstellung verwendete Sprache. FAHRNER bezeichnet dergestalte Überführungen als *semantisch korrekt* (vgl. [Fahr97], S. 67). In der Regel werden Anpassungen am Metamodell im Rahmen des ME zu einer Erweiterung des Sprachumfangs führen. Wenn Konzepte entfallen, sind in den meisten Fällen auch deren Ausprägungen in den Modellen überflüssig. Da diese Bedingung jedoch nicht immer erfüllt sein muss, wird sich die Spezifikation der Transformation sowohl mit der inhaltlichen Umwidmung von Konzepten und deren Eigenschaften als auch mit der alternativen Gestaltung von Modellen ohne die Verwendung der Konzepte auseinandersetzen.

Die zweite wichtige Anforderung an die Transformation ist deren **Effizienz**. Diese wird definiert als das Verhältnis zwischen dem Leistungsniveau und dem Umfang der eingesetzten Betriebsmittel (vgl. [Sti+97b], S. 216). Bei der Transformation ist vor allem die eingesetzte Zeit ein wesentlicher Einflussfaktor, nicht zuletzt dadurch, dass umfangreiche Projekte vor allem durch zahlreiche Modelle gekennzeichnet sind. Die Automatisierung bzw. Teilautomatisierung der Transformation ist in jedem Fall erstrebenswert.

#### 8.3.5.2 Ablauf der Transformation

In Anlehnung an die Arbeit von FAHRNER gliedert sich das Vorgehen in die folgenden drei Phasen (vgl. [Fahr97], S. 16), dargestellt in Abbildung 74:

1. Vorbereitung: Das Ausgangsmodell wird auf die Überführung derart vorbereitet, dass Konflikte bei der anschließenden Überführung identifiziert und möglichst bereits vermieden werden. Als Techniken werden hier die Umgestaltung des Ausgangsmodells unter Vermeidung der veränderten Konzepte und die Markierung der Elemente, deren Typen Veränderungen unterworfen wurden, eingesetzt. Jedes Element ist dahingehend zu überprüfen, ob die Zuordnung zum Typen im veränderten Metamodell korrekt ist. Die Entscheidung muss durch das Fachbegriffsmodell entsprechend unterstützt werden. Sie wird i. d. R. dadurch erleichtert, dass die Erstellung und damit verbundene Typzuordnung dieser Elemente zumeist problembehaftet war, wodurch eine Überarbeitung der Methode überhaupt angestoßen wurde. An der Vorbereitung sollte zwingend der Entwickler des Modells mitarbeiten, da inhaltliche Entscheidungen über Modellelemente nicht vom Methoden-Entwickler zu fällen sind.

2. Überführung: Die eigentliche Überführung der Modelle ist nach einer entsprechenden Vorbereitung weitgehend automatisierbar. Den Modellelementen werden entsprechend den Typen des unveränderten Metamodells Typen im veränderten Metamodell zugeordnet. Eine Behandlung der im ersten Schritt markierten Elemente ist Gegenstand der Nachbereitung.

3. Nachbereitung: Bei diesem Schritt wird in einem ersten Arbeitsgang die Konsistenz des entstandenen Zielmodells sichergestellt. Entsprechend sind vor allem Wertebereiche und Kardinalitäten im Metamodell auf ihre korrekte Anwendung im Modell hin zu überprüfen. Für die in der Vorbereitung markierten Elemente werden anschließend Ausdrucksmöglichkeiten im neuen Metamodell gesucht. Auch hier gilt, dass diese Entscheidungen nicht ohne den Entwickler des Modells getroffen werden können. Die Tätigkeiten sind i. d. R. nicht automatisierbar, da sie dem kreativen Prozess der Modellerstellung entsprechen (vgl. [Cor$^+$94], S. 203).

Mit den Elementen der Instanzenebene des E$^3$-Modells werden ebenfalls Properties überführt. Diese bilden gemäß Abschnitt 6.3.1 Beziehungen zwischen Objekten und deren Eigenschaften ab. Im Einzelnen sind folgende Punkte bei der Überführung der Werte dieser Properties zu berücksichtigen:

- Werte klassischer Datentypen, wie z. B. Zeichenketten oder Zahlen, können nicht beliebig ineinander überführt werden. Eine Veränderung der Strukturtypen muss in jedem Fall zu einer entsprechenden Strukturierung der Werte im Zielmodell führen.

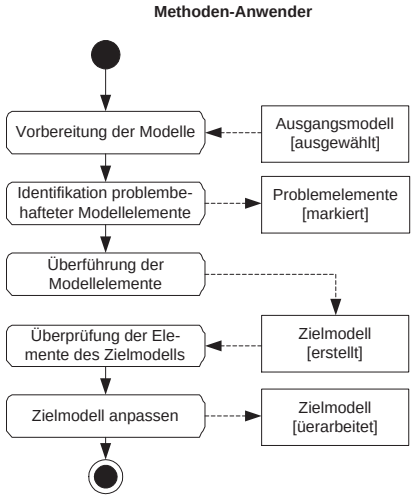



Abbildung 74: Transformation der Metamodell-Instanzen

- Werte, deren Typ ein $E^3$-Element darstellt, entfallen möglicherweise, wenn der Wertebereich verändert wurde. Unter diesem Aspekt müssen auch die grafischen Darstellungen und Beziehungen innerhalb der Viewebene überprüft werden.
- Eine Veränderung der Darstellungsformen auf Ebene der Objekte oder Properties können zu fehlerhaften Darstellungen führen und müssen gesondert überprüft werden.

Der Verlust von Informationen des Ausgangsmodells durch die Transformation lässt sich aufgrund der Unterschiede zwischen den beteiligten Metamodellen nicht immer vermeiden.[133] Der Erstellung eines syntaktisch korrekten Zielmodells muss in den ersten beiden Phasen der Transformation die höhere Bedeutung beigemessen werden. Eine Nachbereitung in Kooperation mit der Modellentwicklung lässt sich in ihrem Aufwand bei zahlreichen Änderungen nur schwer abschätzen, sollte auf jeden Fall bei einer Nutzenanalyse für die Änderungen der Methode Beachtung finden.

### 8.3.6 Überprüfung der Ablauforganisation und Methodenanforderungen

Nach der Änderung einer Methode ist in Kooperation mit der Qualitätssicherung durch den Methodenverantwortlichen die Methode dahingegen zu überprüfen, ob alle Anforderungen erfüllt sind, die an sie in diesem oder in vorherigen Durchläufen während der Analysephase (siehe Abschnitt 8.2) gestellt wurden.

[133] Praktische Erfahrungen zeigen jedoch, dass dieser dann zumeist aber auch gewollt ist und aus geänderten Anforderungen an die Methode resultiert.

Durch die Veränderungen des Metamodells können sich für das Projekt weitreichende Konsequenzen ergeben. Beispielsweise können Aufgabentypen neu entstehen oder wegfallen, Meilensteine einen neuen Inhalt erhalten oder möglicherweise müssen die Termin- und Ressourcenplanung neu überdacht werden. An dieser Stelle muss der Projektleitung klar sein, dass durch die Methode nur ein idealtypisches[134] Vorgehen beschrieben wird, wenn sie unverändert eingesetzt wird. Insofern mag der Eindruck entstehen, dass zwar nach Durchführung des Projektes möglicherweise eine gute Methode entstanden ist, aber der Projekterfolg nicht garantiert ist. Abgesehen von der Tatsache, dass Letzteres allein durch eine Methode nicht möglich ist, ist auch in Projekten, die in ein ME eingebettet sind, deren Erfolg stark vom Projektmanagement abhängig. Das Vorgehen gewährleistet „lediglich“ eine Systematik und Struktur mit den im Abschnitt 3.3 und 3.1 geschilderten Vorteilen.

### 8.3.7 Methodenhandbuch

Ähnlich wie bei der Dokumentation von Softwareprojekten ist die *Projekt-* und die *Produktdokumentation* beim ME vielschichtig motiviert. Sie ermöglicht die Kommunikation in den Entwicklungsphasen, den Einsatz und die Wartung der entstehenden Produkte und dient der Kalkulation der Kosten (vgl. [Mayr01], S. 80), um nur einige Beispiele zu nennen. Die Dokumentation der Methode ist zum einen in das Konfigurationsmanagement (siehe dazu Abschnitt 9) integriert und erfolgt zum anderen in Form eines **Methodenhandbuches**. Letzteres dient vor allem folgenden Aufgaben:

- Qualitätsmanagement: Im Rahmen des Qualitätsmanagements (QM) wird eine Beschreibung der Methode zum wesentlichen Bestandteil eines QM-Handbuchs, wie es in der ISO gefordert wird. Durch das Methodenhandbuch wird insbesondere der Punkt *dokumentierte Verfahren und Verweise darauf* (vgl. [Deut00], S. 26) abgedeckt.[135]

- Anschauliche Beschreibung der Methode: Im Rahmen der Kommunikation der Methode vermittelt das Methodenhandbauch die Zusammenhänge der Konzepte und Aufgabentypen. Es ist somit ein wichtiges Hilfsmittel bei der Schulung der Methode und gleichzeitig Referenz für den Methodenanwender beim Einsatz.

- Dokumentation der Methodenprodukte: Durch die Beschreibung der Produkte und Prozesse, die zu deren Entstehung führen, wird eine wichtige Grundlage für die Produktdokumentation gelegt.

[134] oder gemäß den Ausführungen aus Abschnitt 3.3 zumindest *wiederholbares*

[135] Die Punkte *Umfang des QM-Systems* und *Beschreibung der Abfolge und Wechselwirkungen der Prozesse des QM-Systems* können ebenfalls Bestandteil der Methode und somit Gegenstand des Methodenhandbuchs sein, womit eine Abgrenzung und getrennte Erstellung beider Bücher nicht mehr sinnvoll erscheint.

Aus jedem dieser Bereiche werden projektspezifisch Anforderungen an das Methodenhandbuch entstehen. Im praktischen Einsatz hat sich bei Projekten der semture GmbH der in Tabelle 11 gezeigte Aufbau bewährt.

| Abschnitt | Beschreibung |
|---|---|
| Einleitung | Beschreibung des Geltungsbereiches und Gegenstandes der Methode |
| Überblick | Beschreibung der durch die Methode abgedeckten Sichten sowie Aufzählung und Zuordnung der Konzepte; Angabe eines Metamodells der Konzepte |
| Beispiel | Beschreibung eines durchgehenden Beispiels, welches durch die Methode bearbeitet wird |
| Definitionen | Fachwörterbuch der Konzepte; Beschreibung der Notation und Definition der Regeln |
| Ablauf | Beschreibung der Aufgabentypen und Aufgabenträgertypen; Vorgehensmodell der Methode |

Tabelle 11: Aufbau eines Methodenhandbuchs

Mit Bezug zum im Projekt eingesetzten QM-System enthält das Methodenhandbuch meist auch eine Angabe der qualitätssichernden Maßnahmen. Zu diesem Zweck können z. B. Instrumente, wie die Grundsätze ordnungsmäßiger Modellierung, für die Methode formuliert werden.

## 8.4 Anwendung und Evaluierung

Die Anwendung der Methode im Projekt besteht zum einen aus den Aktivitäten, welche mit der Methode beschrieben werden, und zum anderen aus Tätigkeiten des Projektmanagements. Erstere unterscheiden sich nicht wesentlich von den klassischen Aufgabentypen, die in Projekten ohne Anpassung der Methode existieren. HARMSEN beschreibt hier als wesentlichen Unterschied die höhere Flexibilität im Vorgehen, die eine entsprechende Schulung der Methoden-Anwender verlangt (vgl. [Harm97], S. 39). Aus Sicht des ME ergeben sich hieraus die Aufgaben der Kommunikation der Methode und Identifikation von Mängeln an dieser. Die Überprüfung des Methodeneinsatzes zur Konsistenzsicherung der Produkte im Prozess ist ebenfalls denkbar.

Die **Kommunikation der Methode** im Projekt erfolgt durch den Methoden-Verantwortlichen und wird durch die im vorherigen Abschnitt beschriebene Dokumentation der Methode in Form eines Methodenhandbuches unterstützt. Im Vorfeld der Methodennutzung empfiehlt sich die Durchführung von Schulungsveranstaltungen, die neben der Vermittlung der Methode zur Übung auch eine Anwendung auf einem für die Methoden-Anwender möglichst fachfremden Gebiet beinhaltet. Der Methoden-Verantwortliche ist des Weiteren für den gesamten Projektablauf Ansprechpartner für Fragestellungen zur Methode. Ein wichtiger Punkt bei der Kommunikation der Methode ist die Darstellung von deren Veränderbarkeit, um die Vorteile einer anpassbaren Methode auszuschöpfen. Eine detailliert beschriebene Methode vermittelt dem Anwender den Eindruck von etwas Abgeschlossenem, auf das er wenig bis gar keinen Einfluss hat. Das dem nicht so ist, sollte nicht nur den in die Analysephase zur Methodenentwicklung integrierten Projektmitgliedern vermittelt werden.

Während der Methodenanwendung erfolgt die projektbegleitende **Evaluierung** der Methode. Aus Sicht des ME werden hier vor allem Forderungen nach neuen Eigenschaften der Methode erarbeitet. Die direkte Änderung von Anforderungen an die Methode, wie sie der Abschnitt 8.3 beschreibt, kann ebenfalls notwendig sein und wird insbesondere in den Reviews[136] des Projektes, in welches das ME eingebettet wird, erarbeitet. Entsprechend der Unterteilung einer Methode in Produkt- und Prozessbeschreibung werden Änderungen von ontologischen Eigenschaften einer Methode am häufigsten auftreten. Konflikte im Projektvorgehen müssen durch das Projektmanagement behoben werden und führen am Ende des Projektes zu einer Überarbeitung des Vorgehensmodells, wenn sich die Änderungen bewährt haben.

Alle Anforderungen können sich praktisch aus jedem Aufgabentypen der Methode ergeben. Gerade in den frühen Phasen werden Mängel an der Methode verstärkt festgestellt und bedürfen einer schnellen Behebung. Mängel, die in Bezug auf die Konzepte der Methode auftreten können, sind:

- Es sind zu wenig Konzepte vorrätig, d. h. es wird mindestens ein neues Konzept benötigt. Dies äußert sich vor allem in der Tatsache, dass im Projekt die Abbildung bestimmter Sachverhalte gar nicht oder nur eingeschränkt erfolgt. Die Konzepte werden dabei inhaltlich nicht korrekt verwendet und unter Uminterpretation ihrer Semantik eingesetzt.

- Die vorrätigen Konzepte sind nicht ausreichend präzise beschrieben. Die Aufteilung eines Konzepts in zwei oder mehrere Spezialisierungen ist notwendig, wenn der Wunsch nach präziserer Beschreibung eines Sachverhaltes existiert und die Methode bereits ein (wenn auch „unscharfes“ so doch geeignetes) Konzept zur Verfügung stellt.

- Es sind zu viele Konzepte vorrätig. Der Fall, dass die Ausdrucksmächtigkeit einer Sprache zu groß ist, wird problematisch bewertet, wenn aus Sicht des Methoden-Anwenders mehrere Konzepte für die Beschreibung eines Sachverhalts in Frage kommen, ohne dass der Methoden-Anwender eine begründete Auswahl treffen kann. Ein reiner Konzeptüberschuss ohne diese Probleme führt eventuell zu einem unnötigen Aufwand bei der Wartung und Kommunikation der Methode aus Sicht des Methoden-Verantwortlichen.

- Es sind zu viele Konzepte vorhanden, die Unterscheidung zwischen zwei oder mehreren Konzepten basiert auf keiner Notwendigkeit im Kontext des Projektes. Diese Konzepte werden entsprechend zu einem Konzept zusammengelegt (vereinigt).

Analog werden Mängel an den Eigenschaften der Konzepte wahrgenommen. Zusätzlich beinhaltet diese Gruppe jedoch auch die Defizite, die sich aus den Beziehungen der Konzepte zu-

[136] MAYR beschreibt einen Review als mehr oder weniger formale Überprüfung von Zwischenergebnissen eines Projektes in Gruppensitzungen (vgl. [Mayr01], S. 78f).

einander ergeben, da sie im $E^3$-Modell als deren Eigenschaften abgebildet werden. Beide Kategorien sind nicht immer leicht von den Mängeln und neuen Anforderungen zu trennen, die an die Repräsentation der Konzepte gebunden sind, zumal über diese der Methoden-Anwender den Zugang zu den Konzepten erlangt. Oftmals ergibt sich erst im Entwurf des angestoßenen Überarbeitungsdurchlaufs die Entscheidung, ob an der Repräsentation oder am Konzept eine Veränderung vorgenommen wird. Hierbei spielt auch der aus solchen Veränderungen resultierende Aufwand bei der Migration bereits existierender Produkte eine nicht unwesentliche Rolle.

Die Mängel und neuen Anforderungen werden entweder beim Einsatz durch den Methoden-Anwender oder erst bei der Qualitätssicherung der Produkte im Projekt identifiziert. Entsprechend ist der Methoden-Verantwortliche organisatorisch auch in die **Konsistenzsicherung der Produkte** im Prozess zu integrieren. Um das im Abschnitt 9 beschriebene Verfahren der Steuerung und Verfolgung von Änderungen zu bedienen, werden bei diesem Aufgabentypen Änderungsanträge in Form eines *Change Request Forms* (vgl. [Somm95], S. 681) verfasst und dem im Abschnitt 9.3 beschriebenen *Change Request Board* vorgelegt. Eine vollständige Strukturbeschreibung eines Änderungsformulars befindet sich in Anhang 14.3.

## 8.5 Nachbereitung

Die Nachbereitung erfasst das Projekt im Vorgehens- und Referenzmetamodell. Dazu wird untersucht, inwieweit sich die Methode im Projekt bewährt hat. Hier ist nicht allein der messbare Projekterfolg maßgeblich, sondern auch die subjektive Einschätzung aller Projektbeteiligten. Problematisch ist in dieser Phase, dass die hier entstehenden Aufwendungen den Erfolg zukünftiger Projekte beeinflussen, aber in Verantwortung des abgeschlossenen Projektes stehen. Entsprechend muss die Organisation, in der das Projekt eingebettet ist, über Rahmenvorgaben verfügen und das Projekt auch anhand der Ergebnisse dieser Phase bewerten.

Während der Nachbereitung ist anhand des Projektplans die Einhaltung des Vorgehensmodells zu überprüfen. Bei Abweichungen ist dessen Anpassung zu erwägen. Folgende Kriterien sind dafür maßgeblich die Voraussetzung:

- Das Vorgehen hat zum Erfolg geführt. Die wesentlichen Ziele des Projektes wurden erreicht. Die Rahmenbedingungen der durchführenden Organisation wurden eingehalten.
- Das Vorgehen ist auf andere Projekte übertragbar. Gerade auf der Ebene der detaillierten Ablaufbeschreibung existiert ein starker Bezug zu den Artefakttypen. Diese können in Abhängigkeit von der Allgemeingültigkeit des Metamodells zu Abläufen führen, die kaum in anderen Projekten stattfinden werden.

# 9 KMS der E$^3$-Methode

Ziel der Ausführungen in den folgenden Abschnitten ist die Entwicklung einer Konzeption zum KM in Modellen. Hierzu wird die in der Anforderungsanalyse verwendete Unterteilung nach Struktur-, Organisations- und Prozesssicht eingesetzt. Abschließend erfolgt die Integration der einzelnen Teilsichten, um eine ganzheitliche Unterstützung der Modellbildung entsprechend der Anforderungen aus Kapitel 4.5 zu ermöglichen. Das Kapitel schließt mit der Darstellung des Prozesses zum Einbringen von Change Requests in ein Modellierungsprojekt. Im Anhang 14.2 werden die folgenden natürlichsprachlichen Darstellungen durch Diagramme des E$^3$-Modells untermauert, Anhang 14.1 enthält zusätzlich ein einfaches Beispiel für die Anwendung des in das KM integrierte Versionierungskonzepts.

Die folgenden Ausführungen lassen sich allgemein auf Modellierungsansätze anwenden. Für den Bezug zur E$^3$-Methode schließt sich an jede Teilspezifikation des Konzepts eine kurze Erläuterung an. Grundlage der Ausführungen ist die bereits im Abschnitt 4.5.2 vorgestellte Definition eines Konfigurationsmanagement-Systems gemäß DIN EN ISO 10007 (vgl. [Deut96]). Unter einem KMS bzw. **Modell-KMS** wird im Folgenden sowohl das zu entwickelnde Konzept, als auch dessen Realisierung in einem konkreten System verstanden. Sollte die beabsichtigte Bedeutung nicht eindeutig aus dem Kontext hervorgehen, so wird sie durch zusätzliche Erläuterungen weiter bestimmt.

## 9.1 Begriffliche Grundlagen

### 9.1.1 Version

Eine **Version** stellt im Allgemeinen einen Zustand bzw. Aspekt eines Objekts dar (vgl. auch [West91], S. 51; [CoWe96], S. 8; [Rose92], S. 9; [Habe93], S. 209 und [Zell97], S. 9). Hierbei wird von der Struktur des versionierten Objekts abstrahiert. Diese Betrachtungsweise wird als *coarse-grained* bezeichnet (vgl. auch [West91], S. 14 und [West99], S. 68). Somit können die betrachteten Objekte **atomaren** Charakter besitzen, aber auch aus anderen Objekten zusammengesetzt sein (**komplexe** Objekte). Die Anwendung des Versionsbegriffes zeigt jedoch, dass die Versionierung von Modellen allein, also die Aufzeichnung verschiedener Entwicklungsstände, unzureichend für die hier verfolgten Zwecke ist.

Ziel muss es sein, Veränderungen am Modell zu verfolgen und kontrollieren zu können. Voraussetzung hierfür ist eine feingranulare Aufzeichnung derselben (z. B. auf Ebene der Modellelemente und deren Beziehungen untereinander). Damit folgt die Arbeit den Argumenten innerhalb der Ansätze des Software-Konfigurationsmanagements (SKM), die nach den Ausführungen von LIN und REISS (vgl. [LiRe94], S. 101ff) Änderungen auf Ebene von Softwaremodulen protokollieren sollten. In der Praxis versionieren KMS zumeist Dateien und Verzeichnisse (vgl.

z. B. CVS [Cede98]). Viele Programmiersprachen erlauben es jedoch, Module in separaten Dateien zu definieren. So fordern beispielsweise die Java Code Konventionen, dass pro Datei lediglich eine Klasse zuzüglich interner Klassen definiert wird (vgl. [Sun 99], S. 2). Werden diese in ein SKM-System eingebracht, ergibt sich zwangsläufig eine Versionierung auf Basis einzelner Module. Erfolgen dagegen Änderungsaufzeichnungen stets nur auf Ebene des Gesamtsystems bzw. des Modells, so ist ihre Aussagekraft in den meisten Fällen unzureichend.

Allerdings ist eine Protokollierung der Modifikationen allein nicht zufrieden stellend. Zusätzlich wird ihr kontrolliertes Einbringen in das jeweilige Objekt angestrebt (vgl. [Somm95], S. 676). Neben der hierzu erforderlichen Evaluierung und Planung soll u. a. auch die notwendige Dokumentation sichergestellt werden, um unerwünschte Seiteneffekte zu verhindern und Weiterentwicklungen auch in Zukunft zu ermöglichen. Außerdem sollten die eingebrachten Änderungen auch die Konsistenz des verwalteten Objekts nicht beeinträchtigen, womit zusätzlich Maßnahmen zur Konsistenzsicherung bzw. -wahrung notwendig werden. Aus den angeführten Zielen ist klar abzuleiten, dass eine reine Versionierung von Modellen lediglich einen Teilaspekt der Änderungsverwaltung darstellen kann und somit in einen größeren Gesamtzusammenhang gestellt werden muss.

### 9.1.2 Konfigurationsmanagement

Ziel eines **Konfigurationsmanagements (KM)** ist „... die gegenwärtige Konfiguration eines Produkts sowie den Stand der Erfüllung seiner physischen und funktionellen Forderungen zu dokumentieren und volle Transparenz herzustellen. Ein weiteres Ziel ist, dass jeder am Projekt Mitwirkende zu jeder Zeit des Produktionslebenslaufs die richtige und zutreffende Dokumentation verwendet.“ ([Deut96], S. 7) Ähnliche Definitionen finden sich in den Zusammenstellungen des ACME Projektes (vgl. [Appl00]), bei CONRADI, WESTFECHTEL (vgl. [CoWe96], S. 3) und GIESEL (vgl. [Gies98], S. 32). Das Konfigurationsmanagement im Bereich der Softwareentwicklung befasst sich nach Meinung dieser Autoren mit der Identifikation, Organisation und dem Controlling von Veränderungen an der Software. Ziel ist dabei die Minimierung von Fehlern und Steigerung der Produktivität des Entwicklerteams. Obwohl sich die angegebenen Definitionen in ihrem Inhalt unterscheiden, wird deutlich, dass mit dem Konzept des KM der im Abschnitt 9.1.1 geforderte Rahmen für eine umfassende Änderungsverwaltung in Modellen gefunden wurde. Dessen Fokus liegt sowohl auf der Verwaltung des Produktes und seiner Teilelemente als auch auf der Überwachung und Kontrolle seiner Veränderungen.

### 9.1.3 Weitere Begriffe zum Konfigurationsmanagement

Den zentralen Bestandteil eines KM bildet die **Konfiguration**, die funktionelle und physische Merkmale eine Produkts, wie sie in seinen technischen Dokumenten beschrieben und im Pro-

dukt verwirklicht sind, abbildet (vgl. [Deut96], S. 5). Sie setzt sich aus Versionen von im KMS verwalteten Objekten zusammen (vgl. [Rose92], S. 10 und [Deut96], S. 6). Objekte, auf die ein KM angewendet werden soll, heißen **Konfigurationseinheiten** (vgl. [Deut96], S. 6). Die verschiedenen Zustände bzw. Versionen werden im KMS als **Konfigurationselemente (KE)** verwaltet. Hierbei ist zunächst von Inhalt bzw. Struktur der Objektversionen zu abstrahieren, so dass KEs wiederum aus anderen KEs zusammengesetzt sein können (**komplexe** KEs). Infolgedessen stellen auch Versionen der Konfiguration KEs dar, wie es insb. in den neueren Ansätzen von CONRADI und WESTFECHTEL vertreten wird (vgl. auch [CoWe97] und [CoWe96], S. 14f). Somit kann eine einheitliche Behandlung aller in einem KMS verwalteten Elemente erfolgen. Versionen eines Objekts stehen über eine Menge an gemeinsamen Eigenschaften, der **Invariante**, zueinander in Beziehung (vgl. auch [CoWe96], S. 8; [West91], S. 57 und [Rose92], S. 8).[137] Die Gesamtheit aller Versionen eines Objekts bzw. einer Konfigurationseinheit wird als **Versionsfamilie** bezeichnet.

Neue Versionen werden durch Änderungen des Objektzustands, also durch Modifikation der Attributausprägungen des aktuellen KEs, erzeugt. Hierfür lassen sich drei Versionsklassen unterscheiden (vgl. [EsCa95], S. 122; [CoWe96], S. 8 und [Zell97], S. 9f). *Historische Versionen* entstehen durch die Entwicklung, Fehlerbereinigung und Wartung des Objekts im Zeitablauf und werden im Wesentlichen für die spätere Nachvollziehbarkeit von Änderungen und zugehörigen Entscheidungen aufgezeichnet. Sie stehen untereinander durch Nachfolger-Beziehungen im Zusammenhang. Im Folgenden wird für sie der Begriff der **Revision** verwandt.

Als *logische Versionen* werden **Varianten** eines Objekts bezeichnet, die durch dessen Anpassung an unterschiedliche Umgebungen (z. B. Anpassung eines Softwaremoduls an verschiedene Betriebssysteme) begründet sind. Sie entstehen ferner auch durch die Weiterentwicklung eines Produkts bei gleichzeitiger Wartung bzw. Fehlerbereinigung bereits ausgelieferter Versionen. Varianten sind nicht über eine Nachfolger-Beziehung untereinander verknüpft, sondern existieren parallel. Aus einer Version können somit mehrere Varianten folgen. Mit ihrer Erzeugung kommt es zur Aufspaltung des Entwicklungspfads in mindestens zwei Äste. Dieser Vorgang wird unter den Begriff des **Branching** subsumiert.

Eine dritte Versionsklasse bilden die *temporären Varianten*. Sie dienen der Unterstützung einer parallelen Objektbearbeitung, besitzen lediglich temporären Charakter und entsprechen hinsichtlich ihrer Semantik den logischen Versionen. Nach Abschluss der Arbeiten einzelner Entwickler werden die Teilergebnisse aus den temporären Varianten in einer gemeinsamen Version verschmolzen. Dieser Fusionsprozess wird auch als **Merge** bezeichnet. Er ist ebenfalls für Varianten im Sinne logischer Versionen erlaubt, um z. B. Fehlerbereinigungen auch in parallel weiterentwickelten Objektvarianten verfügbar zu machen. Im Gegensatz zu den temporären Varianten bleiben die Verzweigungen des Entwicklungspfads nach dem Merge jedoch erhalten.

[137] Diese wird im späteren Verlauf im Detail zu bestimmen sein.

An den Kunden ausgelieferte Objektversionen heißen **Release**.

Die Darstellung des Entwicklungspfads eines Objekts erfolgt traditionell mit Hilfe von **Versionsgraphen** ([CoWe96], S. 10f; [Zell97], S. 10f und [West91], S. 20f). Hauptkritikpunkt ihrer Verwendung bildet die mangelnde Entscheidbarkeit der Versionscharaktere ([CoWe96], S. 11f und [Zell97], S. 10f), da sie sich erst aus einem abgeschlossenen Graphen ableiten lassen. Aus diesem Grund wurden verbesserte Verfahren entwickelt, die z. B. eine dreidimensionale Darstellung des Versionsraums ermöglichen ([Reic95], S. 62ff und [CoWe96], S. 12) oder Elemente der Mengendarstellung verwenden (vgl. [Zell97], S. 91ff).

Trotz der angeführten Kritik werden in der vorliegenden Arbeit Versionsgraphen für die Darstellung von Versionsfamilien verwandt, da sie sich aufgrund ihrer intuitiven Verständlichkeit gut für die hier beabsichtigten Erläuterungszwecke eignen. Die mangelnde Unterscheidbarkeit der Versionscharaktere ist in dieser Arbeit nicht relevant, da für die Konzeption abgeschlossene und somit entscheidbare Entwicklungspfade betrachtet werden. Abbildung 75 zeigt anhand eines exemplarischen Versionsgraphen die oben erläuterten Versionsarten.

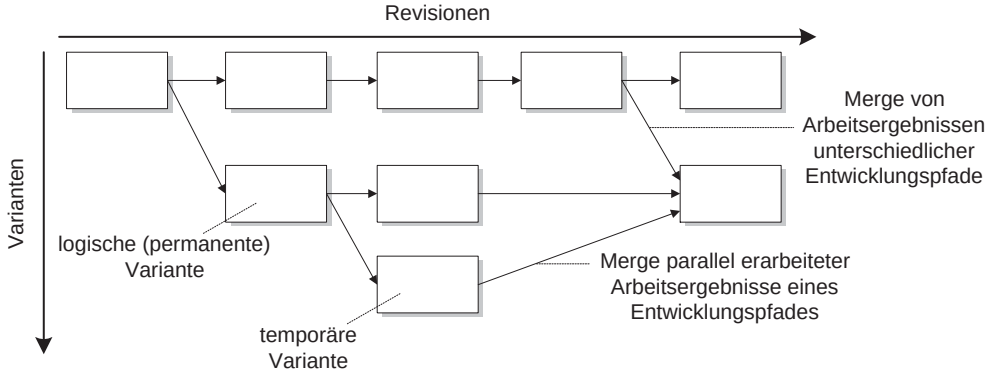



Abbildung 75: Versionsarten am Beispiel eines Versionsgraphen (i. A. a. [Zell97], S. 11)

Die Visualisierung von Konfigurationen erfolgt mit Hilfe von **Konfigurationsgraphen**, welche die enthaltenen KEs und deren Beziehungen untereinander darstellen (vgl. [West99], S. 24). Abbildung 76 zeigt zusammenfassend die eingeführten Elemente und deren Beziehungen anhand eines Beispiels. Die Darstellung der internen Struktur der Konfigurationsversionen erfolgt hierbei mit Hilfe von Konfigurationsgraphen.

## 9.2 Strukturelle Spezifikation

### 9.2.1 Konfigurationseinheitstypen

Aufgrund der in Abschnitt 9.1 eingeführten identischen Behandlung aller Konfigurationseinheiten werden diese in einem Supertypen **Version** zusammengefasst. Zur Abbildung der indivi-

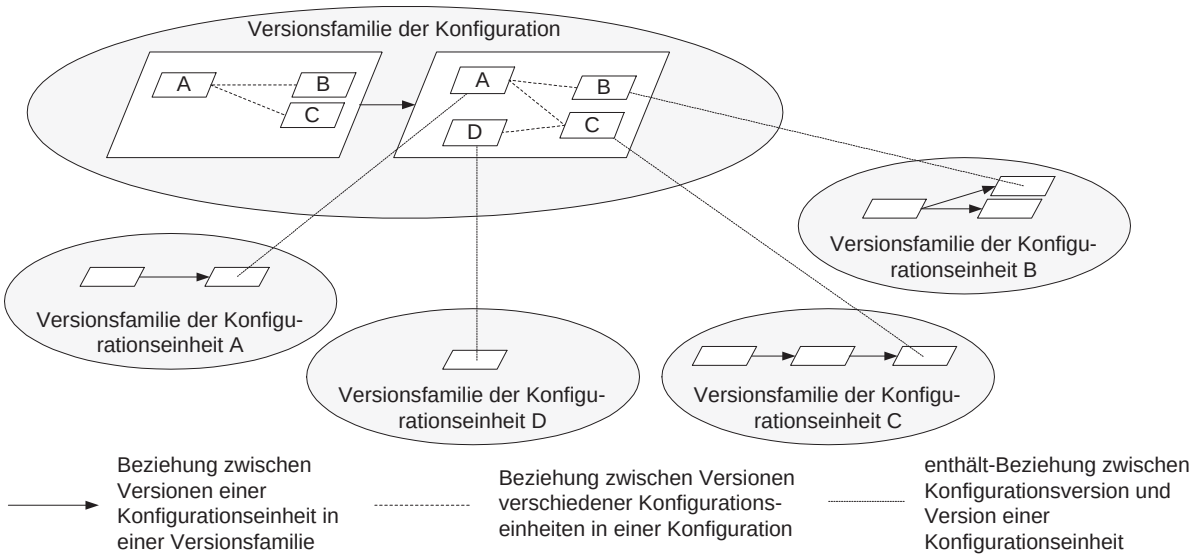



Abbildung 76: Elemente eines KMS und ihre Beziehungen

duellen fachlichen Anforderungen an die Konfigurationseinheitstypen werden daraus entsprechende Spezialisierungen abgeleitet. Die zugehörigen Attribute können aus den Ergebnissen der Anforderungsanalyse direkt bestimmt werden.

#### 9.2.1.1 Version

Bevor die strukturelle Spezifikation der Version durch Zuordnung der notwendigen Attribute vervollständigt werden kann, sind zunächst grundsätzliche Fragen zur Speicherung und Identifikation von Versionen zu klären. In der vorliegenden Arbeit wird hierzu das *zustandsorientierte* Versionierungsmodell eingesetzt (vgl. [CoWe96], S. 8), das stets die vollständige physische Speicherung einzelner Versionen vorsieht. Es bietet gegenüber dem ebenfalls möglichen *änderungsorientierten* Konzept den Vorteil einer vereinfachten Umsetzung bei gleichzeitiger Abdeckung aller relevanten Anforderungen.

Zur Identifikation von Versionen einer Konfigurationseinheit wurde in Abschnitt 9.1 bereits die Invariante eingeführt. Sie lässt sich jedoch für die unterschiedlichen Arten von Modellelementen erst nach der Auswahl des jeweiligen Metamodells festlegen. Um eine möglichst universell einsetzbare, aber dennoch umfassende Spezifikation zu ermöglichen, werden hier KEs einer Versionsfamilie über einen gemeinsamen, systemweit eindeutigen Identifikator unterschieden. Zur Differenzierung und Einordnung von KEs innerhalb einer Versionsfamilie wird eine Versionsnummer verwandt.[138] Diese muss sowohl die Rekonstruktion des Entwicklungspfads einer Version als auch die Ermittlung gebildeter Verzweigungen zulassen. Aus diesem Grund kommt

[138] Zum extensionalen Konzept siehe [Ask+99], S. 102ff.

das **Concurrent Versioning System (CVS)** zum Einsatz (vgl. [Cede98], S. 7f). Abbildung 77 zeigt seine Anwendung anhand eines Beispiels.

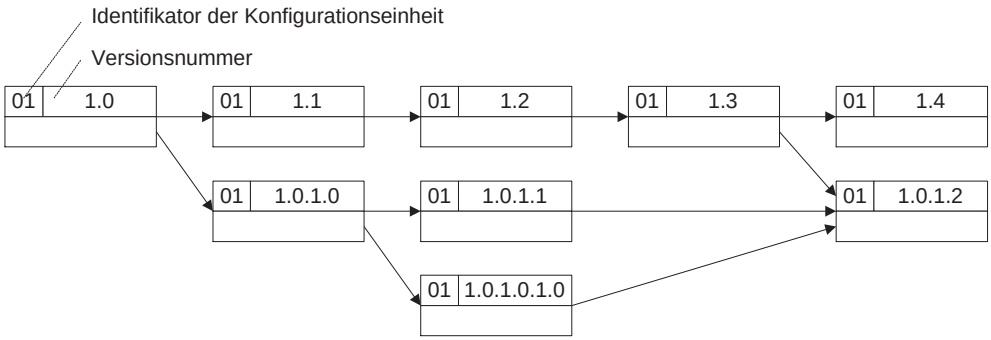



Abbildung 77: Beispiel zum Versionsnummernkonzept

Jede Version ist grundsätzlich mit einer geraden Anzahl an Nummern gekennzeichnet, die durch Punkte getrennt sind. Geradzahlige Positionen in der Versionsnummer kennzeichnen Revisionen des Objekts, während die ungeraden Rückschlüsse auf vorherige Verzweigungen zulassen. Eine initiale Version ist stets mit der Nummer 1.0 zu versehen. Mit der Erzeugung einer neuen Revision wird die am weitesten rechts stehende Position der Ursprungsversionsnummer um eins erhöht. Verzweigungen im Versionsgraphen indes erweitern die bisherige Versionsnummer um zwei Stellen. Die erste bezeichnet die Anzahl der bisherigen Verzweigungen ausgehend von der Ursprungsversion (inkl. der aktuellen Verzweigung), während die zweite mit Null belegt wird, um die initiale Version des neuen Zweiges zu kennzeichnen.

Die einzelnen Konfigurationseinheitstypen sind mit einem geeigneten Zustandsmodell zu verknüpfen.[139] Somit besitzt jedes KE einen aktuellen Zustand, der den jeweiligen Entwicklungsabschnitt der Version repräsentiert. Es ist zu beachten, dass er keine Eigenschaft des abgebildeten Elements, sondern der Version repräsentiert. Hieraus folgt, dass einer Zustandsveränderung keine Modifikation der abgebildeten Konfigurationseinheit zugrunde liegt, und deshalb auch nicht die Ableitung einer neuen Revision erforderlich wird.

Das hier verwendete Zustandsmodell aus Abbildung 78 lehnt sich an die Arbeiten von WESTFECHTEL (vgl. [West91], S. 12) an. Allerdings ist das Löschen eines KE aus dem KMS explizit ausgeschlossen. Wird es innerhalb einer Modellversion nicht länger benötigt, so kann es aus dieser entfernt werden. Das KE selbst bleibt jedoch im System weiter verfügbar. In Erweiterung des Ansatzes von WESTFECHTEL wird das Zustandsmodell explizit auf Anforderungen aus der Organisation des Entwicklungsprozesses abgestimmt. So muss eine zur endgültigen Freigabe bestimmte Version (Zustand *released*) sowohl erfolgreich getestet worden sein (Zustand *tested*)

[139] Dieses dient im Wesentlichen der Einschränkung der möglichen Operationen auf die Konfigurationselemente zu einem bestimmten Zeitpunkt, wie sie auch im Abschnitt 9.5 beschrieben wird.

als auch eine Überprüfung der Maßnahmen zur Qualitätssicherung durchlaufen haben (Zustand *quality_assured*). Komplexe KEs können einen bestimmten Zustand nur erreichen, wenn alle enthaltenen Versionen ebenfalls mindestens denselben erreicht haben. Hiermit wird beispielsweise verhindert, dass eine Modellversion bereits zur Auslieferung freigegeben wird, obwohl sich einzelne Modellkomponenten noch in der Entwicklung befinden.

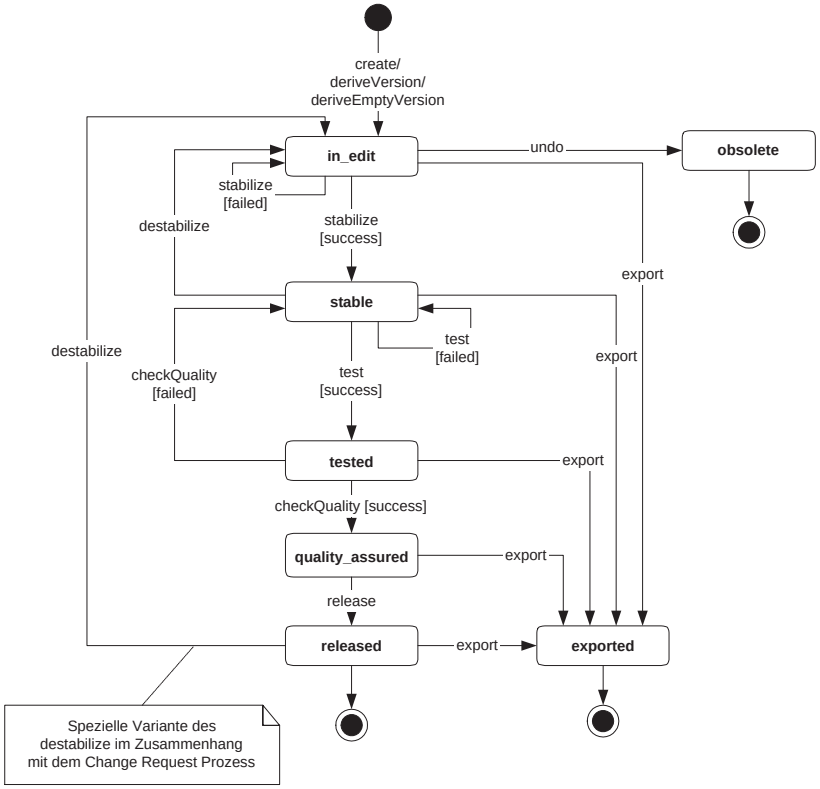



Abbildung 78: Zustandsübergänge der Version

Die Zustände *exported* bzw. *obsolete* kennzeichnen keine Entwicklungsabschnitte, sondern basieren auf Anforderungen des KM bzw. der Modellierung. Eine Version im Zustand *exported* wurde im Zuge einer ***Export*** Operation (vgl. Abschnitt 9.4.1.1) in den aktuellen Workspace überführt und steht für weitere Versionsbildungen nicht zur Verfügung. Der Zustand *obsolete* kennzeichnet hingegen durch Anwendung der ***Undo*** Operation (vgl. Abschnitt 9.4.2) nicht länger benötigte Versionen. Sie sind zwar noch existent, werden durch das KMS aber als nicht vorhanden betrachtet.

Durch die Zustandsdefinitionen wurden jedoch lediglich die Voraussetzungen zur Einhaltung der organisatorischen Regeln und Fixpunkte geschaffen. Zu ihrer Realisierung sind sie in Abschnitt 9.5 geeignet mit den zugehörigen Operationen zu verknüpfen.

Die Dokumentation der KEs erfolgt durch das *Versionsdokument* (vgl. Abschnitt 9.2.2). Inwiefern zu jeder Version ein solches existieren muss, kann nicht allgemeingültig definiert

werden. So erfordert die veränderte Darstellungsgröße eines Modellelements u. U. eine neue Version desselben. Ihre explizite Begründung in einem separaten Dokument erscheint indes nicht erforderlich. Andererseits enthält die Versionsbeschreibung aber auch relevante Informationen über den Entwicklungsprozess der Version. Als Kompromiss wird daher angenommen, dass stets ein Versionsdokument zu führen ist. Allerdings wird seine Vollständigkeit (Angabe von Erstellungsgründen und Versionsmerkmalen) nur für spezielle, z. B. an den Kunden auszuliefernde, KEs gefordert.

Eine *gültige* Version zeichnet sich durch die Vergabe von Identifikator und Versionsnummer aus, während eine *vollständige* zusätzlich durch ein ebenfalls *vollständiges* Versionsdokument beschrieben ist.

#### 9.2.1.2 Modellelement

Zur Abbildung von Modellelementen sind ebenfalls entsprechende Konfigurationseinheitstypen erforderlich, deren vollständige Spezifikation jedoch erst bei bekanntem Metamodell und den darin beschriebenen Typen möglich ist. Um diese für das KMS bedeutende Elementklasse in der allgemeinen Spezifikation angemessen berücksichtigen zu können, wird eine allgemeine Klasse **Modellelement** als spezielle Ausprägung zur **Version** eingeführt.

Gemäß den Darlegungen in Abschnitt 4.5.3 werden die Entitäten des Originals über deren Eigenschaften wahrgenommen. Sie werden in Form von Attributausprägungen des korrespondierenden Modellelements in das Modell eingebracht. Die Festlegung der möglichen Attribute eines Modellelements erfolgt durch seinen Typen im zugehörigen Metamodell. Allen Modellelementen gemeinsam ist die prinzipielle Möglichkeit, mit anderen Elementen in Beziehung zu stehen (z. B. Typbeziehung, Beziehung zu anderen Modellelementen). Zur Verwaltung dieser Verknüpfungen stehen zwei Strategien zur Verfügung, die ihre Ablage entweder in entsprechenden Attributen der Typen der beteiligten KEs oder in einer separaten Konfigurationseinheit vorsehen.

Ihre Verwaltung als Attribut erscheint auf den ersten Blick intuitiver, da sie als Merkmal der Elemente aufgefasst werden kann. Zur Erstellung einer Beziehung wären die entsprechenden Eigenschaften der beteiligten Modellelemente zu modifizieren und eine neue Version derselben zu bilden. Allerdings erzeugt dieses Vorgehen eine endlose Wiederholung, deren Entstehung im Folgenden kurz dargelegt wird.

Es sollen die Versionen zweier Modellelemente $x$ und $y$ ($_xME^i$ und $_yME^j$) miteinander verknüpft werden. Für die Ablage einer Verknüpfung müssen im Modellelement Identifikator und Versionsnummer des jeweiligen Partnerelements erfasst werden. Durch die Eintragung von $_xME^i$ als Partnerelement von $y$ entsteht eine neue Version $_yME^{j+1}$. Wird sie als Partnerelement von $x$ eingetragen, wird $_xME^{i+1}$ erzeugt. Hiermit steht $_xME^{i+1}$ zwar mit der aktuellen Version von $y$ in Beziehung, $_yME^{j+1}$ jedoch lediglich mit $_xME^i$. Wird durch eine geeigne-

te Operation $y$ entsprechend aktualisiert, so kommt es zur Bildung von ${}_yME^{j+2}$. Im Ergebnis steht sie nun mit der aktuellen Version von $x$ in Beziehung, jene allerdings mit ${}_yME^{j+1}$.

Zur Lösung dieses Dilemmas könnten spezielle Attribute eingeführt werden, deren Modifikation keine neue Elementversion nach sich zieht. Eine weitere Möglichkeit besteht darin, die Erstellung von Verknüpfungen als atomare Operation zu betrachten. Sie überführt beide Modellelemente jeweils in eine neue Version und setzt die betreffenden Attribute. Beide Ansätze erfordern jedoch Ausnahmen vom hier verwendeten Konzept zur Versionsbildung. Deshalb wird ein separater Konfigurationseinheitstyp eingeführt, dessen Instanzen Beziehungen zwischen Versionen von Modellelementen verwalten (vgl. auch Abschnitt 9.2.1.5). Ein weiterer Vorteil des Ansatzes liegt darin, dass hiermit prinzipiell alle auftretenden Beziehungen zwischen KEs zentral verwaltet werden können.

Aus Sicht des KMS beinhalten Konfigurationen alle Modellelemente der Modell- und Metamodell-Ebene. Beziehungen zwischen einem Element und seinem Typen werden in der gleichen Konfiguration beschrieben, wie die Beziehungen zu den Elementen der gleichen Ebene, z. B. die eines Objekts zu seinen Eigenschaften. Für die Gültigkeit eines Modells im Modell-KMS ist zu fordern, dass jedes Modellelement genau eine Beziehung zu einem Modellelement des zugehörigen Metamodells besitzt.

#### Bezug zur E$^3$-Methode

Elemente der Typ-, Instanzen- und Vorgangsebene werden im KMS als Modellelement abgebildet. Sie bilden somit die zu versionierenden Modellelemente bzw. Konfigurationseinheiten. Veränderungen ihrer Attributwerte (z. B. des Namens eines Objekts) führen zur Bildung von Versionen derselben, die durch entsprechende KEs im Modell-KMS repräsentiert werden. Deren Attribute ergeben sich aus der Definition der korrespondierenden E$^3$-Elemente.

### 9.2.1.3 Konfigurationsmanagementplan

Der **Konfigurationsmanagementplan (KMP)** legt für ein spezifisches Produkt oder Projekt die KM-Organisation und die anzuwendenden Verfahren fest (vgl. [Deut96], S. 6). Er enthält die Beschreibung der anzuwendenden Verfahren, die zu ihrer Durchführung berechtigten Projektmitarbeiter und Rollen sowie den Durchführungszeitpunkt (vgl. [Deut96], S. 17). Gleichzeitig wird aber auch seine anpassende Funktion deutlich, da mit den Festlegungen sowohl eine Auswahl aus den verfügbaren Verfahren erfolgt, als auch das KMS insgesamt auf die Projektanforderungen abgestimmt wird. Somit bilden die Informationen des KMP die Grundlage zum Audit des Modell-KMS. Diesbezügliche Erfordernisse werden allerdings maßgeblich vom Kunden beeinflusst, so dass der KMP neben dem Problembezug ebenfalls einen wichtigen Vertragsbestandteil darstellt. Obwohl seine Aufgabe somit primär dokumentierenden Charakter trägt,

wird er trotzdem als Konfigurationseinheit behandelt. Damit können zu jedem Modellstand die aktuell zutreffenden KM-Bestimmungen verwaltet werden.

Seine Attributierung lehnt sich an die Empfehlungen der DIN Norm (vgl. [Deut96], S. 19f) an, die neben den KM-Funktionsbereichen[140] auch Abschnitte zu allgemeinen Grundsätzen, Verfahren und Projektinformationen vorsehen. Wie einige Beispiele aus der Praxis zeigen (vgl. [US D00] und [Info97]), hat sich eine derartige Strukturierung bewährt. Da auch die Norm lediglich Empfehlungen zu den einzelnen Abschnitten des KMP enthält, wird hier ebenfalls nur eine Attributierung auf Kapitelebene vorgeschlagen, deren Ausgestaltung im Detail den Gegebenheiten im aktuellen Projekt überlassen bleibt.

#### Bezug zur E$^3$-Methode

Der Konfigurationsmanagementplan wird innerhalb des Aufgabentyps *Projektorganisation* (siehe Abschnitt 8) erstellt.

### 9.2.1.4 Konfiguration

Gemäß den Ausführungen des Abschnitts 9.1 werden die zu einem Modell zugehörigen Versionen der Modellelemente, des Problembezugs, der Namenskonvention, des KMP und der Konfigurationsstruktur zentral durch Versionen der Konfiguration verwaltet. Sie können damit begrifflich Modellversionen gleichgesetzt werden. Zu ihrer Integration in die KM-Konzeption wird ein entsprechender Konfigurationseinheitstyp **Konfiguration** eingeführt.

Aufgrund der Heterogenität der verwalteten Konfigurationseinheiten kann die Invariante der Konfiguration nicht eindeutig bestimmt werden. Die Frage, ob die Ausführung einer Operation zur Erstellung eines neuen Modells oder lediglich zur Bildung einer neuen Version führt, lässt sich nur bei gleichzeitiger Betrachtung der Operation und des Typs der betroffenen Konfigurationseinheit beantworten. So sollte das Einfügen eines Modellelements zur Bildung einer Modellversion führen. Wird dagegen der Problembezug aktualisiert, muss ein neues Modell erzeugt werden, da sich die Ziele und Anforderungen der Anwender verändert haben. Damit wird im Prinzip ein neues Projekt begründet. Die zusätzlichen Restriktionen sind im Rahmen der Integration der Teilsichten (vgl. Abschnitt 9.5) entsprechend zu berücksichtigen.

#### Bezug zur E$^3$-Methode

Entsprechend den Ausführungen aus Kapitel 6.1 werden Modelle und Metamodelle in einer Konfiguration abgelegt, um die Beziehungen der Elemente aus Typ- und Instanzenebene in einer Konfigurationsstruktur abzubilden. Nur so lässt sich beispielsweise die Änderung einer grafischen Darstellung durch den Wechsel eines Präsentationsobjekttypen im Sinne des KMS

[140] Konfigurationsidentifizierung, -überwachung, -buchführung und -auditierung

abbilden. Der aktuelle Stand eines Modells wird durch eine Konfigurationsversion repräsentiert, die u. a. auf Versionen von E$^3$-Elementen verweist. Im Anhang 14.1 wird ein einfaches Beispiel der Versionierung von Modellelementen, Konfigurationen und Konfigurationsstrukturen vorgestellt.

#### 9.2.1.5 Konfigurationsstruktur

Die Verwaltung von Modellen erfordert nicht nur die Erfassung der Modellelemente, sondern auch der Beziehungen zwischen diesen (siehe dazu Abschnitt 4.5.3). Zur Verwaltung Letzterer wird der Konfigurationseinheitstyp **Konfigurationsstruktur** in das KMS aufgenommen. Da die Konfiguration alle zugehörigen KEs direkt verwaltet, ist eine separate Behandlung der Beziehungen zwischen ihnen und der jeweiligen Modellversion nicht erforderlich. Somit reduzieren sich die in der Konfigurationsstruktur erfassten Verknüpfungen auf:

- *Typbeziehungen* – Beziehungen zwischen Basistypen/Modellelementversionen aus der Metamodellversion und Modellelementversionen im bearbeiteten Modell und
- *Modellinterne Beziehungen* – Beziehungen zwischen Modellelementversionen innerhalb des bearbeiteten Modells.

Aufgrund der unterschiedlichen Verknüpfungssemantiken gestaltet sich die Festlegung der Invariante ähnlich komplex wie im Falle der Konfiguration. Werden modellinterne Beziehungen entfernt, so handelt es sich lediglich um eine Weiterentwicklung des Modells, die durch die Erstellung einer neuen Version der Konfigurationsstruktur repräsentiert wird. Wird dagegen eine Typbeziehung verändert, so ändert sich gleichzeitig auch der Charakter des betroffenen Modellelements. Dies ist zwingend durch die Erstellung einer neuen Konfigurationseinheit für das Modellelement abzubilden.

#### Bezug zur E$^3$-Methode

Die Typ- und Abhängigkeitsbeziehungen aller Elemente der Typ-, Instanzen- und Vorgangsebene werden zentral in der Konfigurationsstruktur verwaltet. Mit ihnen sind Konsistenzregeln im E$^3$-Modell verbunden, deren Verletzung eine weitere Modellierung nicht zulässt.[141]

### 9.2.2 Konfigurationsdokumenttypen

Aufgrund der Anforderungen aus der hier zugrunde gelegten DIN 10007 lassen sich zwei Dokumenttypen identifizieren. Zum einen dient das **Versionsdokument** zur Beschreibung einzelner

[141] So muss beispielsweise ein Viewobjekttyp stets mit einem vorgeordneten Objekttyp oder Viewobjekttyp, sowie einem Viewtyp verknüpft sein (siehe Abschnitt 6.2.6).

KEs, während **Change Request Dokumente (CRD)** die notwendigen Informationen zu einem eingebrachten Änderungsantrag verwalten. Allen Dokumenten ist jedoch gemeinsam, dass sie nicht versioniert werden, da sie der Beschreibung eines Elements im KMS dienen.

Aufgabe des Versionsdokuments ist es insbesondere, die Erstellungsgründe und Eigenschaften eines KE zu beschreiben. Ferner werden auch die für seine Entwicklung verantwortlichen Mitarbeiter erfasst, um später eine genaue Analyse der Entwicklung zu ermöglichen. Im Gegensatz dazu dokumentiert das CRD neben den Gründen und Zielen einer beantragten Änderung auch ihren Genehmigungsstatus sowie von der Realisierung betroffene Konfigurationseinheiten. Außerdem werden ebenfalls die Ergebnisse der Änderungsrealisierung, die verursachten Aufwände und die jeweils beteiligten Mitarbeiter beschrieben.

## 9.3 Organisationale Spezifikation

Mit der Analyse der zu unterstützenden Benutzergruppen in Abschnitt 4.5.1 wurden Rollen identifiziert, die sowohl lang- als auch kurzfristige Interessen des Unternehmens verfolgen. Aufgabe der Projektleitung ist es, das Projekt sowohl im Budget- als auch Zeitrahmen fertig zustellen. Sie verfolgt damit primär kurzfristige Ziele. Im Gegensatz dazu dienen die Tätigkeiten der Tester und Qualitätssicherung im Wesentlichen der Wahrung langfristiger Ziele, wie z. B. der Weiterentwickelbarkeit des Produkts.

Um die konträren Interessen im Projekt aufeinander abzustimmen, können keine Hierarchiebeziehungen zwischen den Gruppen eingeführt werden. Dies hätte stets eine potentielle Benachteiligung der jeweils untergeordneten Rollen zur Folge. Stattdessen wird im Modell-KMS ein integrierendes Gremium, das **Change Control Board (CCB)** eingeführt. Es umfasst Mitglieder aller relevanten Benutzergruppen. Seine Aufgabe besteht neben dem Ausgleich der Interessen insb. auch in der Kontrolle von einzubringenden Änderungen am Produkt. Es dient somit gleichzeitig auch als zentrale Genehmigungsinstanz für Change Requests.

### Bezug zur E$^3$-Methode

Durch die Implementierung des KM werden keine grundsätzlichen Änderungen der organisationalen Spezifikation notwendig, da keine neuen Rollen eingeführt bzw. bereits bestehende verändert werden. Allerdings bieten sich verschiedene Optionen zu einer detaillierteren Einschränkung der Benutzerprivilegien an. So kann das Benutzermanagement sie zunächst auf Typ- bzw. Instanzenebene begrenzen. Hiermit kann z. B. gewährleistet werden, dass die Fachabteilung lediglich Zugriff auf Modelle zur Ablage fachlichen Wissens hat. Eine Bearbeitung der verwendeten Metamodelle ist jedoch nur durch entsprechend ausgebildete Mitarbeiter möglich. Infolgedessen wird abgesichert, dass Veränderungen am Metamodell, neben der Kontrolle durch den Change Request Prozess, lediglich durch einen überschaubaren Personenkreis durchgeführt

werden können.

Neben der Beschränkung des Zugangs zu einer E$^3$-Ebene erscheinen außerdem zusätzliche Restriktionen innerhalb derselben sinnvoll. Hierzu sollte eine Dominanz der vor- über die nachgeordneten Elemente im E$^3$-Modell zugrunde gelegt werden. Dies bedeutet, dass der Zugriff auf E$^3$-Elemente der Kontextebene gleichzeitig auch entsprechende Rechte auf die nachgeordneten Elemente (View, Präsentation) impliziert. Dagegen erlaubt das Privileg zur Veränderung von Elementen der Präsentationsebene keine Arbeiten auf View- bzw. Kontextebene. In der nächsten Granularitätsstufe ist die Präzisierung der Limitierungen auf Ausprägungen eines E$^3$-Elements und ihre Weitergabe gemäß der Abhängigkeitsbeziehungen denkbar. So würde die Freigabe eines Objekts gleichzeitig auch die Erlaubnis zur Manipulation ihm direkt bzw. indirekt nachgeordneter Properties, Viewobjekte, Viewproperties, Präsentationsobjekte und Präsentationsproperties beinhalten. Eine abschließende Spezialisierungsmöglichkeit besteht darin, Privilegien getrennt nach ihrer Art (anlegen, lesen, modifizieren, löschen) zu vergeben.

Um den durch die Detaillierung gestiegenen Verwaltungsaufwand zu verringern, ist die Zusammenfassung von Freigaben in Gruppen möglich. Diese werden einzelnen Benutzern z. B. in Form von Spezialisierungen der Entwicklerrolle zugewiesen. Ferner lassen sich Defaultregeln hinterlegen, die dem Erzeuger eines Elements alle Rechte bezüglich desselben zusichern, Mitgliedern seiner Gruppe lesenden und modifizierenden Zugriff erlauben und anderen prinzipiell Berechtigten Lesefreigaben zuordnen.

## 9.4 Funktionale Spezifikation

Mit den folgenden Ausführungen werden die zur Verwaltung und Entwicklung von Modellen im KMS erforderlichen Operationen dargestellt. Sie lassen sich in KM-, Modellier- und Berichtsoperationen klassifizieren. Aufgrund ihrer Bedeutung für die Abwicklung der Modellerstellung konzentrieren sich die folgenden Darstellungen auf KM- und Modellieroperationen. Die verbleibenden Funktionen zur Erstellung von Berichten sind stark von den Anforderungen des Projektes abhängig. Infolgedessen gibt auch DIN 10007 lediglich Anhaltspunkte zu ihrer Ausgestaltung (vgl. [Deut96], S. 16f). Dementsprechend sollten die gewünschten Berichte flexibel entsprechend den aktuellen Benutzeranforderungen erstellt werden können. Die folgenden Ausführungen konzentrieren sich auf die Beschreibung der Ziele und Inhalte der einzelnen Funktionen.

### 9.4.1 Konfigurationsmanagement-Operationen

Aufgabe der KM-Operationen ist insb. die Verwaltung verschiedener Zustände von Modellen und ihrer Elemente. Sie realisieren den Transfer von Modellversionen und ihrer Konfigurationseinheiten zwischen Workspaces und die Integration von separat erstellten Modellversionen zu

einem Gesamtmodell.

#### 9.4.1.1 CheckIn und CheckOut

Die zu einem Modell-KMS gehörenden Workspaces stehen untereinander in einer hierarchischen Beziehung, d. h. jeder Arbeitsbereich besitzt genau einen übergeordneten Workspace. Die Spitze der Hierarchie wird durch einen speziellen Workspace, das **Repository** gebildet. Er ist stets vorhanden, benötigt keinen übergeordneten Arbeitsbereich, und dient primär der Archivierung erstellter und ausgelieferter Modellversionen.

Der Transfer von Modellversionen erfolgt mit Hilfe von ***CheckOut*** - ***CheckIn*** Sequenzen. Dabei dient der ***CheckOut*** der Übertragung einer Modellversion in einen untergeordneten, der ***CheckIn*** in einen übergeordneten Workspace. Als Granulat der Operationen wird stets eine Konfigurationsversion festgelegt. Hiermit ist sichergestellt, dass alle relevanten KEs der Modellversion (inkl. der Metamodellversion und seiner KEs) in den Zielworkspace übertragen werden.

Als Varianten dieser Werkzeuge stehen des Weiteren ***Import*** und ***Export*** Operationen zur Verfügung. Mit Hilfe Ersterer können Modelle in das Modell-KMS integriert werden, die keine vorhergehende Version im übergeordneten Workspace besitzen. Ihrer Erstellung ging somit nicht ein ***CheckOut*** einer Ausgangsmodellversion voraus. Hiermit wird es dem Modellierer ermöglicht, losgelöst vom Repository zu arbeiten. Der ***Export*** kommt hingegen zum Einsatz, wenn eine Konfigurationsversion zwar in einen Zielarbeitsbereich transferiert, dort aber vor Änderungen geschützt werden soll (z. B. als Metamodellversion in der aktuellen Modellierung). Er dient gleichzeitig zur Auslieferung des Modells an den Kunden. Deshalb ist er mit geeigneten Konvertierverfahren zu verknüpfen, um das Produkt im gewünschten Format ausliefern zu können.

Die Anwendung des allgemeinen ***CheckIn***, also die Einbringung des aktuellen Modellstands als Nachfolgeversion des Ausgangsmodells im übergeordneten Workspace, ist jedoch nur zulässig, solange keine Arbeiten am Modell im übergeordneten Workspace seit dem ***CheckOut*** erfolgten. Ist dies nicht gegeben, so kommen ihre Spezialisierungen ***Merge*** bzw. ***Branch*** zum Einsatz.

#### 9.4.1.2 Merge

Die ***Merge*** Operation dient der Verschmelzung zweier Modellversionen, die auf einer gemeinsamen Ausgangsversion beruhen. Sie müssen dabei in verschiedenen Zweigen des Versionsgraphen liegen. Anderenfalls würden sie in einer Revisionsbeziehung stehen, die eine Verschmelzung unnötig macht. Im Ergebnis entsteht eine Modellversion, die die Veränderungen beider Modellversionen gegenüber der gemeinsamen Ausgangsversion enthält.

Aufgrund der gemeinsamen Ausgangsmodellversion kann zu jedem zu integrierenden KE festgestellt werden, ob es neu erzeugt oder verändert wurde. Aufbauend auf diesen Informationen kommen die jeweils geeigneten Verschmelzungsverfahren zum Einsatz. Wurde eine Konfigurationseinheit in beiden Modellversionen verändert, so ist das aus der Integration resultierende KE durch einen attributweisen ***Merge*** zu bilden.

### Bezug zur E$^3$-Methode

Für die folgenden Ausführungen wird eine Verschmelzung zweier Modellversionen $MC^{i+x}$ und $MC^{i+y}$ zu $MC^{i+x+1}$ angenommen, die sich im selben Workspace befinden. Beim ***Merge*** von $MC^{i+x}$ bzw. $MC^{i+y}$ ist zu beachten, dass existierende Abhängigkeitsbeziehungen auch in $MC^{i+x+1}$ wiederhergestellt werden müssen. Hieraus folgt, dass zur Verschmelzung von Versionen eines E$^3$-Elements zunächst die entsprechenden Versionen der jeweils vorgeordneten Elemente zu fusionieren sind. Im Anschluss kann geprüft werden, ob $MC^{i+x+1}$ auch eine Version der jeweils vorgeordneten Elemente enthält. Ist dies nicht der Fall, so ließe sich die Abhängigkeitsbeziehung für die resultierende Elementversion in $MC^{i+x+1}$ nicht wiederherstellen. Dementsprechend ist es nicht in $MC^{i+x+1}$ einzubringen. Der Versionsgraph für dieses Beispiel wird in Abbildung 79 dargestellt.

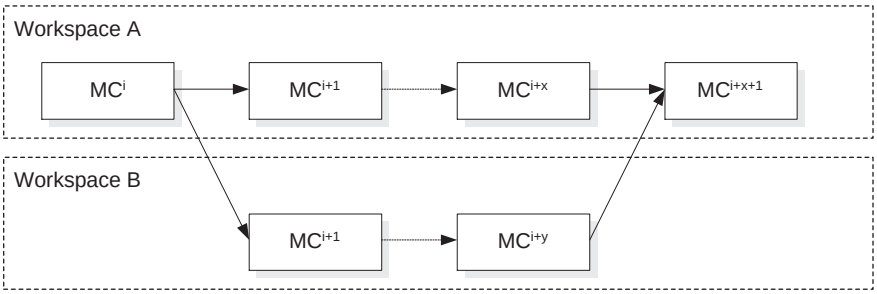



Abbildung 79: Beispielhafter Versionsgraph zur Merge Operation

Im Verschmelzungsergebnis $MC^{i+x+1}$ können sowohl Konsistenzregeln bez. der Kardinalitäten von Abhängigkeitsbeziehungen als auch bez. der maximal zulässigen Anzahl an Elementbeziehungen eines Property verletzt sein.[142]

[142]Dies soll im Folgenden anhand eines Beispiels verdeutlicht werden. Ausgangspunkt der Betrachtungen bildet eine Basis-Modellversion $MC^v$, die neben der Version eines Modells, auch Versionen eines Objekts und einer View enthält. Dem Objekt wurde des Weiteren ein Property zugeordnet, dessen aktuelle Version ebenfalls Bestandteil von $MC^v$ ist. Sein Propertytyp erlaubt maximal eine Elementbeziehung zu einem Viewobjekt. Auf Basis von $MC^v$ erfolgt in zwei getrennten Workspaces die Entwicklung zweier neuer Schemaversionen $MC^{v+s}$ und $MC^{v+t}$. Dabei wird jeweils ein Viewobjekt erzeugt und den bereits in $MC^v$ vorhandenem Objekt bzw. View nachgeordnet. Beide Viewobjekte sind weiterhin demselben Viewobjekttypen untergeordnet. Zusätzlich wird das erzeugte Viewobjekt mit dem ebenfalls bereits in $MC^v$ vorhandenen Property verbunden. Sowohl $MC^{v+s}$ als auch $MC^{v+t}$ erfüllen somit alle anwendbaren Konsistenzregeln des E$^3$-Modell.

Verletzungen bez. der maximal zulässigen Werteanzahl werden im Rahmen der Entwicklerfreigabe nach Abschluss der Modellierarbeiten geprüft und ggf. erzwungen. Sie sind somit nicht durch die ***Merge*** Operation zu behandeln. Die Verletzung von Konsistenzregeln bez. der Abhängigkeitsbeziehungen erfordert jedoch eine sofortige Behandlung im Rahmen des Verschmelzens, da sie eine weitere Modellierung im Schema unmöglich macht. Eine nachträgliche Feststellung derartiger Probleme zieht jedoch stets die Manipulation von Abhängigkeitsbeziehungen nach sich, die aber bereits explizit ausgeschlossen wurde. Deshalb ist die Spezifikation des Modell-KMS derart zu verändern, dass derartige Probleme stets vermieden werden.

Konsistenzverletzungen dieser Art entstehen, da unabhängig voneinander erzeugte Konfigurationseinheiten im KMS durch verschiedene Identifikatoren gekennzeichnet werden. Wird demnach in zwei Varianten eines Modells jeweils ein Viewobjekt erzeugt und demselben View-Objekt-Paar sowie demselben Viewobjekttyp zugeordnet, so stellen sie aus Sicht des KMS aufgrund ihrer Identifikatoren verschiedene Konfigurationseinheiten dar. Im Kontext des $E^3$-Modells sind sie jedoch identisch, da sie jeweils das gleiche Objekt mit der gleichen View verknüpfen.

Identische Ausprägungen der oben aufgeführten $E^3$-Elemente zeichnen sich durch Abhängigkeitsbeziehungen zu den gleichen vorgeordneten Elementen aus. Um diesen Sachverhalt im Modell-KMS geeignet zu repräsentieren, ist der Identifikator der betreffenden Konfigurationseinheiten aus denen ihrer vorgeordneten Elemente zusammenzusetzen. Zu einem Viewobjekt und einer Präsentation können jedoch mehrere Präsentationsobjekte mit jeweils verschiedenen Präsentationsobjekttypen gehören. Die Konstruktion bezieht daher auch den Identifikator des jeweiligen Typen mit ein. Deshalb besitzen z. B. Versionen unabhängig voneinander erzeugter Viewobjekte gleichen Viewobjekttyps, die denselben vorgeordneten Elementen zugeordnet wurden, identische Identifikatoren. Sie können somit problemlos im Rahmen der ***Merge*** Operation miteinander verschmolzen werden.

#### 9.4.1.3 Branch

Mit Hilfe der ***Branch*** Operation kann ein neuer Zweig im Entwicklungspfad eines Modells eröffnet werden. Sie kommt zum Einsatz, wenn während der Arbeit an einem Modell im übergeordneten Workspace Weiterentwicklungen an der Ausgangsmodellversion vorgenommen wurden. Im Gegensatz zur ***Merge*** Operation sollen die Ergebnisse jedoch nicht miteinander verschmolzen werden, sondern als Modellvarianten gleichberechtigt nebeneinander existieren.

Zum Abschluss der Entwicklungsarbeiten werden $MC^{v+s}$ und $MC^{v+t}$ zu einer gemeinsamen Schemaversion $MC^{v+s+1}$ verschmolzen. Im Ergebnis dieser Operation sind dem bereits in $MC^{v}$ vorhandenen View-Objekt-Paar zwei unterschiedliche Viewobjekte zugeordnet (Verletzung der Kardinalitäten von Abhängigkeitsbeziehungen). Beide Viewobjekte sind jedoch semantisch identisch, da sie das gleiche Objekt mit der gleichen View verbinden. Gleichzeitig ist das Property mit zwei verschiedenen Viewobjekten verknüpft (Verletzung der maximal zulässigen Werteanzahl).

Bei der Durchführung des ***Branch*** ist zu beachten, dass für die einzelnen Bestandteile der neuen Modellvariante bereits Revisionen der Ausgangsversion existieren können. In diesem Fall sind sie ebenfalls als Varianten der vorhandenen Revisionen in den Zielworkspace und die Modellvariante einzubringen.

#### 9.4.1.4 Lend

Im Rahmen der Anforderungsanalyse des Kapitels 4.5 wurde deutlich, dass im KMS auch die Kooperation zwischen den Modellerstellern zu unterstützen ist. Dies kann bereits über die Nutzung der ***CheckOut*** und ***CheckIn*** Operationen realisiert werden, die den Austausch von Modellversionen über einen gemeinsamen übergeordneten Workspace erlaubt. Im Gegensatz dazu beschreibt die Verleihoperation das direkte Übertragen eines KE. Existierten dabei im Ziel-Workspace bereits Versionen dieses KE, so wird es als aktuelle Revision eingebracht. Anderenfalls bildet es nach entsprechender Anpassung der Versionsnummer die initiale Version des KE im Ziel-Workspace.

Die ***Lend*** Operation ist nicht für Konfigurationen und Konfigurationsstrukturen definiert. Ein Austausch ersterer kann bereits durch Anwendung der ***CheckOut*** und ***CheckIn*** Operationen realisiert werden. Das Verleihen von Konfigurationsstrukturversionen bedingt konsequenterweise ebenfalls eine Überlassung der in den Beziehungen referenzierten KEs, um eine Verfälschung des Modells bzw. die Verletzung von Konsistenzbedingungen zu vermeiden. Hierzu ist wiederum ein Transfer der zugehörigen Modellversion notwendig. Deshalb können lediglich unabhängige KEs Gegenstand der ***Lend*** Operation sein.

Allerdings gestaltet sich das Verleihen von Elementen der Instanzenebene aufgrund der extern gespeicherten Typbeziehung problematisch. Infolgedessen werden zusätzliche Einschränkungen für den Austausch einer Modellelementversion $ME^i$ erforderlich. Zunächst ist sicherzustellen, dass die bearbeiteten Modellversionen im Quell- und Ziel-Workspace dieselbe Metamodellversion verwenden. Ist das durch $ME^x$ repräsentierte Modellelement in einer beliebigen Version ebenfalls Bestandteil der aktuellen Modellversion im Ziel-Workspace, so wird weiterhin gefordert, dass beide dieselbe Typbeziehung aufweisen. In allen anderen Fällen ist ein Austausch nicht durchführbar.

Nach Abschluss der Operation sind in beiden Workspaces Modifikationen an der betreffenden Konfigurationseinheit zulässig. Aufgrund der Spezifikation des hier verwendeten ***Merge*** werden parallel vorgenommene Änderungen spätestens bei der Integration der jeweiligen Modellversionen mit der Basismodellversion zusammengeführt.

| Operation | Beschreibung | Beispiel der Anwendung | Unterarten |
|---|---|---|---|
| CheckOut | Übertragung einer Modellversion in einen untergeordneten Workspace in Form einer Konfiguration | Parallele Bearbeitung und Verteilung von Modellversionen | Export |
| CheckIn | Übertragung einer Modellversion in einen übergeordneten Workspace in Form einer Konfiguration | Zusammenführung der parallel erstellten Modellversionen | Import, Merge, Branch |
| Import | Integration von Modellversionen, die keine vorhergehende Version im übergeordneten Workspace besitzen | Vom Repository losgelöstes Arbeiten an einem Modell | |
| Export | Transformation einer Konfigurationsversion in einen Ziel-Workspace mit Schutz vor Änderungen | Für Metamodellversionen oder an Kunden ausgelieferte Modelle | |
| Merge | Verschmelzung zweier Modellversionen aus verschiedenen Zweigen des Versionsgraphen, die auf einer gemeinsamen Ausgangsversion beruhen | Einbringung eines Modellstands als Nachfolgeversion im übergeordneten Workspace, wenn Arbeiten am Modell seit dem CheckOut erfolgten | |
| Branch | Eröffnen eines neuen Entwicklungspfades eines Modells | Keine Verschmelzung der Ergebnisse, sondern gleichberechtigte Modellvarianten | |
| Lend | Austausch von KEs (außer Konfiguration und Konfigurationsstruktur) | Übertragung von Modellelementen in beliebige Modelle | |

Tabelle 12: Operationen im Workspace

### 9.4.2 Modellieroperationen

Ausgangspunkt eines jeden Modellierungsprojektes ist die Erstellung einer initialen Konfiguration durch Ausführung der ***CreateModel*** Operation. In ihrem Verlauf sind zumindest die Konfiguration und die Konfigurationsstruktur zu erzeugen. Außerdem ist ebenfalls das eingesetzte Metamodell in die Konfiguration einzutragen. Inwiefern durch die Operation zusätzlich auch Namenskonvention, Problembezug und KMP zu erzeugen sind, hängt vom geforderten Mindestmodellumfang im Projekt ab.

Das Hinzufügen bzw. Entfernen von KEs aus einer Konfigurationsversion erfolgt durch Anwendung der ***AddElement*** bzw. ***RemoveElement*** Operationen. Betrifft ihre Ausführung Versionen von Modellelementen, so sind zusätzlich auch die erforderlichen Typbeziehungen in der Konfigurationsstruktur zu erstellen bzw. zu löschen. Hierdurch wird ebenfalls eine neue Version derselben erzeugt. Mit der notwendigen Aktualisierung der Konfiguration bezüglich des Modellelements ist gleichzeitig auch die aktualisierte Konfigurationsstruktur in die neue Modellversion einzubringen.

Zur Erstellung bzw. Aufhebung modellinterner Beziehungen durch den Modellierer werden entsprechende ***ConnectElements*** bzw. ***DisconnectElements*** Operationen definiert. Sie nehmen die jeweiligen Veränderungen in der Konfigurationsstruktur vor und aktualisieren im Anschluss die Konfiguration. Beim Einsatz des Modell-KMS für einen konkreten Modelltypen ist hierbei zusätzlich eventuellen Beschränkungen der Kardinalitäten Rechnung zu tragen.

Die ***Undo*** Operation dient dem schrittweisen Zurücknehmen von Benutzeroperationen. Aufgrund des hier verfolgten Ansatzes werden sie stets durch eine neue Version des Modells repräsentiert. Dementsprechend reduziert sich ein ***Undo*** auf das Verwerfen der aktuellen Konfigurationsversion, womit die dann aktuelle Version im Workspace den Modellstand vor Anwendung der betreffenden Funktion repräsentiert. Hierbei werden nicht länger erforderliche KEs durch die Überführung in den Zustand *obsolete* als ungültig markiert.

Aufgrund der identischen Behandlung von Modell und Metamodell als Konfigurationen im Modell-KMS ist es auch möglich, einen Austausch des Metamodells in der Entwicklung vorzunehmen. Da hiermit der Charakter des Modells nachhaltig verändert wird, kann dieser Schritt nur durch die Erstellung eines neuen Modells geeignet repräsentiert werden. Es entspricht dem Stand der zugrunde liegenden Modellversion, verwendet aber bereits das neue Metamodell. Modellelemente, deren Typen im neuen Metamodell nicht mehr vorhanden sind, und denen kein neuer Typ durch den Modellierer zugewiesen werden kann, werden nicht in das neue Modell übernommen.

### Bezug zur E$^3$-Methode

Mit den Typ- und Abhängigkeitsbeziehungen aller Elemente der Typ-, Instanzen- und Vorgangsebene existieren Konsistenzregeln im E$^3$-Modell, die durch eine entsprechend angepasste ***AddElement*** Operation geprüft und gewährleistet werden müssen. So werden alle Beziehungen durch die Operation gemeinsam mit dem Element angelegt. In Ergänzung existieren bez. der Kardinalitäten von Abhängigkeitsbeziehungen zusätzliche Beschränkungen.[143] Aus diesem Grund ist die bisherige Spezifikation an entsprechender Stelle so zu verändern, dass derartige Konsistenzverletzungen vermieden werden.

Aktualisierungen eines E$^3$-Elements durch Anwendung der ***UpdateElement*** Operation sind nur für E$^3$-Elemente zulässig, die neben ihren Beziehungen zu anderen Elementen weitere Eigenschaften besitzen.

Das Entfernen eines E$^3$-Elements aus einem Metamodell bzw. Modell erfordert zunächst die korrekte Auflösung seiner Abhängigkeitsbeziehungen. Demzufolge werden in einem ersten Schritt rekursiv alle ihm direkt bzw. indirekt nachgeordneten Elemente entfernt.[144] Hieraus ergeben sich für die Nutzung der ***RemoveElement*** Operation zu beachtende Konsequenzen. Soll durch ihre Anwendung die Version eines E$^3$-Elements aus der aktuellen Konfigurationsversion gelöscht werden, so ist sie zunächst für alle Versionen ihm direkt nachgeordneter Elemente auszuführen, wodurch sie kaskadisch aus der Konfigurationsversion entfernt werden. Trotz der möglicherweise umfangreichen Veränderungen des Modells bzw. Schemas sollte jeweils nur eine Revision von Konfiguration und Konfigurationsstruktur erzeugt werden, da aus Sicht des Benutzers nur ein E$^3$-Element entfernt wurde.

Bezüglich des prinzipiellen Verlaufs der ***ConnectElements*** und ***DisconnectElements*** Operationen sind ebenfalls keine grundsätzlichen Modifikationen erforderlich. Da jedoch nur Elementbeziehungen separat erstellt bzw. gelöst werden dürfen, machen sich zusätzliche Prüfungen erforderlich. Zur Erzeugung der Verknüpfungen ist sicherzustellen, dass eines der Partnerelemente ein Property ist. Dessen Propertytyp muss sie sowohl bezüglich des Wertebereichs als auch der Werteanzahlsgrenzen erlauben. Im Gegensatz dazu ist beim Trennen zuerst zu prüfen, ob die betreffende Verbindung eine Elementbeziehung darstellt. Außerdem muss der Propertytyp des beteiligten Property ihre Auflösung erlauben (Beschränkung bez. minimaler Wertanzahl).

[143] So darf z. B. einem Objekt und einer View in einem Schema nur ein Viewobjekt zugeordnet werden.

[144] So sind z. B. im Falle eines ViewObjekts alle direkt oder indirekt nachgeordneten Viewobjekte, Präsentationsobjekte, Viewproperties und Präsentationsproperties aus dem Modell zu löschen.

| Operation | Beschreibung | Beispiel der Anwendung | Eigentümer |
|---|---|---|---|
| CreateModel | Erzeugen einer Konfiguration und Konfigurationsstruktur | Erstellung einer initialen Konfiguration | Workspace |
| AddElement | Hinzufügen eines KE | Modellierung zusätzlicher Inhalte | Konfiguration |
| RemoveElement | Entfernen eines KE | Löschen von Informationen | Konfiguration |
| ConnectElements | Definition einer Beziehung zwischen zwei Elementen | Verbinden von zwei Elementen | Konfigurationsstruktur |
| DisconnectElements | Löschen einer Beziehung zwischen zwei Elementen | Trennen von zwei Elementen | Konfigurationsstruktur |

Tabelle 13: Operationen auf Konfigurationselemente

## 9.5 Integration der Teilspezifikationen

Mit den Ausführungen zur strukturellen, organisatorischen und funktionalen Spezifikation wurden die notwendigen Grundlagen zur Verwaltung von Modellen in verschiedenen Entwicklungszuständen geschaffen. Die Anforderungsanalyse des Kapitels 4.5 forderte aber auch eine Unterstützung des Entwicklungsprozesses durch das KMS. Ein fest definiertes Vorgehensmodell im KMS steht jedoch im Gegensatz zur notwendigen Kreativität im Modellbildungsprozess (vgl. Abschnitt 4.5.3).

Um die konträren Forderungen möglichst optimal unterstützen zu können, wird im hier entwickelten Konzept auf die Einbeziehung eines starr definierten Vorgehensmodells verzichtet. Stattdessen stehen den KMS-Benutzern prinzipiell alle Strukturen und Funktionen jederzeit zur Verfügung. Die tatsächliche Verfügbarkeit wird jedoch durch drei zusätzliche Komponenten eingeschränkt. Zum einen durch den Typ der bearbeiteten Konfigurationseinheit, zum zweiten durch den Entwicklungsstand des aktuellen KE und abschließend durch die Zugriffsrechte des aktuellen Benutzers. Hiermit entsteht ein Filtermodell zur Bestimmung aktuell verfügbarer Funktionen für den jeweiligen Benutzer. Abbildung 80 zeigt den Ansatz im Überblick.

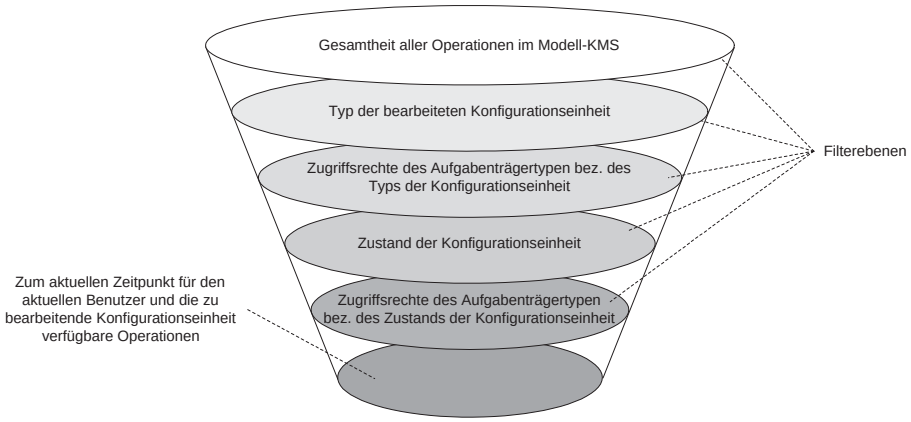



Abbildung 80: Filterung der Operationen im KMS

Mit der Verknüpfung ihrer Ausführbarkeit mit dem Typ der bearbeiteten Konfigurationseinheit wird eine korrekte und sinnvolle Anwendung der Funktionen sichergestellt. Das Heranziehen des Entwicklungsstands eines KE zur Bestimmung der Operationsverfügbarkeit dient dagegen der Einhaltung zwingender Meilensteine im Projekt. So sind beispielsweise die erstellten KEs vor der Auslieferung geeigneten Tests bzw. Qualitätssicherungsmaßnahmen zu unterziehen. Um die Validität dieser Maßnahmen jedoch nicht zu gefährden, dürfen mit Beginn des Tests keine Veränderungen an den zu prüfenden Versionen mehr vorgenommen werden. Deshalb sind in diesem Entwicklungsabschnitt keinerlei Modellieroperationen mehr erlaubt. Die Integration des Ansatzes in das KMS erfolgt durch die Einführung eines Zustandsmodells

für KEs, dessen Zustände die einzelnen Entwicklungsschritte (Erstellung, Entwicklerfreigabe, Test, Qualitätssicherung, Auslieferung) widerspiegeln. Mit ihnen sind jeweils Beschränkungen für einzelne Operationen verknüpft, die dem Anhang zu entnehmen sind.

Mit der Spezifikation des Rollenmodells in Abschnitt 9.3 wurden erste Maßnahmen zur Vermeidung von Interessenskonflikten zwischen den unterschiedlichen Benutzergruppen geschaffen. Aufgrund der fehlenden Integration der funktionalen Spezifikation stehen jedoch allen Benutzern bisher stets sämtliche verfügbaren Operationen zur Verfügung. Somit ist prinzipiell auch ein Modellersteller berechtigt, die Überprüfung der Qualitätssicherungsmaßnahmen am erstellten Modell vorzunehmen. Dies widerspricht jedoch der Zielsetzung des Rollenmodells. Aus diesem Grund werden mit den einzelnen Rollen Ausführungsrechte verknüpft, die die Menge verfügbarer Operationen auf die jeweils zulässigen beschränken.

Allerdings ist dieser Integrationsansatz nicht ausreichend. So erhält z. B. die Projektleitung Zugriff auf alle verfügbaren Konfigurationseinheiten, um sie zur Auslieferung an den Kunden freizugeben. Damit steht ihr jedoch auch die Möglichkeit zur Manipulation von Modellelementen offen, obwohl dies lediglich für Modellentwickler zulässig ist. Aus diesem Grund sind die Berechtigungen einer Rolle zusätzlich mit dem Typ der zu bearbeitenden Konfigurationseinheit und dem Zustand des aktuellen KE zu verknüpfen.

## 9.6 Change Request Prozess

Mit Hilfe der bisher dargestellten Strukturen und Operationen sowie des in Abschnitt 9.5 umrissenen Integrationsansatzes ist eine Verwaltung von Modellen und ihren Veränderungen inkl. der erforderlichen Dokumentation sichergestellt. Allerdings erforderte die Anforderungsanalyse in Kapitel 4.5 nicht nur die Aufzeichnung, sondern auch die Kontrolle von Änderungen. Dies bedeutet, dass Change Requests vor ihrer Realisierung zunächst zu analysieren sind und anschließend durch eine zentrale Instanz genehmigt werden müssen. Ein derart starres Vorgehen in allen Projektphasen würde jedoch nicht den alternierenden Erfordernissen im Zeitablauf gerecht werden. So sind z. B. Veränderungen in den frühen Phasen der eigentlichen Modellentwicklung bzw. Abstimmungen zwischen den Erstellern geschuldet und nicht durch wechselnde Anforderungen der Auftraggeber bedingt.

Aus diesem Grund wird hier ein adaptierbarer Change Request Prozess nach den Ausführungen von PRESSMAN (vgl. [Pres92], S. 706) eingesetzt. Er gliedert das Projekt in drei Phasen, die jeweils unterschiedliche Anforderungen an die Formalisierung von Change Requests stellen.

In der ersten Phase vom Projektstart bis zur Erstellung einer ersten stabilen Version (Bezugskonfiguration) sind Änderungen primär der Erstellung einer ersten Modellversion geschuldet. Ihre separate Erfassung und Genehmigung würde zu einer Überlastung der Genehmigungsinstanz mit nachfolgender Verzögerung der Gesamtentwicklung führen. Aus diesem Grund

kommt im betreffenden Zeitraum ein *informaler Change Request Prozess* zum Einsatz. Er erfordert keinerlei Dokumentation bzw. Genehmigung von Änderungen.

Der Abschnitt von der Erstellung der ersten BK bis zur Auslieferung einer ersten Modellversion an den Kunden wird von einem *projektbezogenen Change Request Prozess* begleitet. Zu realisierende Änderungen sind zwar immer noch primär durch Entwickler motiviert, allerdings ist dem Zusammenwirken unterschiedlicher Modellierergruppen auf Basis einer gemeinsamen BK Rechnung zu tragen. Deshalb sind Änderungsanträge stets mit dem betreffenden (Teil-)Projektleiter abzustimmen.

Nach Auslieferung einer ersten Modellversion an den Kunden tritt hingegen der *formale Change Request Prozess* in Kraft. In dieser Phase notwendige Änderungen basieren nicht mehr auf den Entwicklungsarbeiten der Modellierer, sondern auf Änderungs- bzw. Erweiterungswünschen des Kunden. Diese können als Folgeaufträge der Entwicklung der Basismodellversion aufgefasst werden. Deshalb sind sie vor ihrer Realisierung zu evaluieren, der notwendige Aufwand zu schätzen und geeignete Mittel zur Projektverfolgung einzuführen. Weiterhin sind sie stets schriftlich in Form von CRDs zu formulieren und beim CCB zur Abnahme einzubringen. Die Phase des formalen Change Request Prozesses endet erst mit der Einstellung aller Entwicklungsarbeiten am Modell.

Die Integration des dargelegten adaptiven Change Request Prozesses in das Modell-KMS erfolgt durch die Einführung eines zusätzlichen Zustandsmodells (*created, baselined, delivered*) für Konfigurationen. Es spiegelt jedoch nicht die Entwicklung einer Konfigurationsversion, sondern des Gesamtmodells wieder. Deshalb wird bei Ableitung einer neuen Version der aktuelle Zustand der Ausgangsversion mit übergeben.

| Phase | Art der Änderungen | Change Request Prozess |
|---|---|---|
| Projektstart bis Erstellung einer Bezugskonfiguration (BK) | Primär der Erstellung einer ersten Modellversion geschuldet | Informal (keinerlei Dokumentation und Genehmigung von Änderungen) |
| Erstellung der BK bis Auslieferung einer ersten Modellversion an den Kunden | Primär durch Entwickler motiviert, dem Zusammenwirken unterschiedlicher Teams auf Basis einer gemeinsamen BK ist Rechnung zu tragen | Änderungsanträge sind mit (Teil-) Projektleiter abzustimmen |
| Nach Auslieferung einer ersten Modellversion an den Kunden | Primär auf Basis von Änderungs- bzw. Erweiterungswünschen des Kunden | Schriftliche CRDs, Abstimmung im CCB |

Tabelle 14: Phasen des Change Request Prozesses

# 10 Bewertung der E$^3$-Methode

Die mit dem Kapitel III vorgestellte Methode des ME wird in diesem Abschnitt mit den gleichen Bewertungsmaßstäben wie die Methoden des Abschnittes 5 untersucht. Entsprechend erfolgt im folgenden Abschnitt eine zusammenfassende Darstellung der Methode, die zugleich auch die Ergebnisse dieser Arbeit aufzeigt. Mit dem Abschnitt 10.3 werden insbesondere die Punkte aufgeführt, für die diese Arbeit eine Grundlage bilden kann.

## 10.1 E$^3$-Methode

Die E$^3$-Methode beschreibt einen vollständigen Ansatz zur Entwicklung von Methoden. Die ursprünglichen Konzepte wurden im Rahmen einer Diplomarbeit zur prototypischen Implementierung eines Meta-CASE-Werkzeuges entwickelt. Die Methode erhebt in der vorliegenden Version jedoch den Anspruch auf umfassendere Einsatzgebiete, die die Erstellung von Modellen erfordern. Im Gegensatz zu den anderen im Abschnitt 5.3 vorgestellten Methoden, verfügt die E$^3$-Methode über einen starken Fokus auf den Prozess der Erstellung von Methoden. Der Bezug der E$^3$-Methode zum Methoden- und Modellbegriff wird in Abbildung 81 dargestellt.

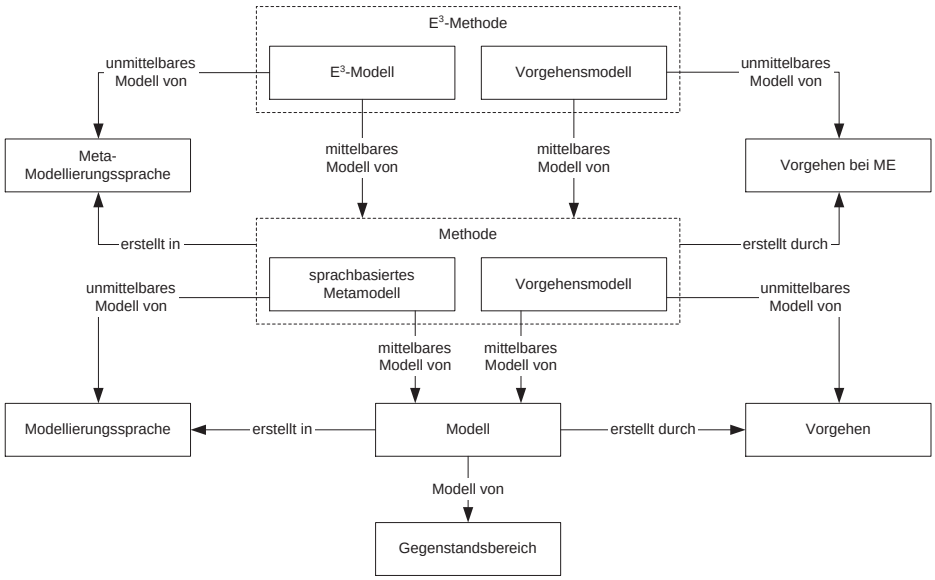



Abbildung 81: Bezug der E$^3$-Methode zum Methoden- und Modellbegriff

### Reifegrad und Validierung

Die Validierung des Ansatzes erfolgt zum einen durch den praktischen Einsatz eines im ursprünglichen Sinne universitären Prototypen[145] in mehreren Praxisprojekten (vgl. [semt02]) und zum anderen auf wissenschaftlichem Gebiet durch diese Arbeit. Die in den Ansatz integrierten Verfahren des Quality Function Deployment haben ihre Wirksamkeit in zahlreichen unterschiedlichen Anwendungsfällen nachgewiesen.[146]

### Eigenschaften der Metamodellsprache

Die Metamodellsprache stellt Konzepte zur Beschreibung von Abläufen, Produkten und Organisationen beim ME zur Verfügung. Der Ansatz beinhaltet explizit die Erfassung grafischer Darstellungsformen und die Bildung von Sichten auf Produkte des ME. Das zugrunde liegende E$^3$-Modell erfasst sowohl Elemente der Typebene zur Beschreibung der Metamodelle als auch Elemente der Instanzenebene zur Abbildung von zugeordneten Modellen. Als Besonderheit enthält die Sprache zur Produktbeschreibung nicht die bei den anderen Ansätzen üblichen Objekte und deren Beziehungen, sondern Beziehungen werden selbst als Objekte aufgefasst. Die Möglichkeit der Zuordnung von mehrstufigen Sichten und Präsentationen ist ebenfalls in dieser Form bei den anderen Ansätzen nicht vorhanden.

Die Abbildung 82 zeigt u. a. , dass die Erfassung von Vorgehens- und Organisationsmodellen vollständig in die Definition der Produkte integriert erfolgt.

### Eigenschaften des Modellierungsprozesses

Der Prozess ist durchgehend dokumentiert in semiformaler und informaler Form. Es werden alle Aufgabentypen im Rahmen des ME beschrieben. Der Ansatz wird ergänzt durch die Wiederverwendungsansätze *Typisierungsmuster* und *Referenzmetamodell*. Sowohl für die Modelle als auch deren Metamodelle wird ein Konfigurationsmanagement spezifiziert und eine organisatorische Einbettung vorgeschlagen. Das KM definiert durch seinen allgemeinen Ansatz einen Prozess der Änderungssteuerung und -dokumentation, der sich nicht nur auf die zu entwickelnde Methode bezieht, sondern auch eine Integration in die Projekte, die die Methode anwenden, gestattet.

[145] Auf den *Generischen Modelleditor (GME)* wurde in dieser Arbeit nicht weiter eingegangen, trotzdem sei an dieser Stelle der Hinweis erlaubt, dass im entsprechenden Projekt zu dessen Entwicklung (vgl. [Grei02]) die getroffen Spezifikationen zur E$^3$-Methode die Grundlage bilden.

[146] Eine ständig aktualisierte Auflistung liefert [Herz01].

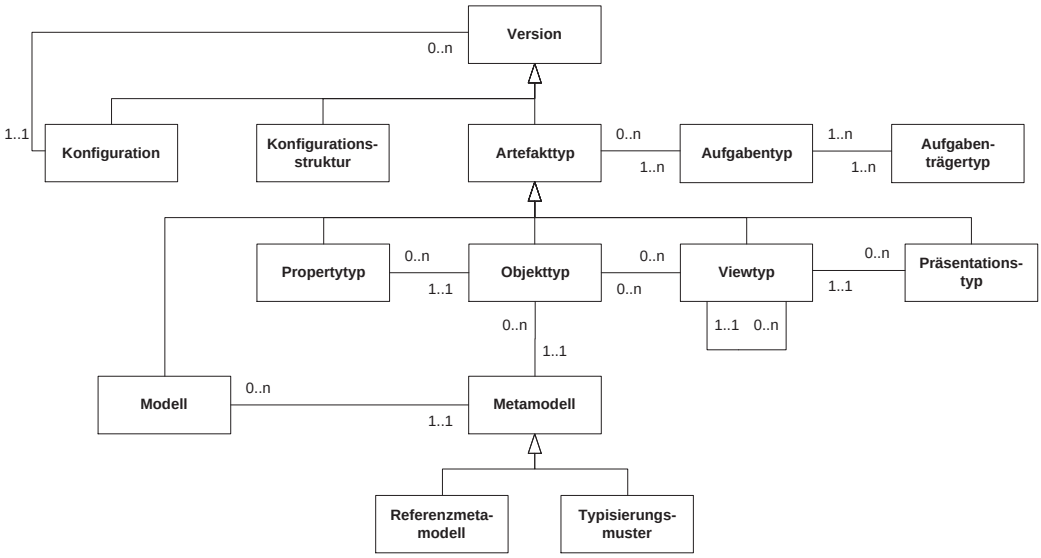



Abbildung 82: Vereinfachtes Metamodell der E$^3$-Methode

## 10.2 Bewertung der E$^3$-Methode

Entsprechend der Kriterien, die in Abschnitt 5.4 bereits auf andere Methoden des ME angewendet wurden, ergibt sich für die E$^3$-Methode die in Tabelle 15 dargestellte Bewertung. Da die Entwicklung der Methode auf die Kriterien aus Abschnitt 5 ausgerichtet war, sind Schwachstellen nur im Punkt *Reifegrad und Validierung* erkennbar. Durch eine Anpassung der Merkmale und Skalen, den Anforderungen und deren Korrelation lässt sich die Art der Bewertung verändern. Für die Erhebung zusätzlicher bzw. Veränderung vorhandener Methodenmerkmale kommen vor allem folgende Punkte in Frage:

- Anpassung der ontologischen Qualifizierung: Hier ist sowohl eine detailliertere Betrachtung der Eigenschaften möglich, die sich nicht nur auf Vollständigkeit und Konsistenz bezieht, als auch eine Anpassung der von der Methode geforderten Konzepte.

- Einfügen neuer Merkmalsgruppen: Bisher nur in geringem Umfang vertretene Merkmale können zu Merkmalsgruppen ausgebaut werden, z. B. kann so einer Forderung nach Beschreibung eines Qualitäts- oder Projektmanagements Ausdruck verliehen werden.

- Ergänzung von Merkmalsgruppen: Die Ergänzung von Merkmalen zu den drei vorhandenen Gruppen ist ebenfalls eine denkbare Anpassung. Beispielsweise lässt sich dem Punkt *Reifegrad und Validierung* eine genauere Untersuchung des Lebenszyklus der Methode hinzufügen.

Das Entfernen ganzer Merkmalsgruppen erscheint unter dem Gesichtspunkt der in dieser Arbeit vorgestellten Methodendefinition nicht sinnvoll. Bei einer Veränderung der Merkmale

ist vor allem der Checkliste von HERZWURM Beachtung zu schenken (siehe Anhang 13.3.3).

| Kriterium | E$^3$-Methode |
|---|---|
| **Reifegrad und Validierung** | |
| Versionen | 3 |
| Werkzeuge | 9 |
| Veröffentlichungen | 0 |
| Interne Validierung | 3 |
| Externe Validierung | 3 |
| Dokumentation | 9 |
| **Eigenschaften der Metamodellsprache** | |
| Ontologische Klarheit | 9 |
| Ontologische Vollständigkeit | 9 |
| Formalität | 3 |
| Konzepte und Beziehungen | 9 |
| Darstellungstechniken | 9 |
| Organisatorische Rahmenbedingungen | 9 |
| Abstraktionsebenen | 9 |
| Regeln | 9 |
| Dokumentation | 9 |
| Sichten | 9 |
| Vorgehensmodelle | 9 |
| **Eigenschaften des Modellierungsprozesses** | |
| Wiederverwendungsansätze | 9 |
| Organisationsmodell | 9 |
| Integration der Modellierungssprache | 9 |
| Modularität | 9 |
| Qualitätssicherung | 9 |
| Vorgehens- und Phasenmodell | 9 |
| Konfigurationsmanagement | 9 |

Tabelle 15: Bewertung der E$^3$-Methode

## 10.3 Ausblick

Im Zentrum dieser Arbeit stand das Method Engineering als wesentliches Hilfsmittel bei der Gestaltung ingenieurmäßiger Verfahren. Deren Anwendungsgebiet ist vielfältig und bezogen auf den Erkenntnisgegenstand der Wirtschaftsinformatik liegt hier ein großes Gestaltungspotential für die Lösung zukünftiger Probleme. Den Schwerpunkt der Ausführungen bildete die Spezifikation der E$^3$-Methode, deren Bewertung im letzten Teil zeigt insbesondere die Überlegenheit gegenüber anderen Methoden in Bezug auf die Beschreibung geeigneter *Verfahren* zur Methodenkonstruktion und -evaluierung. Dieses Ergebnis beruht auf der Tatsache, dass die Entwicklung der E$^3$-Methode auf die Erfüllung der zur Bewertung herangezogenen Kriterien hin ausgelegt wurde. Mit dieser Arbeit entstand aber auch ein anpassbarer Kriterienkatalog, der eine Untersuchung und Einordnung der Methode unter anderen Gesichtspunkten unterstützt. Zudem wird mit der Methode auch ein Großteil der Ergebnisse der Arbeiten von TOLVANEN,

BRINKKEMPER oder FRANK auf diesem Gebiet geeignet integriert, sodass mit dem $E^3$-Modell eine ausdrucksstarke Modellierungssprache für die Abbildung von Methoden der Wirtschaftsinformatik existiert. Die Ausführungen des Teil I belegen beispielhaft, wie uneinheitlich dabei dieser zentrale Begriff in der Wirtschaftsinformatik verwendet wird.

Die Integration der $E^3$-Methode in Methoden der Systemanalyse und -entwicklung bildete keinen Schwerpunkt dieser Arbeit. In diesem Zusammenhang erscheint eine Kopplung des Ansatzes mit praxisbewährten Methoden der Wirtschaftsinformatik besonders reizvoll. Diese Kombination wird zudem der $E^3$-Methode zu einer höheren Akzeptanz in der Praxis verhelfen, zumal ein systematisches bzw. methodisches Vorgehen in der Systementwicklung heute noch nicht selbstverständlich ist. Praktische Erfahrungen der semture GmbH zeigen aber, dass ein starkes Bedürfnis existiert, Verfahren und Darstellungstechniken vor deren Einsatz anzupassen und dieses Wissen für spätere Projekte zu dokumentieren. In diesem Zusammenhang scheint die Ablage von Vorgehensmodellen in einem entsprechenden Referenzmetamodell zahlreiche Vorteile zu bieten. Weiterer Forschungsbedarf existiert hier vor allem bei der Entwicklung eines entsprechenden Referenzmetamodellrahmens. Dieser muss sowohl die Verfahren geeignet strukturieren als auch eine Integration der Konzepte des mit dieser Arbeit vorgestellten Referenzmetamodells ermöglichen.

Das dem $E^3$-Modell zugrunde liegende Konfigurationsmanagement definiert die notwendigen Strukturen und Funktionen zur Verwaltung von Modellen und ihrer Entwicklungsgeschichte. Die Unterstützung der Softwareentwicklung durch den Einsatz von Konfigurationsmanagementsystemen ermöglichte bedeutende Qualitätsverbesserungen. Aufgrund der engen Beziehung zwischen Software und den in dieser Arbeit betrachteten Informationsmodellen können bei Anwendung der $E^3$-Methode ähnliche Potentiale bereits im Modellbildungsprozess realisiert werden. Das hierzu in dieser Arbeit entwickelte Konzept trägt insb. dem Anspruch Rechnung, Veränderungen nicht nur zu verwalten, sondern auch zu kontrollieren. Um eine ganzheitliche Unterstützung der Modellerstellung zu gewährleisten, bezieht der Entwurf außerdem Aufgaben und Ziele potentieller Nutzergruppen in die Konzeption mit ein. Die Anwendung der Spezifikation im $E^3$-Modell zeigt, dass hiermit die angestrebten Ziele erreicht werden können. Um jedoch einen effizienten Einsatz der beschriebenen Mittel zu ermöglichen, sind weiterführende Arbeiten erforderlich. Wie die Ausführungen von BROWN ET AL. darlegen, setzt sich eine vollständige KM-Unterstützung aus den Teilbereichen KM-Primitive, KM-Protokolle und KM-Strategien zusammen (vgl. [Bro+91], S. 44ff). Der entwickelte Ansatz deckt jedoch nur die Ebenen der Primitive und Protokolle ab. In kommenden Arbeiten sind deshalb geeignete Nutzungskonzepte zur Anwendung der Konstrukte zu entwickeln. Mit der hier verfolgten Prämisse der Gewährung maximaler Freiheiten für die Projektbeteiligten bei gleichzeitiger Sicherstellung organisatorischer Notwendigkeiten wurde bereits der verfügbare Rahmen möglicher Strategien

definiert.

Ein Großteil des Nutzens der vorgestellten Methode erwächst aus der Möglichkeit, diese durch geeignete Werkzeuge zu unterstützen. Existierende Ansätze in Modellierungswerkzeugen müssen jedoch insbesondere um eine Unterstützung der angewendeten Prozesse ergänzt werden. Die dafür notwendige Voraussetzung - die Transformation der durch Activity Graphs spezifizierten Vorgehensmodelle in eine interpretierbare formale Sprache - wird in dieser Arbeit nicht beschrieben, u. a. weil der Entwurf einer solcher Process Modeling Language (PML) weiterhin umstritten ist (vgl. [CoLi95], S. 98ff). Absehbar ist jedoch eine immer höhere Automation der Softwareentwicklung selbst, die dafür notwendigen Fachkonzepte können mit der E$^3$-Methode erstellt werden.

# Literaturverzeichnis


[Adam96] ADAMS, S.: *The Dilbert Principle*. London: United Feature Syndicate Inc., 1996

[Akao92] AKAO, Y.: *QFD - Quality Function Deployment*. München: Verlag Moderne Industrie, 1992

[Alex79] ALEXANDER, C.: *The timeles Way of Building*. New York: Oxford University Press, 1979

[AlPo99] ALTMANN, J.; POMBERGER, G.: *Kooperative Softwareentwicklung: Konzepte, Modell und Werkzeuge*. In: SCHEER, A.-W. (Hrsg.); NÜTTGENS, M. (Hrsg.): *Electronic Business Engineering: 4. Internationale Tagung Wirtschaftsinformatik 1999*. Heidelberg: Physica-Verlag, 1999, S. 643–664

[Álv+01] ÁLVAREZ, J. M.; EVANS, A.; SAMMUT, P.: *Mapping between Levels in the Metamodel Architecture*. In: [GoKo01], S. 34–46

[And+91] ANDERSEN, R. (Hrsg.); BUBENKO JR., J. A. (Hrsg.); SOLVBERG, A. (Hrsg.): *Advanced Information Systems Engineering - Third International Conference CAiSE '91, Trondheim Norway*. Lecture Notes in Computer Science 498, Berlin et al.: Springer, 1991

[ApLu95] APPELRATH, H.-J.; LUDEWIG, J.: *Skriptum Informatik: eine konventionelle Einführung*. Stuttgart: Teubner, 1995

[Appl00] APPLETON, B.: *The ACME Project: SCM Definitions*. `http://www.enteract.com/~bradapp/acme/scm-defs.html`, Download: 22.08.2000, 2000

[Ask+99] ASKLUND, U.; BENDIX, L.; CHRISTENSEN, H. B.; MAGNUSSON, B.: *The Unified Extensional Versioning Model*. In: ESTUBLIER, J. (Hrsg.): *System Configuration Management: 9th International Symposium, SCM-9, Toulouse, France, September 1999, Proceedings*. Berlin; Heidelberg: Springer, 1999 Lecture Notes in Computer Science 1675, S. 100–122

[AtKü01] ATKINSON, C.; KÜHNE, T.: *The Essence of Multilevel Metamodeling*. In: [GoKo01], S. 19–33

[AtTo95] ATZENI, P.; TORLONE, R.: *Schema translation between heterogeneous data models in a lattice framework*. In: *Sixth IFIP TC-2 Working Conference on Data Semantics (DS-6)*, 1995, S. 345–364

[Balz92] BALZERT, H.: *Die Entwicklung von Software-Systemen: Prinzipien, Methoden, Sprachen, Werkzeuge*. Mannheim et al.: BI-Wissenschaftsverlag, 1992

[Balz96] BALZERT, H.: *Lehrbuch der Software-Technik: Software-Entwicklung*. Heidelberg, Berlin: Spektrum Akademischer Verlag, 1996

[Balz98] BALZERT, H.: *Lehrbuch der Software-Technik: Software-Management, Software-Qualitätssicherung, Unternehmensmodellierung*. Heidelberg, Berlin: Spektrum Akademischer Verlag, 1998

[Balz99] BALZERT, H.: *Lehrbuch der Objektmodellierung: Analyse und Entwurf*. Heidelberg, Berlin: Spektrum Akademischer Verlag, 1999

[Bec+95] BECKER, J.; ROSEMANN, M.; SCHÜTTE, R.: *Grundsätze ordnungsmäßiger Modellierung*. In: Wirtschaftsinformatik, 37 (1995) 5, S. 435–445

[Bec+97] BECKER, J. (Hrsg.); ROSEMANN, M. (Hrsg.); SCHÜTTE, R. (Hrsg.): *Entwicklungsstand und Entwicklungsperspektiven der Referenzmodellierung*. Münster: Institut für Wirtschaftsinformatik der Westfälischen Wilhelms-Universität Münster, 1997

[Bec+99a] BECKER, J. (Hrsg.); ROSEMANN, M. (Hrsg.); SCHÜTTE, S. (Hrsg.): *Referenzmodellierung: state-of-the-art und Entwicklungsperspektiven*. Heidelberg: Physica, 1999

[Bec+99b] BECKER, J. (Hrsg.); SCHÜTTE, R. (Hrsg.); KÖNIG, W. (Hrsg.); WENDT, O. (Hrsg.); ZELEWSKI, S. (Hrsg.): *Wirtschaftsinformatik und Wissenschaftstheorie: Bestandsaufnahme und Perspektiven*. Wiesbaden: Gabler, 1999

[Beer59] BEER, S.: *Kybernetik und Management*. Frankfurt am Main: Fischer-Verlag, 1959

[Ben+99] BENDECK, F.; GOLDMANN, S.; HOLZ, H.; KÖTTING, B.: *Coordinating Management Activities in Distributed Software Development Projects*. `http://wwwagse.informatik.uni-kl.de/Projects/SFB_Subprojects/A2/Documents/coordinat.ps`, Download: 15.09.2000, 1999

[BeSc97] BECKER, J.; SCHÜTTE, R.: *Referenz-Informationsmodelle für den Handel: Begriff, Nutzen und Empfehlungen für die Gestaltung und unternehmensspezifische Adaption von Referenzmodellen*. In: KRALLMANN, H. (Hrsg.): *Internationale Geschäftsfähigkeit auf der Basis flexibler Organisationsstrukturen und leistungsfähiger Informationssysteme*, 1997, S. 427–447

[BOC 02] BOC GMBH (Hrsg.): *BOC Information Technologies Consulting GmbH*. `http://www.boc-eu.com/german/`, Download: 03.04.2002, 2002

[Boeh88] BOEHM, B. W.: *A Spiral Model of Software Development and Enhancement*. In: IEEE Computer, 21 (1988) 5, S. 61–72

[Booc91] BOOCH, G.: *Object Oriented Design with Aplications*. Redwood City, California: The Benjamin/Cummings Publishing Company, Inc., 1991

[Bra+99] BRANDT, P.; DETTMER, D.; DIETRICH, R.-A.; SCHÖN, G.: *Sprachwissenschaft: Ein roter Faden für das Studium*. Böhlau Studienbücher: Grundlagen des Studiums, Köln: Böhlau, 1999

[Bri+96a] BRINKKEMPER, S.; LYYTINEN, K.; WELKE, R. J.: *Editors' Preface*. In: [Bri+96b], S. vjj

[Bri+96b] BRINKKEMPER, S. (Hrsg.); LYYTINEN, K. (Hrsg.); WELKE, R. J. (Hrsg.): *Method Engineering - Principles of construction and tool support, Proceedings of the IFIP TC8, WG8.1/8.2 Working Conference in Atlanta*. London et al.: Chapman and Hall, 1996

[Bri+98] BRINKKEMPER, S.; SAEKI, M.; HARMSEN, F.: *Assembly Techniques for Method Engineering*. In: PERNICI, B. (Hrsg.); THANOS, C. (Hrsg.): *Advanced Information Systems Engineering - 10th International Conference, CAiSE '98*. Berlin et al.: Springer, 1998 Lecture Notes in Computer Science 1413, S. 381–400

[Bri+99] BRINKKEMPER, S.; SAEKI, M.; HARMSEN, F.: *Meta-Modeling based Assembly Techniques for Situational Method Engineering*. In: Information Systems, 24 (1999) 3, S. 209–228

[Bro+91] BROWN, A.; DART, S.; FEILER, P.; WALLNAU, K.: *The State of Automated Configuration Management*. Software Engineering Institute (Carnegie Mellon University), Arbeitsbericht ATR 92, 1991 `ftp://ftp.sei.cm.edu/pub/case-env/config_mgt/papers/atr_cm_state.pdf`, Download: 16.07.2000

[Bro+99] BROWN, W. J.; "SKIP" MCCORMICK III, H. W.; THOMAS, S. W.: *AntiPatterns and Patterns in Software Configuration Management*. New York: Wiley, 1999

[Bus+96] BUSCHMANN, F.; MEUNIER, R.; ROHNERT, H.; SOMMERLAD, P.; STAL, M.: *A System of Patterns*. Chichester et al.: John Wiley & Sons, 1996

[Cede98] CEDERQVIST, P.: *Version Management with CVS*. `http://wwwinfo.cern.ch/asd/cvs/cvs.ps`, Download: 18.11.2000, 1998

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

[Chro92] CHROUST, G.: *Modelle der Software-Entwicklung*. München; Wien: Oldenbourg, 1992

[CoLi95] CONRADI, R.; LIU, C.: *Process Modelling Languages: One or Many?* In: SCHÄFER, W. (Hrsg.): *Software Process Technology: Fourth European Workshop EWSPT '95*. Berlin et al.: Springer, 1995 Lecture Notes in Computer Science 913, S. 98–118

[Cop+00] COPLIEN, J.; BERCZUK, S.; DEVOS, M.; NEILHARRISON: *OrgPatterns instance of ThoughtsWeaver*. `http://www.bell-labs.com/cgi-user/OrgPatterns/OrgPatterns`, Download: 25.10.2000, 2000

[Cor+94] CORDY, J.; ABU-HAMDEH, R.; MARTIN, T.: *Schema translation using structural transformation*. In: *CASCON'94*, IBM Centre for Advanced Studies, 1994, S. 202–215

[CoWe96] CONRADI, R.; WESTFECHTEL, B.: *Version Models for Software Configuration Management*. RWTH Aachen, Forschungsbericht AIB 96-10, 1996

[CoWe97] CONRADI, R.; WESTFECHTEL, B.: *Towards a Uniform Version Model for Software Configuration Management*. In: CONRADI, R. (Hrsg.): *Software Configuration Management: ICSE'97 SCM-7 Workshop, Boston, MA, USA, May 1997*. Berlin et al.: Springer, 1997 Lecture Notes in Computer Science 1235, S. 1–17

[CrÅg01] CRONHOLM, S.; ÅGERFALK, P. J.: *On the Concept of Method in Information Systems Development*. `http://www.ida.ilu.se/~stecr/publik.methconc.pdf`, 2001

[Cro+98] CRONGORAC, L.; RAO, A.; RAMAMOHANARAO, K.: *Classifying Inheritance Mechanisms in Concurrent Object-Oriented Programming*. In: JUL, E. (Hrsg.): *ECOOP'98 - Object-Oriented-Programming*. Lecture Notes in Computer Science 1445, Berlin et al.: Springer, 1998, S. 571–601

[DaRa97] DAHME, C.; RAEITHEL, A.: *Ein tätigkeitstheoretischer Ansatz zur Entwicklung von brauchbarer Software*. In: Informatik-Spektrum, 20 (1997) 1, S. 5–12

[Dart90] DART, S.: *Spectrum of Functionality in Configuration Management Systems*. Software Engineering Institute (Carnegie Mellon University), Forschungsbericht CMU/SEI-90-TR-11, Pittsburgh, 1990 `ftp:`

//ftp.sei.cmu.edu/pub/case-env/config_mgt/tech_rep/cm_spectrum_of_func_TR11_90.pdf, Download: 19.10.2000

[Dart91] DART, S.: *Concepts in Configuration Management Systems*. In: *SCM 1991: Proceedings of the 3rd International Workshop on Software Configuration Management*. Trondheim, Norway: ACM Press, 1991, S. 1–18

[Dart92] DART, S.: *The Past, Present and Future of Configuration Management*. Software Engineering Institute (Carnegie Mellon University), Forschungsbericht CMU/SEI-92-TR-8, Pittsburgh, 1992 ftp://ftp.sei.cmu.edu/pub/case-env/config_mgt/tech_rep/cm_past_pres_future_TR08_92.pdf, Download: 13.08.2000

[DeMa79] DEMARCO, T.: *Structured analysis and system specification*. Englewood Cliffs: Prentice-Hall, 1979

[Deut80] DEUTSCHES INSTITUT FÜR NORMUNG E.V. (NORMENAUSSCHUSS TERMINOLOGIE): *DIN 2331: Begriffssysteme und ihre Darstellung*. 1980

[Deut93] DEUTSCHES INSTITUT FÜR NORMUNG E.V. (NORMENAUSSCHUSS TERMINOLOGIE): *DIN 2330: Begriffe und Benennungen; Allgemeine Grundsätze*. 1993

[Deut94] DEUTSCHES INSTITUT FÜR NORMUNG E.V. (NORMENAUSSCHUSS QUALITÄTSMANAGEMENT, STATISTIK UND ZERTIFIZIERUNGSGRUNDLAGEN): *DIN EN ISO 9000-1: Normen zum Qualitätsmanagement und zur Qualitätssicherung/QM-Darlegung - Teil 1: Leitfaden für die Anwendung (ISO 9000-1:1994)*. 1994

[Deut96] DEUTSCHES INSTITUT FÜR NORMUNG E.V.: *DIN EN ISO 10007 - Qualitätsmanagement: Leitfaden für Konfigurationsmanagement (ISO10007:1995)*. 1996

[Deut98] DEUTSCHES INSTITUT FÜR NORMUNG E.V. (NORMENAUSSCHUSS QUALITÄTSMANAGEMENT, STATISTIK UND ZERTIFIZIERUNGSGRUNDLAGEN): *DIN EN ISO 9000-3: Normen zum Qualitätsmanagement und zur Qualitätssicherung/QM-Darlegung - Teil 3: Leitfaden für die Anwendung von ISO 9001:1994 auf Entwicklung, Lieferung, Installation und Wartung von Computer-Software (ISO 9000-3:1997)*. 1998

[Deut00] DEUTSCHES INSTITUT FÜR NORMUNG E.V. (NORMENAUSSCHUSS QUALITÄTSMANAGEMENT, STATISTIK UND ZERTIFIZIERUNGSGRUNDLAGEN): *DIN EN ISO 9001: Qualitätsmanagement - Forderungen (ISO/DIS 9001:1999)*. 2000

[Diet02] DIETZSCH, A.: *Systematische Wiederverwendung in der Software-Entwicklung*. Wiesbaden: Deutscher Universitäts-Verlag, 2002

[Dink73] DINKELBACH, W.: *Modell - ein isomorphes Abbild der Wirklichkeit?* In: GROCHLA, E. (Hrsg.); SZYPERSKI, N. (Hrsg.): *Modell- und computergestützte Unternehmensplanung*. Wiesbaden: Gabler, 1973, S. 151–162

[Dit+01] DITTRICH, K. R. (Hrsg.); GEPPERT, A. (Hrsg.); NORRIE, M. C. (Hrsg.): *Advanced Information Systems Engineering: Proceedings of the 13th International Conference, CAiSE 2001, Interlaken, Switzerland*. Lecture Notes in Computer Science 2068, Berlin et al.: Springer, 2001

[Dres99] DRESBACH, S.: *Epistemologische Überlegungen zu Modellen in der Wirtschaftsinformatik*. In: [Bec+99b], S. 71–94

[East94] EASTERBROOK, S.: *Co-ordinating Distributed ViewPoints: the anatomy of a consistency check*. Department of Computing, Imperial College, SW7 2BZ, Forschungsbericht, London, 1994

[Ebe+98] EBERT, J.; SÜTTENBACH, R.; UHE, I.: *Meta-CASE Worldwide*. Universität Koblenz-Landau, Institut für Softwaretechnik, Projekt JKOGGE, Forschungsbericht, 1998

[Ebe+99] EBERT, J.; KULLBACH, B.; WINTER, A.: *GraX - An Interchange Format for Reengineering Tools*. Universität Koblenz-Landau, Institut für Softwaretechnik, Projekt JKOGGE, Forschungsbericht, 1999

[Eco77] ECO, U.: *Zeichen: Einführung in einen Begriff*. Frankfurt am Main: Suhrkamp, 1977

[EsCa95] ESTUBLIER, J.; CASALLAS, R.: *Three Dimensional Versioning*. In: [Estu95], S. 118–135

[Estu95] ESTUBLIER, J. (Hrsg.): *Software Configuration Management: ICSE SCM-4 and SCM-5 Workshops - selected papers*. Lecture Notes in Computer Science 1005, Berlin et al.: Springer, 1995

[Estu00] ESTUBLIER, J.: *Software Configuration Management: A Roadmap*. `http://www.cs.acl.ac.uk/staff/A.Finkelstein/fose/finalestublier.pdf`, Download: 16.07.2000, 2000

[Ever02] EVERSMANN, L.: *Die Wirtschaftsinformatik als Wissenschaft und ihre Erkenntnisziele (Replik zu [Hei+01])*. In: Wirtschaftsinformatik, 44 (2002) 1, S. 91–98

[Fahr97] FAHRNER, C.: *Schematransformation in Datenbanken*. In: *Dissertationen zu Datenbanken und Informationssystemen*. 1997

[FeSi01] FERSTL, O. K.; SINZ, E. J.: *Grundlagen der Wirtschaftsinformatik*. 4. Auflage, München: Oldenbourg, 2001

[Fin+93] FINKELSTEIN, A.; GABBAY, D.; HUNTER, A.; KRAMER, J.; NUSEIBEH, B.: *Inconsistency Handling in Multi-Perspective Specifications*. In: SOMMERVILLE, I. (Hrsg.); PAUL, M. (Hrsg.): *Proceedings of the Fourth-European Software Engineering Conference (ESEC '93), Garmisch-Partenkirchen, Germany, September 1993*. Berlin; Heidelberg: Springer, 1993, S. 84–99

[Flo+97] FLOYD, C.; KRABBEL, A.; RATUSKI, S.; WETZEL, I.: *Zur Evolution der evolutionären Systementwicklung: Erfahrungen aus einem Krankenhausprojekt*. In: Informatik-Spektrum, 20 (1997), Februar 1, S. 13–20

[Fowl02] FOWLER, M.: *The New Methodology*. `http://martinfowler.com/articles/newMethodology.html`, Download 08.10.2002, 2002

[Fran94] FRANK, U.: *Multiperspektivische Unternehmensmodellierung: Theoretischer Hintergrund und Entwurf einer objektorientierten Entwicklungsumgebung*. München et al.: Carl Hanser, 1994

[Fran98a] FRANK, U.: *Evaluating Modelling Languages: Relevant Issues, Epistemological Challanges and a Prelimary Research Framework*. In: FRANK, U. (Hrsg.); HAMPE, F. (Hrsg.): *Arbeitsberichte des Instituts für Wirtschaftsinformatik Nr. 15*. Koblenz: Institut für Wirtschaftsinformatik der Universität Koblenz-Landau, 1998

[Fran98b] FRANK, U.: *The MEMO Meta-Metamodel*. In: FRANK, U. (Hrsg.); HAMPE, F. (Hrsg.): *Arbeitsberichte des Instituts für Wirtschaftsinformatik Nr. 9*. Koblenz: Institut für Wirtschaftsinformatik der Universität Koblenz-Landau, 1998

[Fran98c] FRANK, U.: *The MEMO Object Modelling Language (MEMO-OML)*. In: FRANK, U. (Hrsg.); HAMPE, F. (Hrsg.): *Arbeitsberichte des Instituts für Wirtschaftsinformatik Nr. 10*. Koblenz: Institut für Wirtschaftsinformatik der Universität Koblenz-Landau, 1998

[Fran02] FRANK, U.: *MEMO Center*. `http://www.uni-koblenz.de/~iwi/UM/MEMO/MEMOCenter.html`, Download: 10.07.2002, 2002

[Gait79] GAITANIDES, M.: *Konstruktion von Entscheidungsmodellen und 'Fehler dritter Art'*. In: Wirtschaftswissenschaftliches Studium, 8 (1979) 1, S. 8–12

[Gait83] GAITANIDES, M.: *Prozessorganisation: Entwicklung, Ansätze und Programme prozessorientierter Organsiationsgestaltung*. München: Vahlen-Verlag, 1983

[Garv84] GARVIN, D. A.: *What does Product Quality Really Mean?* In: Sloan Management Review, 26 (1984) 1, S. 25–43

[Gese01] GESELLSCHAFT FÜR INFORMATIK, ARBEITSKREISE BEGRIFFSBILDUNG: *Informatik-Begriffsnetz*. `http://www.fast.de/fg511/giak/`, Download: 21.01.2001, 2001

[Gies98] GIESEL, J.: *Software-Konfigurationsmanagement*. In: [Hei+98], S. 27–41

[GoKo01] GOGOLA, M. (Hrsg.); KOBRYN, C. (Hrsg.): *UML 2001 - The Unified Modeling Language: Modeling Languages, Concepts and Tools*. Lecture Notes in Computer Science 2185, Berlin et al.: Springer, 2001

[Gol+97] GOLDKUHL, G.; LIND, M.; SEIGERROTH, U.: *Method integration as learning process*. In: *The 5th Annual Conference on Methodologies Training and education of methodologoy practitioners and researchers*. Preston: The British Computer Society, 1997

[Gra+97] GRAHAM, I.; HENDERSON-SELLERS, B.; YOUNESSI, H.: *The OPEN process specification*. Harlow et al.: Addison Wesley, 1997

[Grei02] GREIFFENBERG, S.: *Generischer Modelleditor 2001*. `http://wise.wiwi.tu-dresden.de/gme`, Download 15.08.2002, 2002

[GrRo00] GREEN, P.; ROSEMANN, M.: *Integrated Process Modeling: An ontological overview*. In: Information Systems, 25 (2000) 2, S. 73–87

[GuPr01] GUPTA, D.; PRAKASH, N.: *Engineering Methods from Requirments Specifications*. In: Requirments Engineering, 6 (2001) 3, S. 135–160

[Habe93] HABERMANN, H.-J.: *Repository: Eine Einführung*. Handbuch der Informatik 8.1, München: Oldenbourg, 1993

[HaCh93] HAMMER, M.; CHAMPY, J.: *Reengineering the Corporation*. New York: Harper Business, 1993

[Har+94] HARMSEN, F.; BIRKKEMPER, S.; OEI, H.: *Situational Method Engineering for Information System Projects*. In: OLLE, T. W. (Hrsg.); VERRIJN-STUART, A. A.

(Hrsg.): *Methods and Associated Tools for the Information Systems Life Cycle, Proceedings of the IFIP WG8.1 Working Conference CRIS 94,*. Amsterdam: Elsevier Science B. V., 1994, S. 169–194

[Har+96] HARMSEN, F.; BRINKKEMPER, S.; OEI, H.: *Situational Method Engineering for Information System Project Approaches*. In: [Bri+96b], S. 169–194

[Harm97] HARMSEN, A. F.: *Situational Method Engineering*. Utrecht, Moret Ernst & Young management Consultants, Dissertation, 1997

[Hars94] HARS, A.: *Referenzdatenmodelle: Grundlage effizienter Datenmodellierung*. Schriften zur EDV-orientierten Betriebswirtschaft, Wiesbaden: Gabler, 1994

[HeBr96] HESS, T.; BRECHT, L.: *State of the Art des Business Process Redesign*. 2. Auflage, Wiesbaden: Gabler, 1996

[HeBu97] HENDERSON-SELLERS, B.; BULTHUIS, A.: *Object-Oriented Metamethods*. New York et al.: Springer, 1997

[Hei+98] HEILMANN, H. (Hrsg.); KATZSCH, R. M. (Hrsg.); MEIER, A. (Hrsg.); MEINHARDT, S. (Hrsg.); MÖRIKE, M. (Hrsg.); SAUERBURGER, H. (Hrsg.): *Configuration- and Change-Management*. HMD - Praxis der Wirtschaftsinformatik 202, Heidelberg: Hüthig, 1998

[Hei+01] HEINZL, A.; KÖNIG, W.; HACK, J.: *Erkenntnisziele der Wirtschaftsinformatik in den nächsten drei und zehn Jahren*. In: Wirtschaftsinformatik, 43 (2001) 3, S. 223–233

[Hein98] HEINATZ, G.: *Kommunikationsmuster und deren Anwendung in der Informationslandschaft*, TU Dresden, Fakultät Wirtschaftswissenschaften, Dissertation, 1998

[Her+00] HERZWURM, G.; SCHOCKERT, S.; MELLIS, W.: *Joint Requirements Engineering - QFD for Rapid Customer-Focused Software and Internet-Development*. Braunschweig, Wiesbaden: Vieweg, 2000

[Herr91] HERRMANN, H.-J.: *Modellgestützte Planung im Unternehmen: Entwicklung eines Rahmenkonzeptes*. Neue betriebswirtschaftliche Forschung 89, Wiesbaden: Gabler, 1991

[Herz00] HERZWURM, G.: *Kundenorientierte Softwareproduktentwicklung*. Stuttgart et al.: Teubner, 2000

[Herz01] HERZWURM, G.; VEREIN ZUR FÖRDERUNG DER QFD-METHODE QFD INSTITUT DEUTSCHLAND E. V. (Hrsg.): *QFD Institut Deutschland e. V.* `http://www.qfd-id.de/`, Download: 09.04.2001, 2001

[Hes+92] HESSE, W.; MERBETH, G.; FRÖLICH, R.: *Software-Entwicklung: Vorgehensmodelle, Projektführung, Produktverwaltung*. München et al.: Oldenbourg, 1992

[Heym93] HEYM, M.: *Methoden-Engineering: Spezifikation und Integration von Entwicklungsmethoden für Informationssysteme*, Hochschule St. Gallen für Wirtschafts-, Rechts- und Sozialwissenschaften, Dissertation, 1993

[High00] HIGHSMITH, J.: *Extreme Programming*. `http://www.cutter.com/ead/ead0002.html`, Download: 05.03.2001, 2000

[Hofm00] HOFMANN, H. F.: *Requirements Engineering: a situated discovery process*. Wiesbaden et al.: Gabler, 2000

[Hrus98] HRUSCHKA, P.: *Ein pragmatisches Vorgehensmodell für die UML*. In: OBJEKTspektrum, (1998) 2, S. 34–44

[IiHu01] IIVARI, J.; HUISMAN, M.: *The Relationship Between Organisational Culture and the Deployment of System Development Methodologies*. In: [Dit+01], S. 234–250

[Info97] INFORMATION SYSTEM DEVELOPMENT SUPPORT (ISDS) TEAM: *ISDS Product Assurance: Configuration Management Plan*. `http://www-isds.jpl.nasa.gov/cm/html/cm/cmp/isdscmp.htm`, Download: 14.09.2000, 1997

[Ioss80] IOSSIPHIDIS, J.: *Translator to convert the DDL of ERM to the DDL of System 2000*. In: CHEN, P. P.-S. (Hrsg.): *International Conference on Entity-Relationship Approach to Systems Analysis and Design*, North-Holland Publishing Company, 1980, S. 477–504

[Jab+97] JABLONSKI, S.; BÖHM, M.; SCHULZE, W.: *Workflow-Management: Entwicklung Von Anwendungen und Systemen; Facetten Einer Neuen Technologie*. Heidelberg: dpunkt, 1997

[Jäge82] JÄGER, P. K.: *Modellmethodologie und optimale Bestellmenge - Grundriß einer Methodologie der Modellkonstruktion am Modell der optimalen Bestellmenge*, Johann Wolfgang Goethe Universität Frankfurt am Main, Dissertation, 1982

[Jeff00] JEFFRIES, R. ET AL.: *What is eXtreme Programming?* `http://www.xprogramming.com/what_is_xp.htm`, Download: 05.03.2001, 2000

[Jun+00] JUNGINGER, S.; KÜHN, H.; STROBL, R.; KARAGIANNIS, D.: *Ein Geschäftsprozessmanagement-Werkzeug der nächsten Generation - ADONIS: Konzeption und Anwendung*. In: Wirtschaftsinformatik, 42 (2000) 5, S. 392–401

[Kazm98] KAZMEIER, J.: *Modellierung soziotechnischer Systeme im Requirements Engineering bei betrieblicher Software*, Technische Universität München, Institut für Informatik, Dissertation, 1998

[KiKu83] KIESER, A.; KUBICEK, H.: *Organisation*. 2. Auflage, Berlin et al.: de Gruyter, 1983

[Knob74] KNOBLICH, H.: *Die typologische Methode in der Betriebswirtschaftslehre*. In: Wirtschaftswissenschaftliches Studium (WiSt), 1 (1974) 4, S. 141–147

[Knut95] KNUTH, B. E.: *Das Rollenkonzept im Vergleich zu anderen Strategien der Komplexitätsreduktion und die Anwendung bei Methoden der Systementwicklung*. Schriften zur Wirtschaftsinformatik 3, Frankfurt am Main et al.: Lang, 1995

[Kosi62] KOSIOL, E.: *Organisation der Unternehmung*. Wiesbaden: Gabler, 1962

[Kral94] KRALLMANN, H.: *Systemanalyse im Unternehmen*. München; Wien: Oldenbourg, 1994

[Kuhn99] KUHN, T. S.: *Die Struktur wissenschaftlicher Revolutionen*. 15. Auflage, Frankfurt am Main: Suhrkamp, 1999

[KüJu99] KÜHN, H.; JUNGINGER, S.: *An approach to use UML for Business Process Modeling and Simulation in ADONIS*. In: *Proceedings of the 13th European Simulation Multiconference (ESM 99)*, 1999

[Lan+98] LANGNER, P.; SCHNEIDER, C.; JOACHIM: *Petri Net based Certification of Event-Driven Process Chains*. In: DESEL, J. (Hrsg.); SILVA, M. (Hrsg.): *Application and Theory of Petri Nets: 19th International Conference, ICATPN'98, Lisbon, Portugal, June 22-26, 1998, Proceedings*. Berlin et al.: Springer, 1998 Lecture Notes in Computer Science 1420, S. 286–305

[Lang97] LANG, K.: *Gestaltung von Geschäftsprozessen mit Referenzprozeßbausteinen*. Wiesbaden: Deutscher Universitäts Verlag (Gabler Edition), 1997

[Leh+91] LEHNER, F.; AUER-RIZZI, W.; BAUER, R.; BREIT, K.; LEHNER, J.; REBER, G.: *Organisationslehre für Wirtschaftsinformatiker*. München, Wien: Carl Hanser, 1991

[Lehm98] LEHMANN, F. R.: *Normsprache*. In: Informatik-Spektrum, 21 (1998) 6, S. 366–367

[LiRe94] LIN, Y.-J.; REISS, S. P.: *Configuration Management in Terms of Modules*. In: [Estu95], S. 101–117

[LiSu89] LIEBELT, W.; SULZBERGER, M.: *Grundlagen der Ablauforganisation*. Giessen: Dr. Götz Schmidt-Verl., 1989

[Malo99] MALORNY, C.: *TQM umsetzen: Weltklasse neu definieren, Leistungsoffensive einleiten, Business Excellence erreichen*. München et al.: Carl Hanser, 1999

[Mar+96] MARTIIN, P.; HARMSEN, F.; ROSSI, M.: *A Functional Framework for Evaluating Method Engineering Environments: the case of Maestro II/ Decamerone and MetaEdit+*. In: [Bri+96b], S. 63–85

[Mart00] MARTIN, R. C. ET AL.: *Extreme Programming Newsgroup*. `http://groups.yahoo.com/group/extremeprogramming/`, Download: 06.03.2001, 2000

[Mart02] MARTIN, R. C.: *Agile Processes*. `http://www.objectmentor.com/resources/articles/agileProcess.pdf`, Download 09.10.2002, 2002

[Mayr01] MAYR, H.: *Projekt Engineering: Ingenieurmäßige Softwareentwicklung in Projektgruppen*. München et al.: Carl Hanser, 2001

[Meta00] METACASE CONSULTING (Hrsg.): *MetaEdit+ revolutionized the way Nokia develops mobile phone software*. http://www.metacase.com/papers/MetaEdit_in_Nokia.pdf, Download 31.8.2001, 2000

[Micr98] MICROSOFT CORPORATION (Hrsg.): *Microsoft Encarta 98 Enzyklopädie*. CD ROM, 1998

[Moli84] DE MOLIÈRE, F.: *Prinzipien des Modellentwurfs - Eine modelltheoretische und gestaltungsorientierte Betrachtung*, Technische Hochschule Darmstadt, Dissertation, 1984

[Morr72] MORRIS, C. W.: *Grundlagen der Zeichentheorie*. München: Carl Hanser, 1972

[Müll80] MÜLLER-MERBACH, H.: *Modelldenken und der Entwurf von Unternehmensplanungsmodellen für die Unternehmensführung*. In: HAHN, D. (Hrsg.): *Führungsprobleme industrieller Unternehmungen: Festschrift für Friedrich Thomée zum 60. Geburtstag*. Berlin: de Gruyter, 1980, S. 471–489

[NaRa69] NAUR, P. (Hrsg.); RANDELL, B. (Hrsg.): *Software Engineering: A Report on a Conference sponsored by the NATO Science committee*. 1969

[NoSc99] NOACK, J.; SCHIENMANN, B.: *Objektorientierte Vorgehensmodelle im Vergleich*. In: Informatik-Spektrum, 22 (1999) 3, S. 166–180

[Nuse94] NUSEIBEH, B. A.: *A Multi-Perspective Framework for Method Integration*. London, Imperial College of Science, Technology and Medicine, University of London; Department of Computing, Dissertation, 1994

[Obje00a] OBJECT MANAGEMENT GROUP (OMG) (Hrsg.): *Meta Object Facility (MOF) Specification Version 1.3*. `ftp://ftp.omg.org/pub/docs/formal/00-04-03.pdf`, Download: 18.10.2000, 2000

[Obje00b] OBJECT MANAGEMENT GROUP (OMG) (Hrsg.): *Unified Modeling Language Specification Version 1.3*. `ftp://ftp.omg.org/pub/docs/formal/00-03-01.pdf`, Download: 25.10.2000, 2000

[Oess93] OESS, A.: *Total Quality Management: Die ganzheitliche Qualitätsstrategie*. 3. Auflage, Wiesbaden: Gabler, 1993

[Oest01] OESTEREICH, B.: *Objektorientierte Softwareentwicklung: Analyse und Design mit der Unified Modelling Language*. 5. Auflage, München; Wien: Oldenbourg, 2001

[Opda97] OPDAHL, A. L.: *A Comparision of Four Families of Multi-Perspective Problem Analysis Methods*. In: *Proceedings of „Information Systems Research in Scandinavia“, IRIS 20*. 1997

[OpHe99] OPDAHL, A. L.; HENDERSON-SELLERS, B.: *Evaluating and improving OO modelling languages using the BWW-model*. In: *Proceedings of the Information Systems Foundations Workshop: Ontology, Semiotics and Practice*, 1999

[OpHe02] OPDAHL, A. L.; HENDERSON-SELLERS, B.: *Ontological Evaluation of the UML Using the Bunge-Wand-Weber Model*. In: Software and Systems Modeling, 1 (2002) 1, S. 43–67

[Ortn02] ORTNER, E.: *Sprachingenieurwesen - Empfehlung zur inhaltlichen Weiterentwicklung der (Wirtschafts-)Informatik*. In: Informatik-Spektrum, 25 (2002) 1, S. 39–51

[Över00] ÖVERGAARD, G.: *Formal Specification of Object-Oriented Meta-modelling*. In: MAIBAUM, T. (Hrsg.): *Fundamental Approaches to Software Engineering*. Lecture Notes in Computer Science 1783, Berlin et al.: Springer, 2000, S. 193–207

[Part91] PARTSCH, H.; ENDRES, A. (Hrsg.); KRALLMANN, H. (Hrsg.); SCHNUPP, P. (Hrsg.): *Requirements Engineering*. Handbuch der Informatik (Band 5.5), München et al.: Oldenbourg, 1991

[Pau+93] PAULK, M. C.; CURTIS, B.; CHRISSIS, M. B.; WEBER, C. V.: *Capability Maturity Model, Version 1.1*. `http://www.sei.cmu.edu/cmm/cmm.articles.html`, Download: 17.10.2000, 1993

[PeSt01] PERSSON, A.; STIRNA, J.: *Why Enterprise Modelling? An Explorative Study into Current Practice*. In: [Dit+01], S. 465–468

[Pfei00] PFEIFER, W.: *Etymologisches Wörterbuch des Deutschen*. 5. Auflage, München: Deutscher Taschenbuch Verlag, 2000

[PiMa94] PICOT, A.; MAIER, M.: *Ansätze der Informationsmodellierung und ihre betriebswirtschaftliche Bedeutung*. In: Zeitschrift für betriebswirtschaftliche Forschung, 46 (1994) 2, S. 107–126

[PoBl93] POMBERGER, G.; BLASCHEK, G.: *Software-Engineering: Prototyping und objektorientierte Software-Entwicklung*. München et al.: Carl Hanser, 1993

[Popp94] POPPER, K. R.: *Logik der Forschung*. Die Einheit der Gesellschaftswissenschaften, 10. Auflage, Tübingen: J. C. B. Mohr (Paul Siebeck), 1994

[Pres92] PRESSMAN, R. S.: *Software Engineering: A Practitioner's Approach*. 3. Auflage, New York: McGraw-Hill, Inc., 1992

[Quib99] QUIBELDY-CIRKEL, K.: *Entwurfsmuster: Design-Patterns in der objektorientierten Softwaretechnik*. Berlin et al.: Springer, 1999

[Rati00] RATIONAL SOFTWARE CORPORATION (Hrsg.): *Rational Unified Process: Glossary*. `http://www.rational.com/products/rup/index.jsp`, Download 17.10.2000, 2000

[REFA92] REFA VERBAND FÜR ARBEITSSTUDIEN UND BETRIEBSORGANISATION E.V.: *Methodenlehre der Betriebsorganisation; Aufbauorganisation*. München: Carl Hanser, 1992

[Reic95] REICHENBERGER, C.: *VOODOO: A Tool for Orthogonal Version Management*. In: [Estu95], S. 61–79

[Remm97] REMME, M.: *Konstruktion von Geschäftsprozessen: ein modellgestützter Ansatz durch Montage generischer Prozesspartikel*. Wiesbaden: Gabler, 1997

[RoBr95] ROSSI, M.; BRINKKEMPER, S.: *Metrics in Method Engineering*. In: IIVARI, J. (Hrsg.); LYYTINEN, K. (Hrsg.); ROSSI, M. (Hrsg.): *Advanced Information Systems*

*Engineering - 7th International Conference, CAiSe '95*. Berlin et al.: Springer, 1995 Lecture Notes in Computer Science 932, S. 200–216

[RoSc99] ROSEMANN, M.; SCHÜTTE, R.: *Multiperspektivische Referenzmodellierung*. In: BECKER, J. (Hrsg.); ROSEMANN, M. (Hrsg.); SCHÜTTE, R. (Hrsg.): *Referenzmodellierung: State-of-the-Art und Entwicklungsperspektiven*, Physica-Verlag, 1999, S. 22–44

[Rose92] ROSE, T.: *Entscheidungsorientiertes Konfigurationsmanagement*. Informatik-Fachberichte 305, Berlin et al.: Springer, 1992

[Rose96] ROSEMANN, M.: *Komplexitätsmanagement in Prozeßmodellen*. Schriften zur EDV-orientierten Betriebswirtschaft, Wiesbaden: Gabler, 1996

[Royc70] ROYCE, W. W.: *Managing The Development Of Large Software Systems*. In: Proc. IEEE WESCON, (1970), S. 1–9

[Rum+91] RUMBAUGH, J.; BLAHA, M.; PREMERLANI, W.; EDDY, F.; LORENSEN, W.: *Object-Oriented Modeling and Design*. Englewood Cliffs: Prentice Hall, 1991

[Saka80] SAKAI, H.: *A unified approach to the logical design of a hierarchical data model*. In: CHEN, P. P.-S. (Hrsg.): *International Conference on Entity-Relationship Approach to Systems Analysis and Design*, North-Holland Publishing Company, 1980, S. 61–74

[Sayn98] SAYNISCH, M.: *Grundlagen des Konfigurationsmanagements*. In: [Hei+98], S. 7–26

[Sche97] SCHEER, A.-W.: *ARIS House of Business Engineering - Konzept zur Beschreibung und Ausführung von Referenzmodellen*. In: [Bec+97], S. 3–15

[Sche98] SCHEER, A.-W.: *ARIS-Vom Geschäftsprozeß zum Anwendungssystem*. 3. Auflage, Berlin et al.: Springer, 1998

[Sche99] SCHEER, A.-W.: *„ARIS House of Business Engineering": Konzept zur Beschreibung und Ausführung von Referenzmodellen*. In: [Bec+99a], S. 2–21

[Schö94] SCHÖLER, H. R.; KÖPPEN, R. (Hrsg.): *Einführung in die Methode des Quality Functional Deployments*. Gifhorn: Verlag Dr. J. Heizmann, 1994

[Schö99] SCHÖNRICH, G.: *Semiotik zur Einführung*. Hamburg: Julius, 1999

[Schü98] SCHÜTTE, R.: *Grundsätze ordnungsmäßiger Referenzmodellierung: Konstruktion konfigurations- und anpassungsorientierter Modelle*. Neue betriebswirtschaftliche Forschung 233, Wiesbaden: Gabler, 1998

[Schu01] SCHULZE, D.: *Grundlagen der wissensbasierten Konstruktion von Modellen betrieblicher Systeme*. Aachen: Shaker, 2001

[ScRu94] SCHRADER, M.; RUNDSHAGEN, M.: *Objektorientierte Systemanalyse*. Berlin et al.: Springer, 1994

[semt02] SEMTURE GMBH (Hrsg.): *semture GmbH - Projekte*. `http://www.semture.com`, Download: 29.05.2002, 2002

[ShWe49] SHANNON, C. E.; WEAVER, W.: *The Mathematical Theory of Communication*. Urbana: University of Illinois Press, 1949

[Sinz98] SINZ, E. J.: *Modellierung betrieblicher Informationssysteme - Gegenstand, Anforderungen und Lösungsansätze*. In: POHL, K. (Hrsg.); SCHÜRR, A. (Hrsg.); VOSSEN, G. (Hrsg.): *Proceedings Modellierung '98, Angewandte Mathematik und Informatik*, 1998, S. 27–28

[Smo+91] SMOLANDER, K.; LYYTINEN, K.; TAHVANAINEN, V.-P.; MARTTIIN, P.: *MetaEdit - A Flexible Graphical Environment for Methodology Modelling*. In: [And+91], S. 168–193

[Smol91] SMOLANDER, K.: *OPRR: A Model for Modelling Systems Development Methods*. In: *Next Generation CASE Tools*. Amsterdam: IOS Press, 1991

[Somm95] SOMMERVILLE, I.: *Software Engineering*. 5. Auflage, Harlow: Addison-Wesley, 1995

[Stac73] STACHOVIAK, H.: *Allgemeine Modelltheorie*. Wien et al.: Springer, 1973

[StHa97] STAHLKNECHT, P.; HASENKAMP, U.: *Einführung in die Wirtschaftsinformatik*. 9. Auflage, Berlin et al.: Springer, 1997

[Sti+97a] STICKEL, E. (Hrsg.); GROFFMANN, H.-D. (Hrsg.); RAU, K.-H. (Hrsg.): *GABLER-Wirtschaftsinformatik-Lexikon*. Wiesbaden: Gabler, 1997

[Sti+97b] STICKEL, E.; GROFFMANN, H.-D.; RAU, K.-H.: *Gabler-Witschaftsinformatik-Lexikon*. Wiesbaden: Gabler, 1997

[Stra96] STRAHRINGER, S.: *Metamodellierung als Instrument des Methodenvergleichs: Eine Evaluierung am Beispiel objektorientierter Analysemethoden*. Berichte aus der Betriebswirtschaft, Aachen: Shaker, 1996

[Stra99] STRAHRINGER, S.: *Probleme und Gefahren im Umgang mit „Meta"-Begriffen: ein Plädoyer für eine sorgfältige Begriffsbildung*. In: *Konferenzband KnowTechForum '99*. Potsdam, 1999

[Su+92] SU, S. Y. W.; FANG, S. C.; LAM, H.: *An object-oriented rule-based approach to data model and schema translation*. Departament of Computer and Information Sciences, Forschungsbericht TR-92-015, University of Florida, USA, 1992

[Sun 99] SUN MICROSYSTEMS, INC.: *Java Code Conventions*. `ftp://ftp.javasoft.com/docs/codeconv/CodeConventions.pdf`, Download: 19.10.2000, 1999

[Thal98] THALLER, G. E.: *SPICE - ISO 9001 und Software in der Zukunft*. Kaarst: bhv, 1998

[Tolv01] TOLVANEN, J.-P.: *Domänenspezifische Modellierungssprachen für Produktfamilien*. In: OBJEKTspektrum, (2001) 5, S. 17–22

[ToLy93] TOLVANEN, J.-P.; LYYTINEN, K.: *Flexible method adaption in CASE - The Metamodeling Approach*. In: *Scandinavian Journal of Information Systems* 5 (1993) 1, `http://iris.informatik.gu.se/sjis/vol5/tolvanen.shtml`, Download: 23.08.2002, 1993

[Uhe00] UHE, I.: *Symbolspezifikation und -verarbeitung im CeraNet-Prototyp*. Universität Koblenz-Landau, Institut für Softwaretechnik, Projekt JKOGGE, Forschungsbericht, 2000

[US D00] US DEPARTMENT OF TRANSPORTATION: *Surface Search Radar (SSR) Project: Configuration Management Plan (CMP)*. `http://www.hjford.com/cg/ssr/Configuration/CMP.html`, Download: 14.09.2000, 2000

[VaKu99] VAN HILLERGERSBERG, J.; KUMAR, K.: *Using metamodeling to integrate object-oriented analysis, design and programming concepts*. In: Information Systems, 24 (1999) 2, S. 113–129

[Very92] VERYCKEN, L.: *Auf den Spuren der Abstraktion*. In: *Geist der Revolte*. Penzberg: Grundriss, 1992

[Wall90] WALLMÜLLER, E.: *Software-Qualitätssicherung in der Praxis*. München: Carl Hanser, 1990

[Wand89] WAND, Y.: *A Proposal for a Formal Model of Objects*. In: KIM, W. (Hrsg.); LOCHOVSKY, F. H. (Hrsg.): *Object-Oriented Concepts, Databases and Applications*. New York: ACM Press, 1989, Kapitel 21, S. 537–559

[Watz85] WATZLAWIK, P.: *Die erfundenene Wirklichkeit - Was wissen wir was wir zu wissen glauben?* 3. Auflage, München: Piper, 1985

[WaWe95] WAND, Y.; WEBER, R.: *On the deep structure of information systems*. In: Information Systems Journal, (1995) 5, S. 203–223

[Wei+92] WEIDNER, W.; FREITAG, G.; GERNET, E.; ULBRICH, K.: *Organisation in der Unternehmung: Aufbau- und Ablauforganisation; Methoden und Techniken praktischer Organisationsarbeit*. 4. überarb. Auflage, München et al.: Carl Hanser, 1992

[Welk92] WELKE, R. J.: *The CASE Repository: More than another database application*. In: *Challenges and Strategies for Research in Systems Development*. Chichester: Wiley, 1992

[Wern95] WERNER, D.: *Taschenbuch der Informatik*. 2. Auflage, Leipzig: Fachbuchverlag, 1995

[West91] WESTFECHTEL, B.: *Revisions- und Konsistenzkontrolle in einer integrierten Softwareentwicklungsumgebung*. Informatik-Fachberichte 280, Berlin et al.: Springer, 1991

[West99] WESTFECHTEL, B.: *Models and tools for managing development processes*. Lecture Notes in Computer Science 1646, Berlin et al.: Springer, 1999

[Wint00] WINTER, A.: *Ein Referenz-Metaschema der Beschreibungsmittel für Organisationen und Softwaresysteme*, Universität Koblenz-Landau, Dissertation, 2000

[Wiss94] WISSENSCHAFTLICHE KOMMISSION WIRTSCHAFTSINFORMATIK: *Profil der Wirtschaftsinformatik*. In: Wirtschaftsinformatik, 36 (1994) 1, S. 80–81

[Wiss96] WISSENSCHAFTLICHER RAT DER DUDENREDAKTION (HRSG.): *Duden, Rechtschreibung der deutschen Sprache*. 21, Mannheim: Bibliographisches Institut und F. A. Brockhaus AG, 1996

[Wiss00] WISSENSCHAFTLICHER RAT DER DUDENREDAKTION (HRSG.): *Duden, Das große Fremdwörterbuch*. 2. Auflage, Mannheim et al.: Dudenverlag, 2000

[Witt81] WITTGENSTEIN, L.: *Tractatus logico-philosophicus*. London: Routledge & Kegan Paul, 1981

[Wodt97] WODTKE, D.: *Modellbildung und Architektur von Verteilten Workflow-Management-Systemen*. Sankt Augustin: Infix, 1997

[WoKa80] WONG, E.; KATZ, R. H.: *Logical Design and Schema Conversion for relational and DBGT Databases*. In: CHEN, P. P.-S. (Hrsg.): *International Conference on Entity-Relationship Approach to Systems Analysis and Design*, North-Holland Publishing Company, 1980, S. 311–321

[Work98] WORKFLOW MANAGEMENT COALITION (WFMC) (Hrsg.): *The Workflow Reference Model*. `http://www.aiim.org/wfmc/standards/docs/tc003v11.pdf`, Download: 16.03.2001, 1998

[Zele96] ZELEWSKI, S.: *Eignung von Petrinetzen für die Modellierung komplexer Realsysteme - Beurteilungskriterien*. In: Wirtschaftsinformatik, 38 (1996) 4, S. 369–381

[Zell97] ZELLER, A.: *Configuration Management with Version Sets*. Braunschweig, Technische Universität Braunschweig, Dissertation, 22.05.1997

[Zema92] ZEMANEK, H.; ZEIDLER, G. (Hrsg.): *Das geistige Umfeld der Informationstechnik*. Edition SEL-Stiftung, Berlin et al.: Springer, 1992

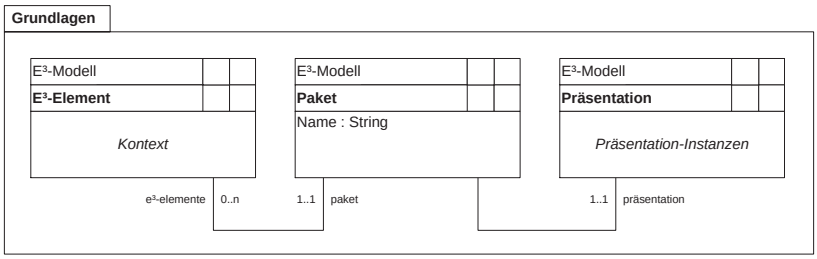
[Zink95] ZINK, K. J.: *TQM als integratives Managementkonzept: Das europäische Qualitätsmodell und seine Umsetzung*. München et al.: Carl Hanser, 1995

# Teil IV
# Anhang

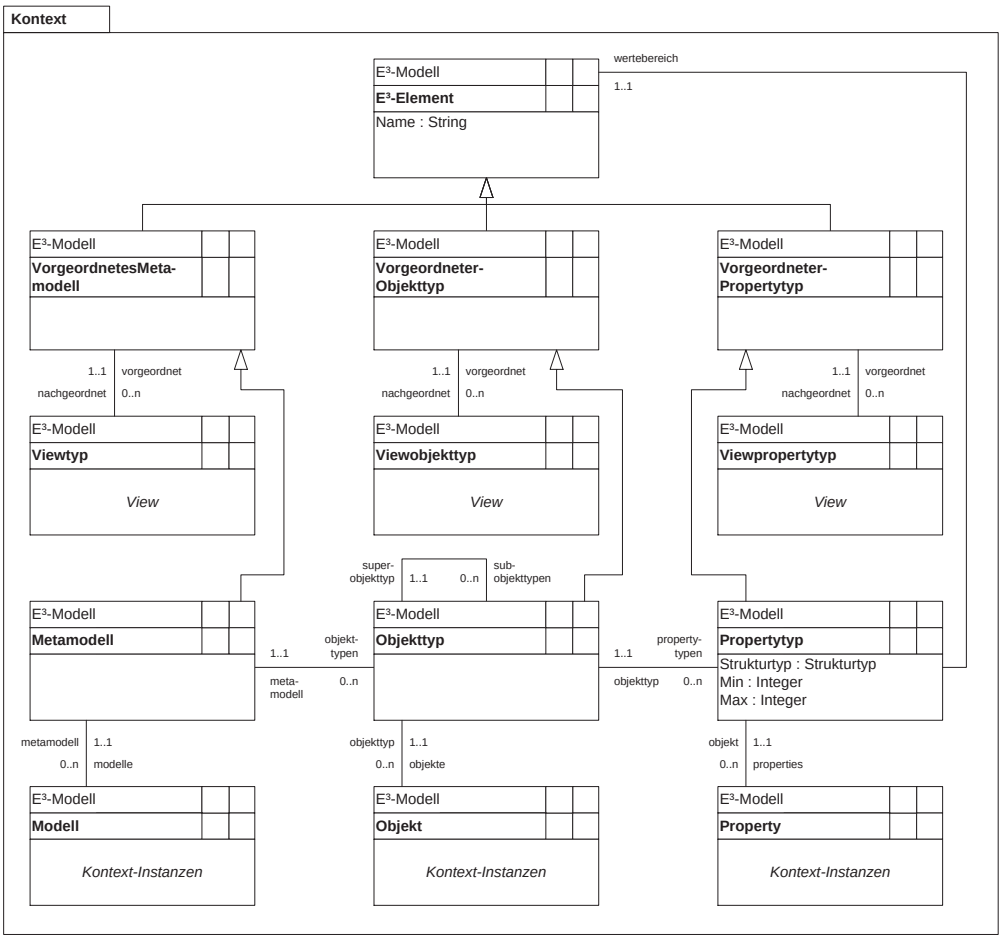
# 11 Spezifikation des E$^3$-Modells

## 11.1 Spezifikation des E$^3$-Modells

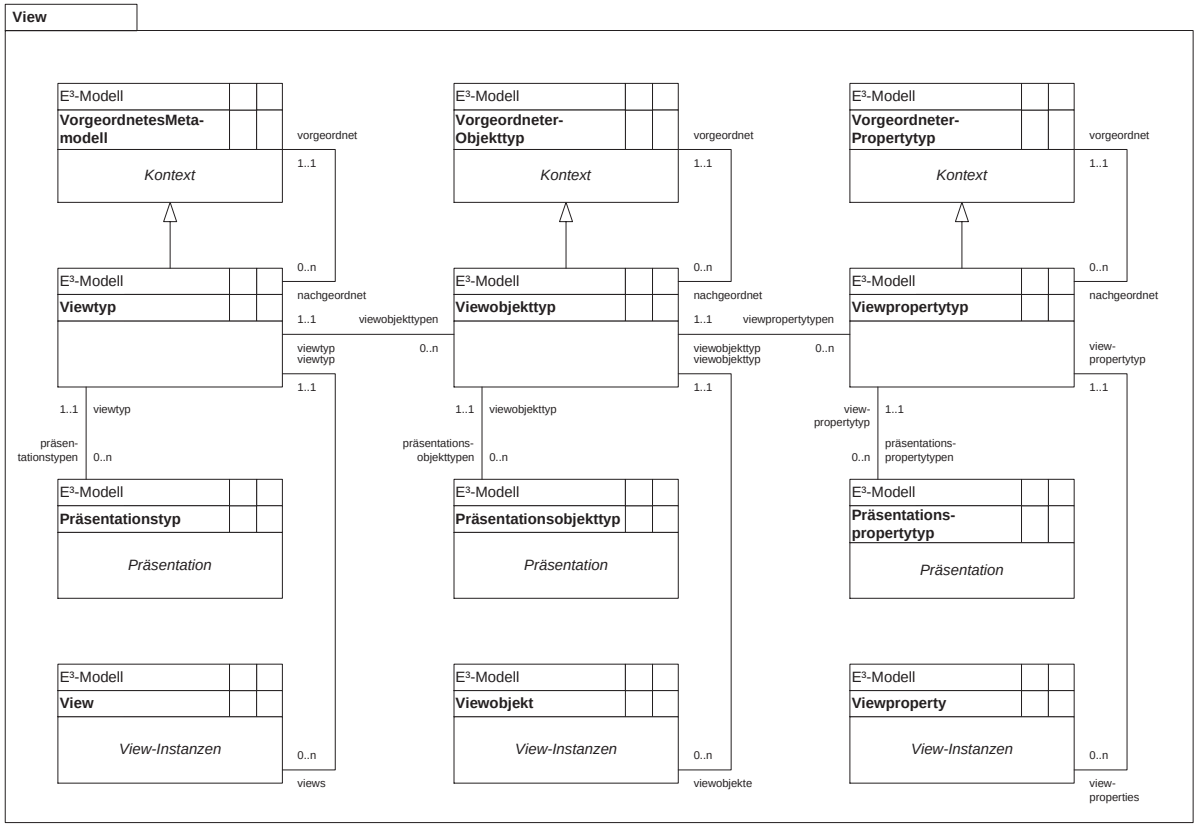
### Grundlagen



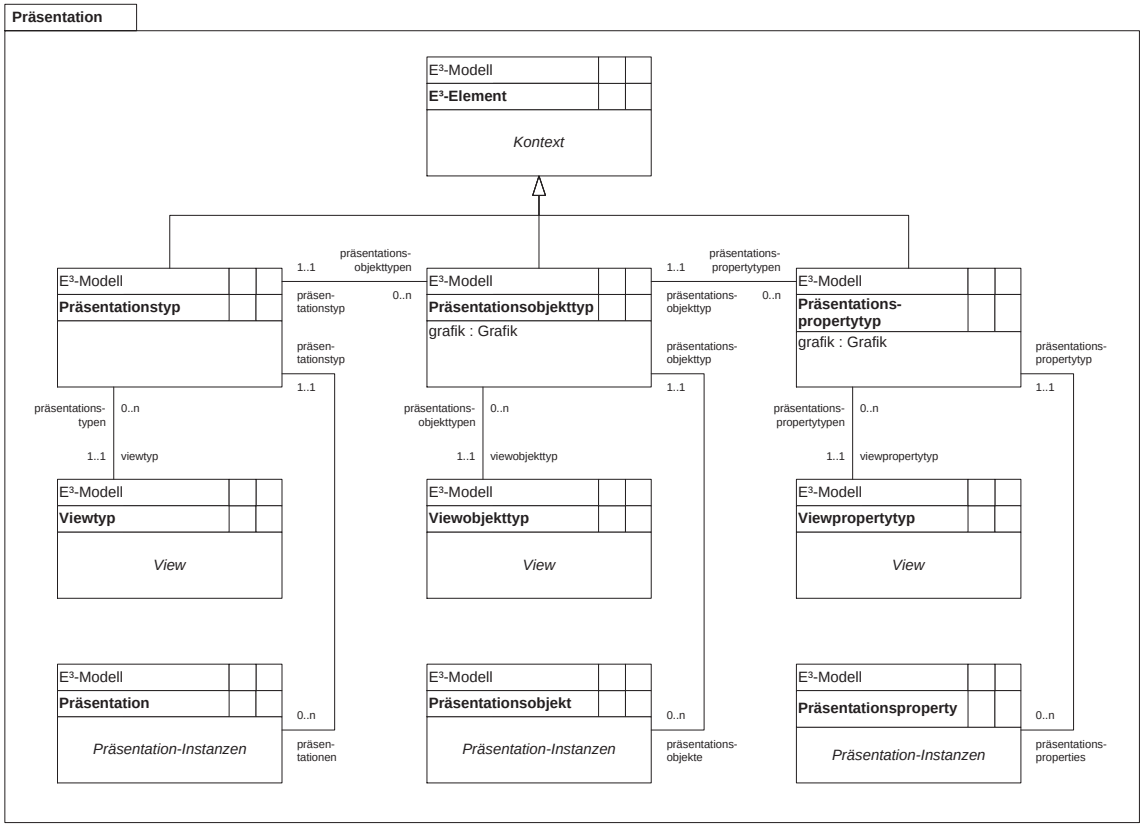

### Typen der Kontextebene



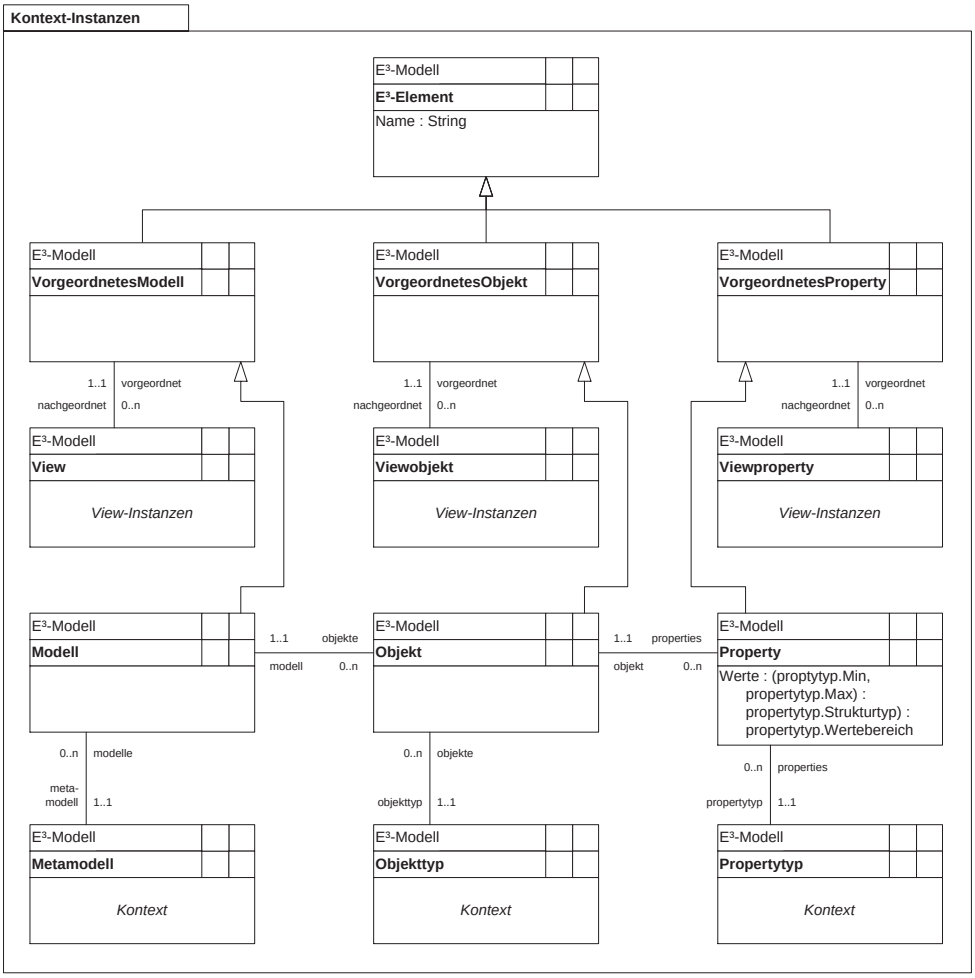
## Typen der Viewebene



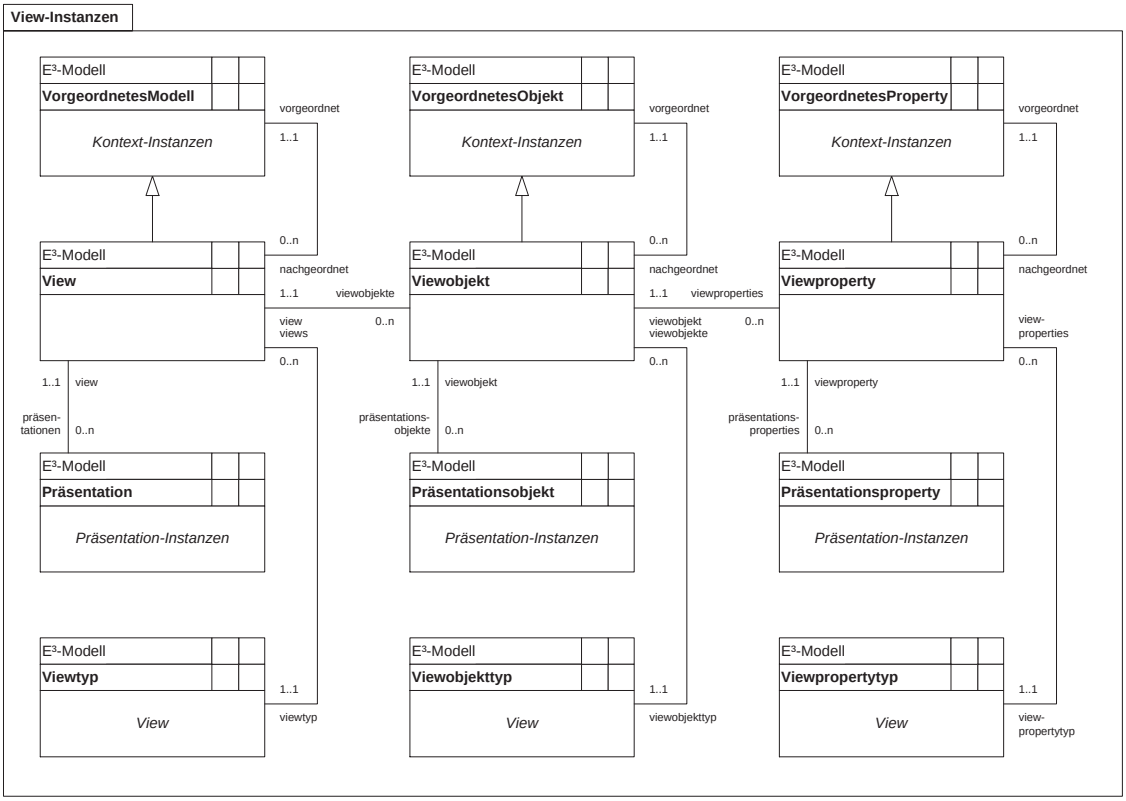
## Typen der Präsentationsebene



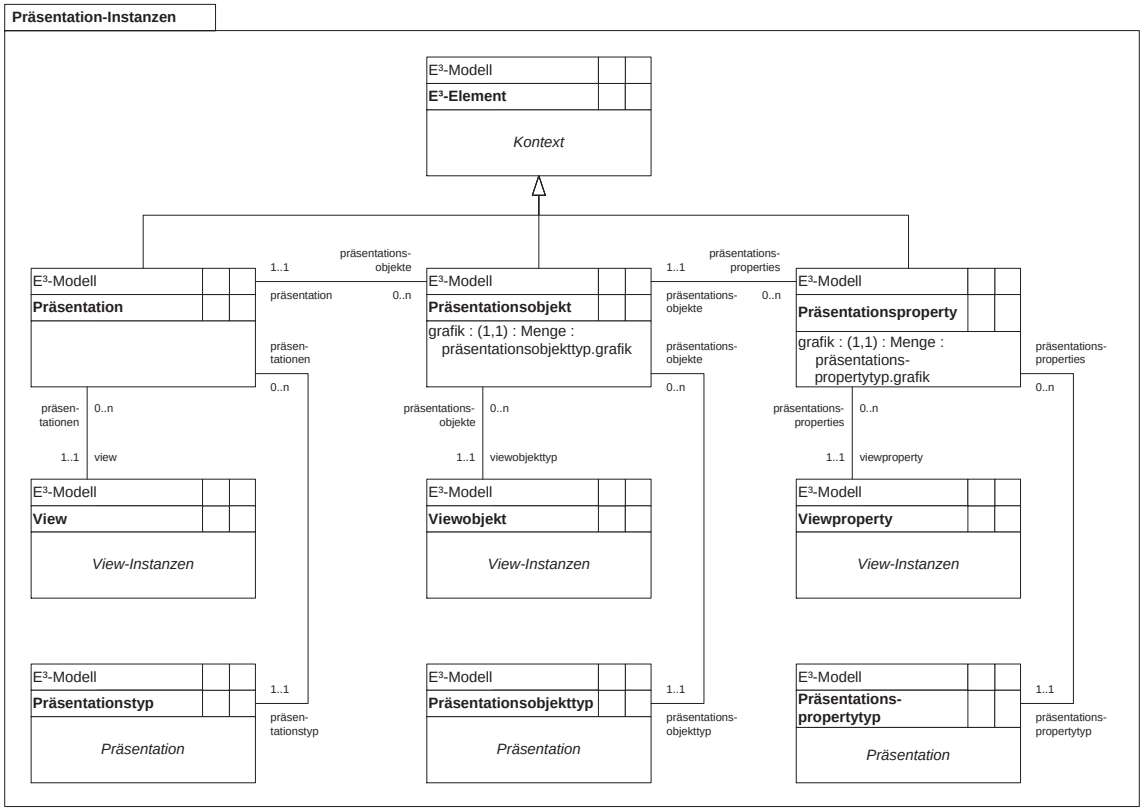
## Instanzen der Kontextebene



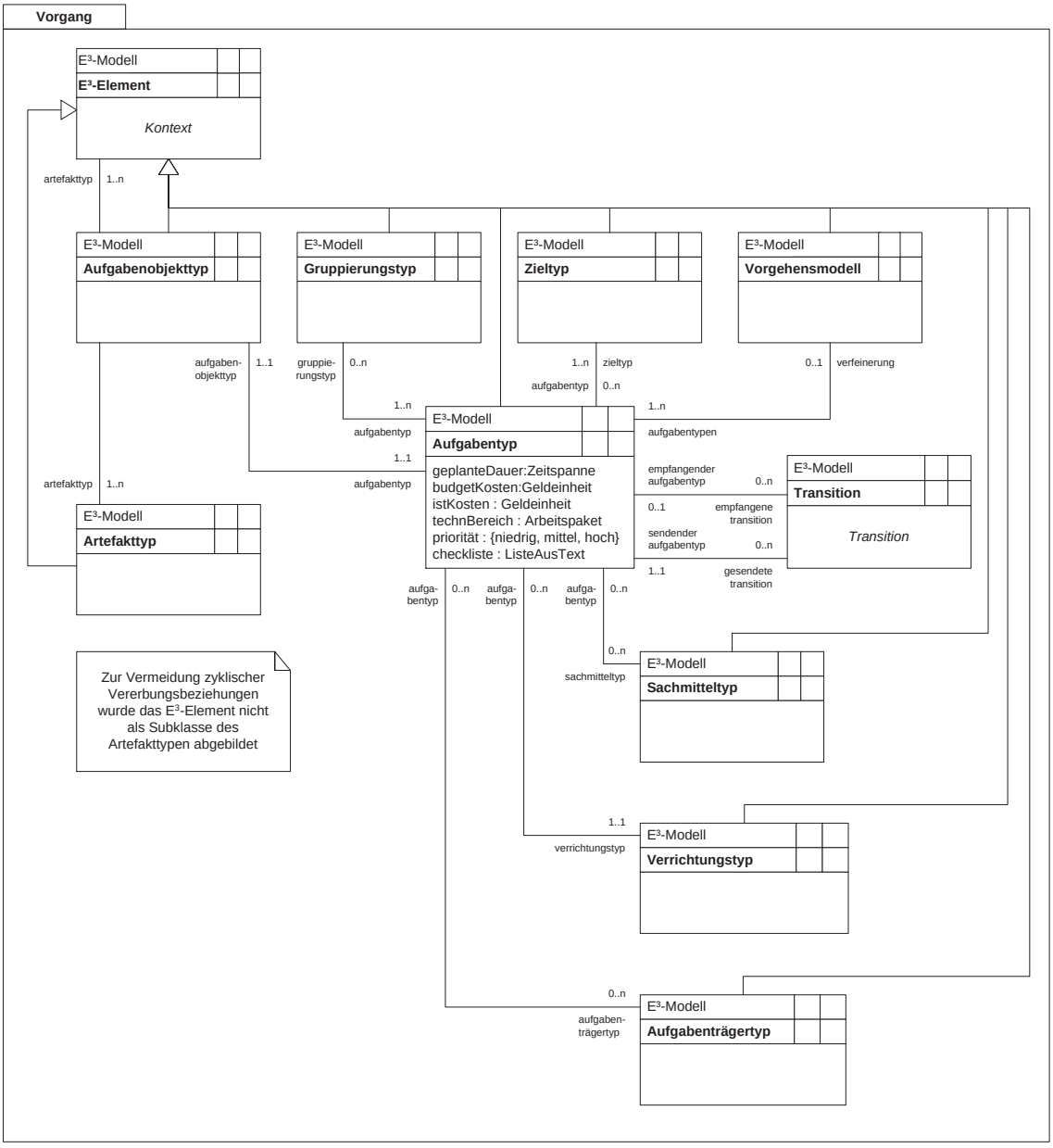
## Instanzen der Viewebene

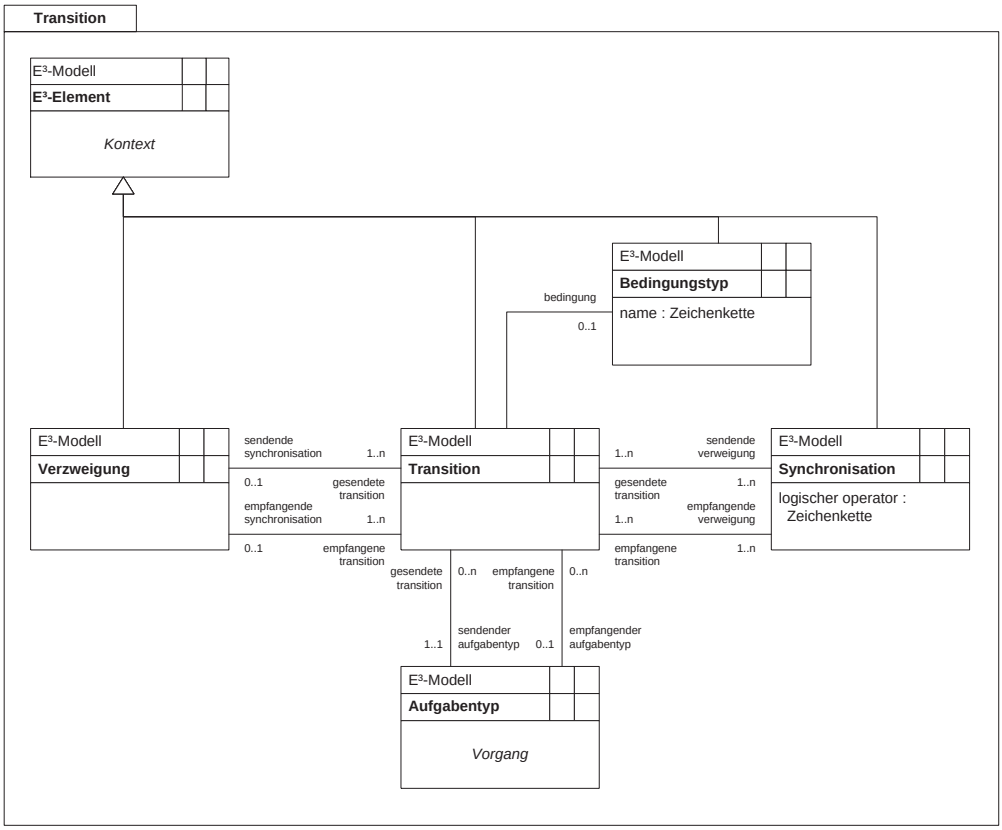


## Instanzen der Präsentationsebene



## Elemente der Vorgangsebene




Transition
E³-Modell
E³-Element
Kontext
E³-Modell
Bedingungstyp
name : Zeichenkette
bedingung
0..1
E³-Modell
Verzweigung
sendende synchronisation
1..n
0..1
gesendete transition
empfangende synchronisation
1..n
0..1
empfangene transition
E³-Modell
Transition
1..n
sendende verweigung
gesendete transition
1..n
1..n
empfangende verweigung
empfangene transition
1..n
E³-Modell
Synchronisation
logischer operator : Zeichenkette
gesendete transition
0..n
empfangene transition
0..n
1..1
sendender aufgabentyp
0..1
empfangender aufgabentyp
E³-Modell
Aufgabentyp
Vorgang

## 11.2 Fachbegriffsmodell des E$^3$-Modells

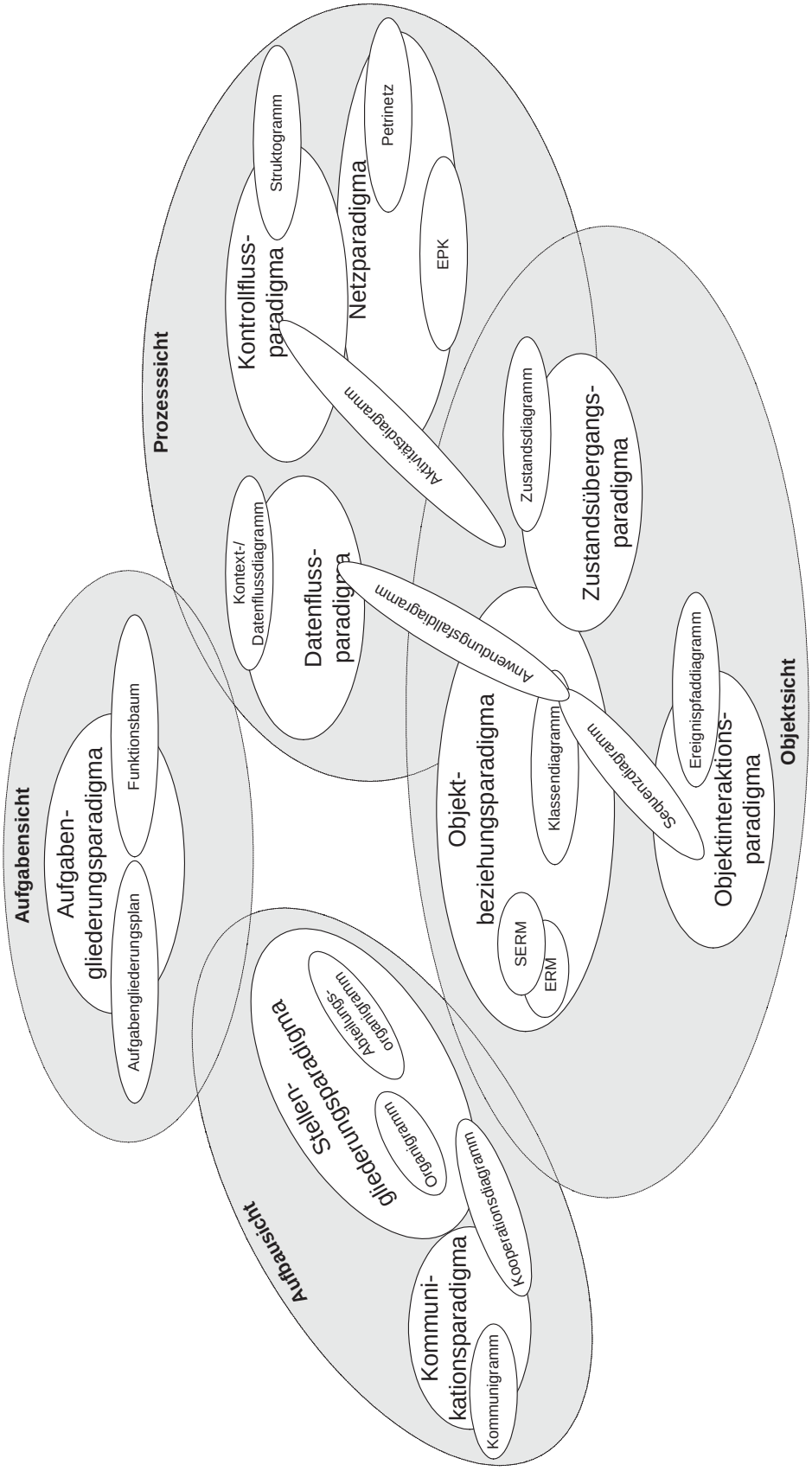
| **E$^3$-Element** | **Beschreibung** |
|---|---|
| Artefakttyp | Eingangs- oder Ausgangsgegenstand für einen **Aufgabentypen**. **Artefakttypen** werden in **Aufgabenobjekttypen** zusammengefasst. Im **Vorgehensmodell** zur E$^3$-Methode bildet das **E$^3$-Element** einen Artefakttypen. |
| Aufgabenobjekttyp | Gegenstand, an dem geforderte Tätigkeiten vollzogen werden (vgl. [Kosi62], S. 43). Der Aufgabenobjekttyp bildet einen Container für verschiedene Gegenstände, welche durch einen **Aufgabentypen** erzeugt, geändert oder in einen anderen Gegenstand eingebracht werden. Für das **E$^3$-Modell** kommt hier insbesondere das **E$^3$-Element** in Frage, welches als ein **Artefakttyp** zugeordnet werden kann. |
| Aufgabenträgertyp | Organisatorische Einheit, welche am **Aufgabenobjekttypen** den **Aufgaben-** bzw. **Verrichtungstypen** durchführt. Im Vorgehensmodell zur E$^3$-Methode handelt es sich um Rollen, die im Verlauf des ME entsprechend durch Projektmitglieder auszufüllen sind. |
| Aufgabentyp | Dauerhaft wirksame Aufforderung, etwas Bestimmtes zu tun (vgl. [LiSu89], S. 18). Der Aufgabentyp ist das zentrale **E$^3$-Element** der **Vorgangsebene**. |
| Bedingungstyp | Logischer Ausdruck, der die Wahrheitswerte *wahr* oder *falsch* liefert. Dient der Beschriftung von **Transitionen** und legt damit die zu deren Eintreten notwendigen Zustände fest. |
| E$^3$-Element | Bestandteil des **E$^3$-Modells**. Die Menge der E$^3$-Elemente gestattet die Beschreibung von Methoden und deren Produkten. |
| E$^3$-Modell | **Modell** zur Beschreibung von Methoden der Modellbildung. Das E$^3$-Modell wird in **Typ-**, **Instanzen-** und **Vorgangsebene** unterteilt. Die in den Ebenen enthaltenen **E$^3$-Elemente** gestatten die Abbildung der Konzepte, Sichten und Darstellungstechniken sowie die bei der Modellbildung angewendeten Vorgehensmodelle. |
| Ereignistyp | Auslöser und Ergebnis eines **Aufgabentypen**. Ereignisse werden durch Zustände von **Aufgabenobjekttypen** beschrieben. |
| Gruppierungstyp | Nach sachlogischen Kriterien gebildete Menge von **Aufgabentypen**. Die Gruppierung erfolgt anhand der Kriterien *Verrichtung*, *Objekt*, *Phase*, *Rang* oder *Zweckbeziehung* (vgl. [Kosi62], S. 42ff). |
| Instanzenebene | Zusammenfassung aller **Modelle**, **Objekte**, **Properties**, **Views**, **Viewobjekte**, **Viewproperties**, **Präsentationen**, **Präsentationsobjekte** und **Präsentationsproperties** des E$^3$-Modells. Im engeren Sinne werden damit alle einem bestimmten **Modell** zugeordneten Elemente der Instanzenebene bezeichnet. |
| Kontextebene | Zusammenfassung aller **Modelltypen**, **Modelle**, **Objekttypen**, **Objekte**, **Propertytypen** und **Properties** im **E$^3$-Modell**. In Bezug auf ein **Modell** innerhalb der **Instanzenebene** werden mit dem Begriff *Kontext* dessen **Objekte** und **Properties** bezeichnet. |
| Modell | System von Zeichen, das der verkürzten und zielgerichten Darstellung natürlicher oder künstlicher Originale in einem bestimmten Zeitraum dient. Ein Modell besteht aus **Objekten** und deren Repräsentation. |
| Metamodell | Zusammenfassung gleichartiger **Modelle**. Ein Metamodell definiert für seine **Modelle** alle Konzepte, Sichten und Darstellungstechniken in Form von **Objekttypen**, **Viewtypen** und **Präsentationstypen**. |

| **E$^3$-Element** | **Beschreibung** |
|---|---|
| Objekttyp | Zusammenfassung gleichartiger **Objekte**, die mittels **Viewobjekten** einer Sicht und mittels **Präsentationsobjekten** der grafischen Darstellung eines Modellsystems zugeordnet werden können. Einem Objekttypen können **Propertytypen** zugeordnet werden. |
| Objekt | Gedankliche Abstraktion oder Entsprechung einer realen Entität. Alle Eigenschaften der Objekte und deren Beziehungen zu anderen Objekten werden über **Properties** abgebildet. |
| Property | Eigenschaft eines **Objekts**. Gültige Werte eines Properties werden über dessen **Propertytypen** definiert. |
| Propertytyp | Merkmalsbeschreibung für **Objekttypen**. Ein Propertytyp definiert den Wertebereich der Properties und die Anzahl der maximal und minimal zuordenbaren Wertauspägung. Wird für einen **Objekttypen** ein **Objekt** erzeugt, so sind für alle dem **Objekttypen** zugeordneten Propertytypen jeweils ein Property anzulegen. |
| Präsentation | Grafische Darstellung einer **View** auf ein **Modell**. Eine Präsentation verwendet auf Basis der durch ihren **Präsentationstypen** definierten Darstellungstechniken **Präsentationsobjekte**, um **Modelle** grafisch zu symbolisieren. |
| Präsentationsebene | Zusammenfassung aller **Präsentationstypen**, **Präsentationen**, **Präsentationsobjekttypen**, **Präsentationsobjekte**, **Präsentationspropertytypen** und **Präsentationsproperties** innerhalb des E$^3$-Modells. Im engeren Sinne werden damit alle einem bestimmten **Viewtypen** zugeordneten Elemente der Präsentationsebene bezeichnet. |
| Präsentationsobjekt | Grafische Darstellung eines **Objekts** innerhalb einer **Präsentation**. Präsentationsobjekte ordnen einem **Viewobjekt** eine **Präsentation** zu. Auf diesem Wege werden Modellausschnitte im **E$^3$-Modell** symbolhaft abgebildet. Die Art der grafischen Darstellung wird über die für das **Viewobjekt** möglichen **Präsentationsobjekttypen** eingeschränkt. |
| Präsentationsobjekttyp | Zusammenfassung gleichartiger **Präsentationsobjekte**. Ein Präsentationsobjekttyp definiert die Art und Weise, mit der **Objekte** innerhalb einer **Präsentation** dargestellt werden. |
| Präsentationstyp | Darstellungstechnik für ein **Metamodell**. Präsentationstypen beschreiben die gültigen Formen der grafischen Darstellung von Modellausschnitten. Zu diesem Zweck fassen sie für diese Technik alle zulässigen **Präsentationsobjekttypen** zusammen. |
| Sachmitteltyp | Notwendige materielle und immaterielle Hilfsmittel für die Erfüllung von **Aufgabentypen**. Sachmitteltypen gehören nicht zu den **Aufgabenobjekttypen**, unterstützen jedoch den **Aufgabenträgertypen** bei der Erfüllung der Aufgabe (vgl. [LiSu89], S. 24). |
| Synchronisation | Zusammenführung von **Transitionen** in einem **Vorgehensmodell**. Für eine Synchronisation wird die Art der logischen Verknüpfung (AND, OR oder XOR) der eingehenden Transitionen als deren Name angegeben. |
| Transition | Verknüpfung von **Aufgabentypen**. Eine Transition kann verzweigt (siehe **Verzweigung**) oder mit anderen Transitionen zusammengeführt (siehe **Synchronisation**) werden. |

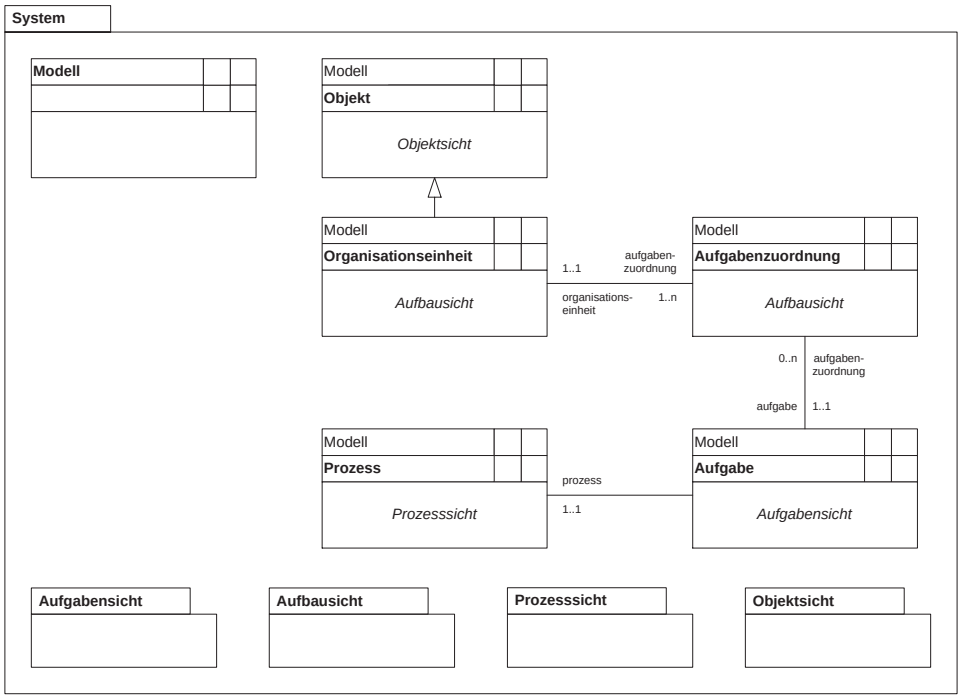
| **E$^3$-Element** | **Beschreibung** |
|---|---|
| Typebene | Zusammenfassung aller **Modell-**, **Objekt-**, **Property-**, **View-**, **Viewobjekt-**, **Viewproperty-**, **Präsentations-**, **Präsentationsobjekt-** und **Präsentationspropertytypen** des E$^3$-Modells. Im engeren Sinne werden damit alle einem bestimmten **Metamodell** zugeordneten Elemente der Typebene bezeichnet. |
| Verrichtungstyp | Tätigkeiten zur Erfüllung eines **Aufgabentypen**. Die Granularität des Verrichtungstypen ist abhängig von der Aufgabe und dem Zweck der Betrachtung. |
| Verzweigung | Parallelisierung von Transitionen. Bei einer Verzweigung erfolgen die ausgehenden Transitionen gleichzeitig. |
| View | Ausschnitt eines **Modells**. Eine View fasst **Viewobjekte** zusammen und wählt damit einen Teil eines **Modells** oder einer vorgelagerten **View** zur Darstellung innerhalb einer **Präsentation** aus. |
| Viewebene | Zusammenfassung aller **Viewtypen**, **Views**, **Viewobjekttypen**, **Viewobjekte**, **Viewpropertytypen** und **Viewproperties** innerhalb des **E$^3$-Modells**. Im engeren Sinne werden damit alle einem bestimmten **Metamodell** zugeordneten Elemente der Viewebene bezeichnet. |
| Viewobjekt | Auswahl eines **Objekts** innerhalb einer **View**. Ist ein **Objekt** Bestandteil einer bestimmten Sicht auf ein **Modell**, so existiert für dieses ein Viewobjekt mit einer Beziehung zur entsprechenden **View**. |
| Viewobjekttyp | Zusammenfassung gleichartiger **Viewobjekte**. Ein **Viewobjekttyp** ordnet einem **Viewtypen** einen **Objekttypen** zu. Damit wird zum Ausdruck gebracht, dass die Instanzen dieses **Objekttypen** zu der durch den **Viewtypen** definierten Sicht auf das **Modell** gehören können. |
| Viewproperty | Zuodnung eines **Property** zu einem **Viewobjekt**. Wird ein **Objekt** zum Bestandteil einer **View** auf ein **Modell**, so können für dessen Eigenschaften Viewproperties gebildet werden, um ebenfalls Bestandteil des Ausschnittes zu sein. |
| Viewpropertytyp | Zusammenfassung gleichartiger **Viewproperties**. |
| Viewtyp | Zusammenfassung gleichartiger **Views**. Ein Viewtyp fasst eine Menge an **Objekttypen** zu einem Modellausschnitt zusammen. Er referenziert zu diesem Zweck deren entsprechende **Viewobjekttypen**. |
| Vorgehensmodell | Beschreibung idealisierter, wiederholbarer und strukturierter Arbeitsabläufe in einem Projekt. Es legt Arbeitspakete und deren zeitliche Reihenfolge fest. |
| Vorgeordnetes ..., Vorgeordneter ... | Hilfskonstrukt bei der Definition von Sichten auf Sichten. Da den **E$^3$-Elementen** der **Viewebene** sowohl ein Element der **Kontext-** als auch der **Viewebene** selbst vorgeordnet werden kann, beziehen sich die Elemente der **Viewebene** stets auf ein *Vorgeordnetes Element*. Alle Elemente der **View-** und **Kontextebene** sind solch ein Vorgeordnetes Element, z. B. erben sowohl **Objekttyp** als auch **Viewobjekttyp** von *Vorgeordneter Objekttyp*. |
| Zieltyp | Definierter, angestrebter Zustand der Erfüllung einer Arbeitsaufgabe (vgl. [REFA92], S. 78). |

# 12 Das Referenzmetamodell der E$^3$-Methode

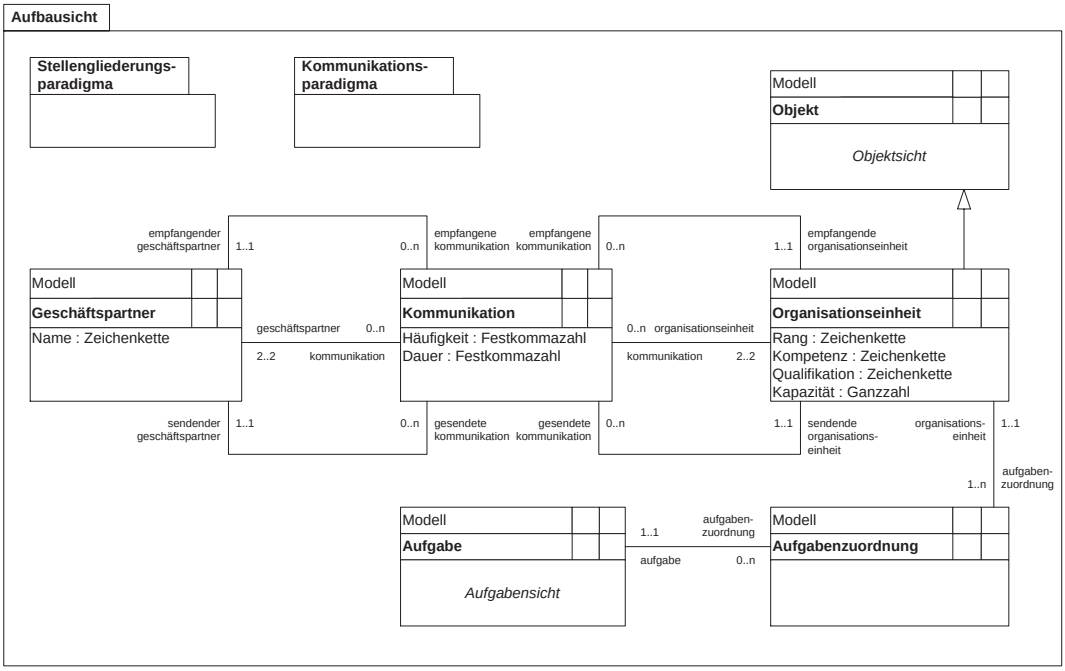
## 12.1 Flächendarstellung des Meta-Referenzmodellrahmens



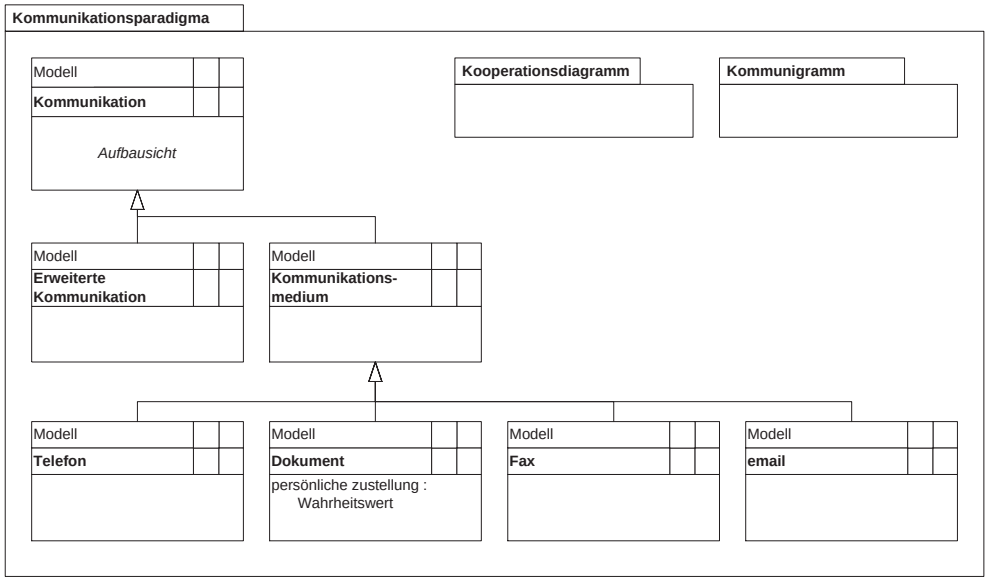
## 12.2 System




### 12.2.1 Aufbausicht



### 12.2.1.1 Kommunikationsparadigma



**Kooperationsdiagramm**

| Modell |
|---|
| **Geschäftspartner** |
| *Aufbausicht* |

| Modell |
|---|
| **Abteilung** |
| *Stellengliederungsparadigma* |

| Modell |
|---|
| **Stelle** |
| *Stellengliederungsparadigma* |

| Modell |
|---|
| **Erweiterte Kommunikation** |
| *Aufbausicht* |

| Modell |
|---|
| **Dokument** |
| *Stellengliederungsparadigma* |

| Modell |
|---|
| **Telefon** |
| *Stellengliederungsparadigma* |

| Modell | **Kooperationsdiagramm** |
|---|---|
| | |

| Kooperationsdiagramm | |
|---|---|
| Stelle | **Stelle** |

name
gesendete kommunikation
empfangene kommunikation
kommunikation

| Kooperationsdiagramm | |
|---|---|
| Geschäftspartner | **Geschäftspartner** |

name
gesendete kommunikation
empfangene kommunikation
kommunikation

| Kooperationsdiagramm | |
|---|---|
| Abteilung | **Abteilung** |

name
gesendete kommunikation
empfangene kommunikation
kommuniktion

| Kooperationsdiagramm | |
|---|---|
| Telefon | **Telefon** |

erweiterte kommunikation

| Kooperationsdiagramm | |
|---|---|
| Dokument | **Persönliche Dokumentübermittlung** |

erweiterte kommunikation

| Kooperationsdiagramm | |
|---|---|
| Dokument | **Unpersönliche Dokumentübermittlung** |

erweiterte kommunikation

| Kooperationsdiagramm | |
|---|---|
| Erweiterte Kommunikation | **Ungerichtete Erweiterte Kommunikation** |

geschäftspartner
organisationseinheit
kommunikationsmedium

| Kooperationsdiagramm | |
|---|---|
| Erweiterte Kommunikation | **Gerichtete Erweiterte Kommunikation** |

sendender geschäftspartner
empfangender geschäftspartner
sendende organisationseinheit
empfangende organisationseinheit
kommunikationsmedium

| Kooperations-diagramm | **Kooperations-diagramm** |
|---|---|
| | |

| Kooperationsdiagramm | |
|---|---|
| Stelle | **Stelle** |

name : Beschriftung
gesendete kommunikation : Verbinder
empfangene kommunikation : Verbinder
kommunikation : Verbinder

| Kooperationsdiagramm | |
|---|---|
| Geschäftspartner | **Geschäftspartner** |

name : Beschriftung
gesendete kommunikation : Verbinder
empfangene kommunikation : Verbinder
kommunikation : Verbinder

| Kooperationsdiagramm | |
|---|---|
| Abteilung | **Abteilung** |

name : Beschriftung
gesendete kommunikation : Verbinder
empfangene kommunikation : Verbinder
kommuniktion : Verbinder

| Kooperationsdiagramm | |
|---|---|
| Telefon | **Telefon** |

erweiterte kommunikation : Verbinder

| Kooperationsdiagramm | |
|---|---|
| Persönliche Dokumentübermittlung | **Persönliche Dokument-übermittlung** |

erweiterte kommunikation : Verbinder

| Kooperationsdiagramm | |
|---|---|
| Unpersönliche Dokumentübermittlung | **Unpersönliche Dokumentübermittlung** |

erweiterte kommunikation : Verbinder



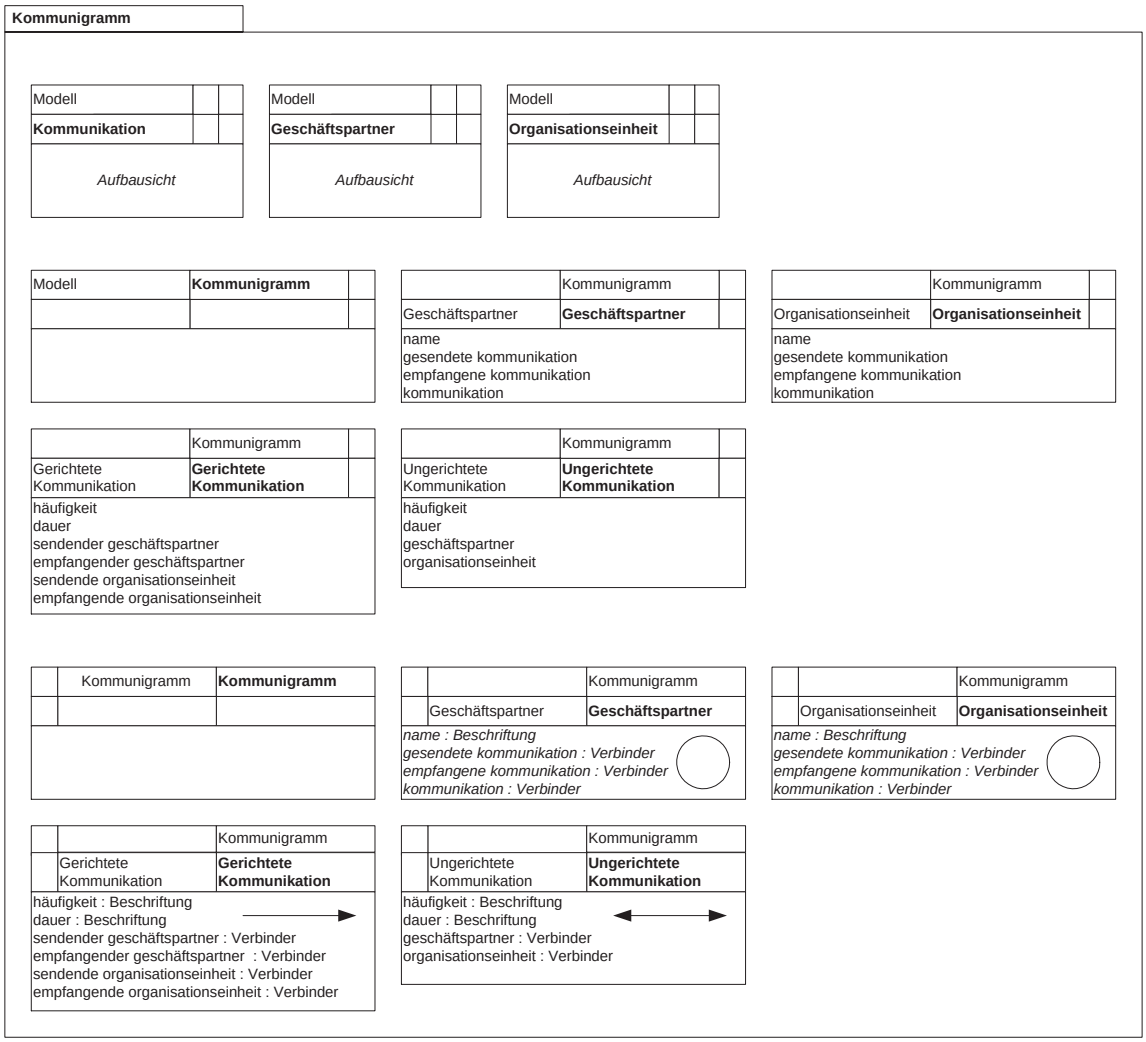
| Kooperationsdiagramm | |
|---|---|
| Ungerichtete Erweiterte Kommunikation | **Ungerichtete Erweiterte Kommunikation** |

geschäftspartner : Verbinder
organisationseinheit : Verbinder
kommunikationsmedium : Verbinder

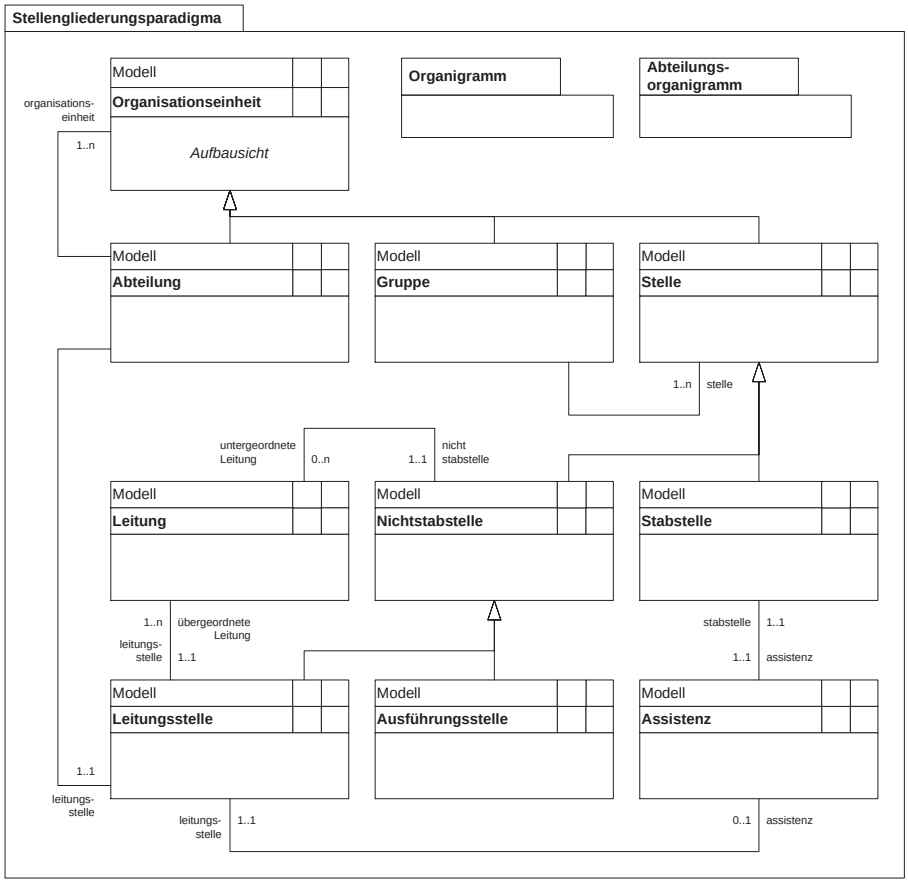


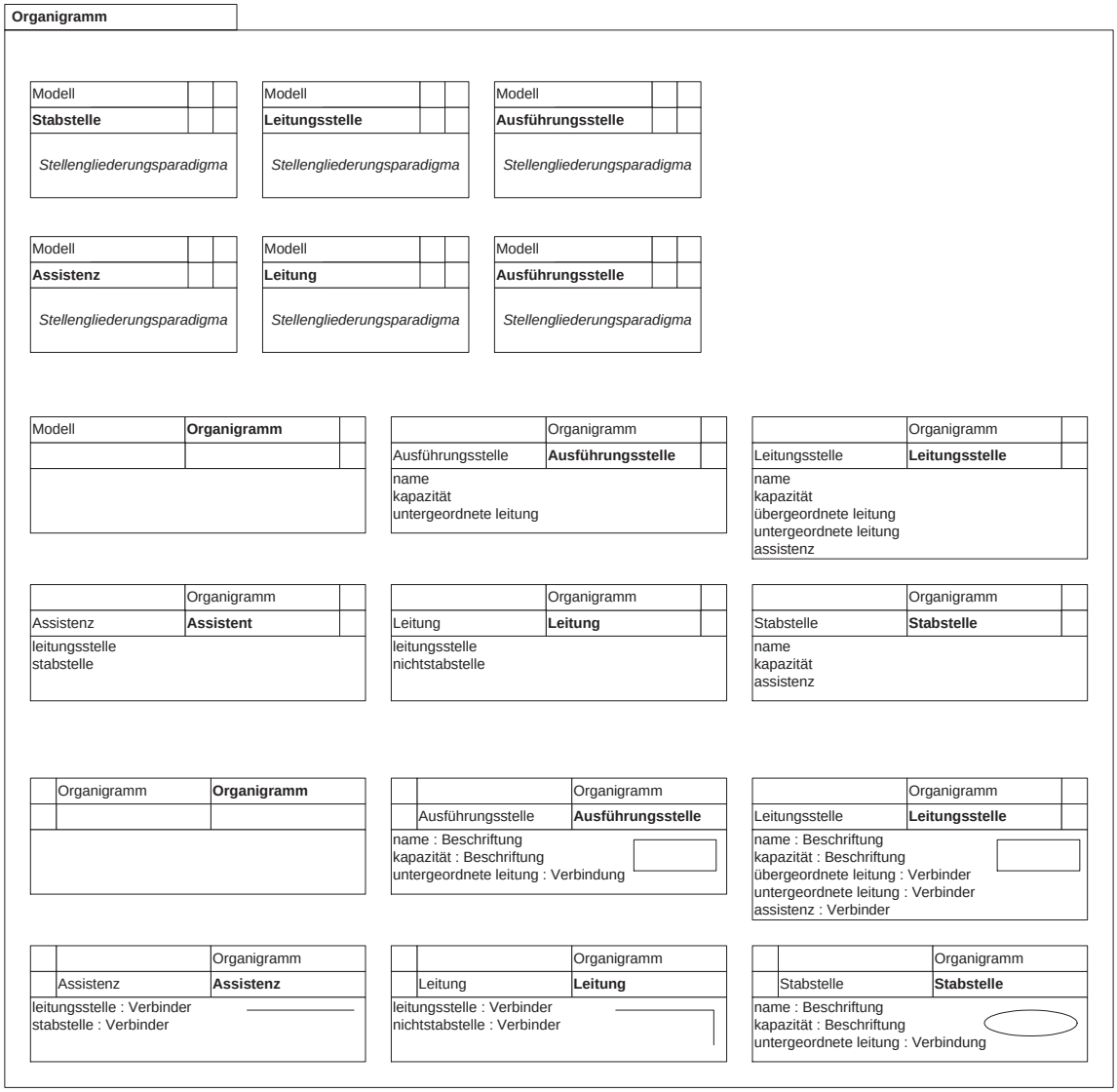
| Kooperationsdiagramm | |
|---|---|
| Gerichtete Erweiterte Kommunikation | **Gerichtete Erweiterte Kommunikation** |

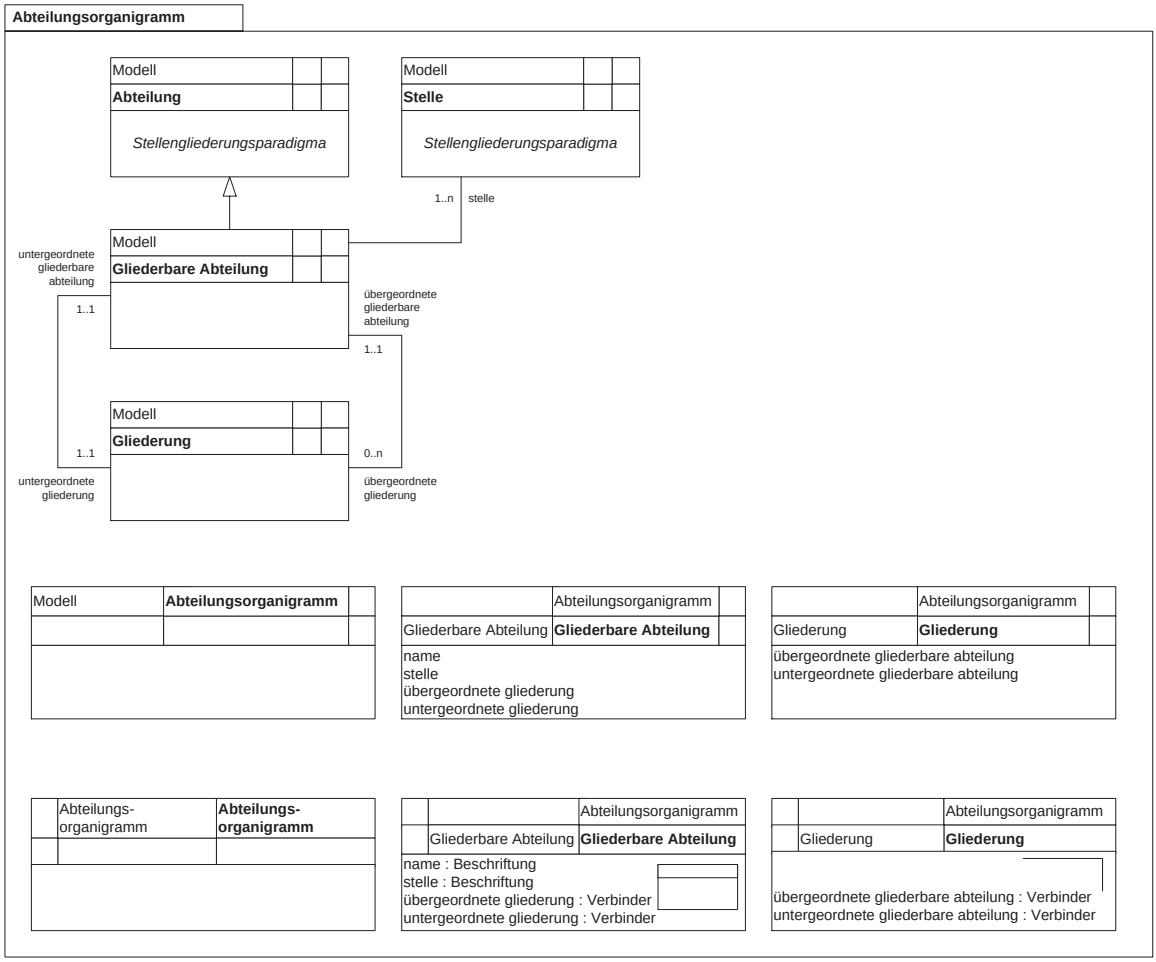
sendender geschäftspartner : Verbinder
empfangender geschäftspartner : Verbinder
sendende organisationseinheit : Verbinder
empfangende organisationseinheit : Verbinder
kommunikationsmedium : Verbinder


Kommunigramm
Modell
Kommunikation
Aufbausicht
Modell
Geschäftspartner
Aufbausicht
Modell
Organisationseinheit
Aufbausicht
Modell
Kommunigramm
Kommunigramm
Geschäftspartner
Geschäftspartner
name
gesendete kommunikation
empfangene kommunikation
kommunikation
Kommunigramm
Organisationseinheit
Organisationseinheit
name
gesendete kommunikation
empfangene kommunikation
kommunikation
Kommunigramm
Gerichtete Kommunikation
Gerichtete Kommunikation
häufigkeit
dauer
sendender geschäftspartner
empfangender geschäftspartner
sendende organisationseinheit
empfangende organisationseinheit
Kommunigramm
Ungerichtete Kommunikation
Ungerichtete Kommunikation
häufigkeit
dauer
geschäftspartner
organisationseinheit
Kommunigramm
Kommunigramm
Kommunigramm
Geschäftspartner
Geschäftspartner
name : Beschriftung
gesendete kommunikation : Verbinder
empfangene kommunikation : Verbinder
kommunikation : Verbinder
Kommunigramm
Organisationseinheit
Organisationseinheit
name : Beschriftung
gesendete kommunikation : Verbinder
empfangene kommunikation : Verbinder
kommunikation : Verbinder
Kommunigramm
Gerichtete Kommunikation
Gerichtete Kommunikation
häufigkeit : Beschriftung
dauer : Beschriftung
sendender geschäftspartner : Verbinder
empfangender geschäftspartner : Verbinder
sendende organisationseinheit : Verbinder
empfangende organisationseinheit : Verbinder
Kommunigramm
Ungerichtete Kommunikation
Ungerichtete Kommunikation
häufigkeit : Beschriftung
dauer : Beschriftung
geschäftspartner : Verbinder
organisationseinheit : Verbinder

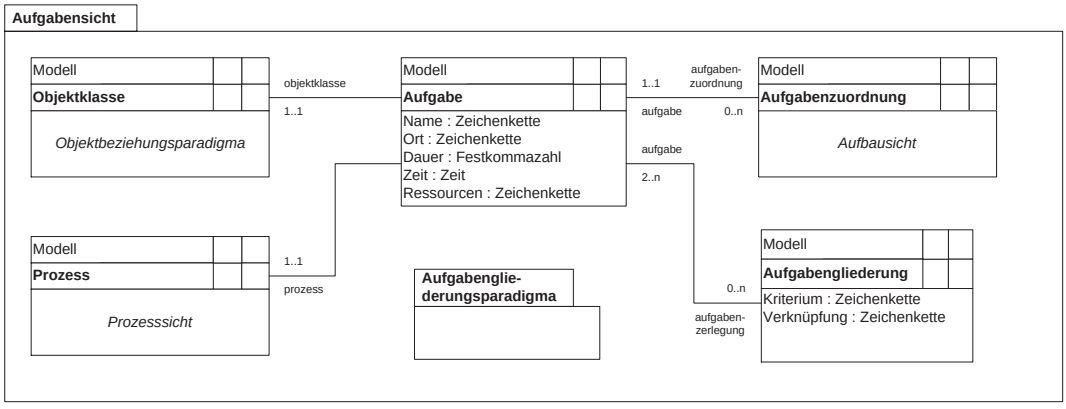
### 12.2.1.2 Stellengliederungsparadigma



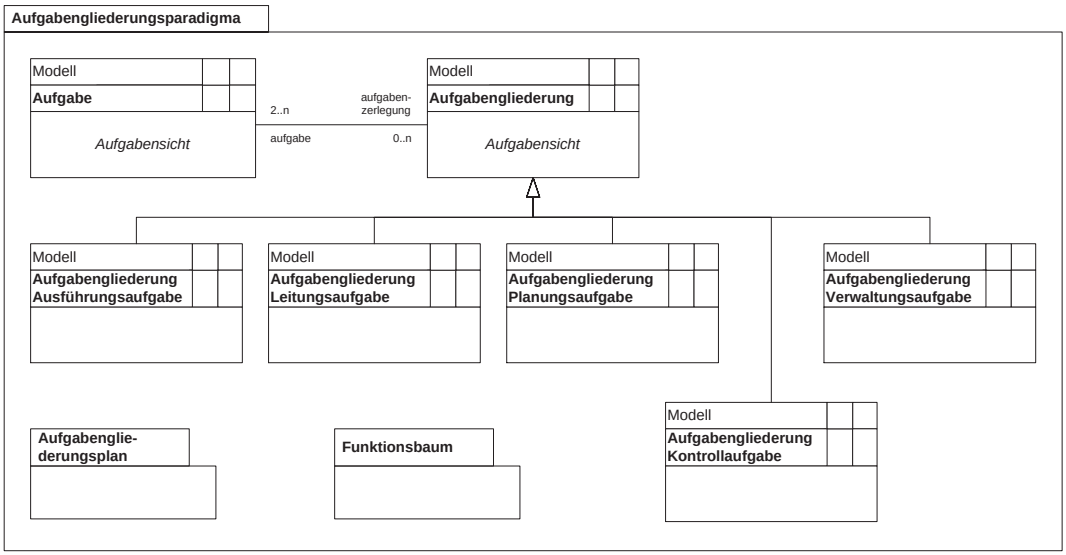

Organigramm
Modell
Stabstelle
Stellengliederungsparadigma
Modell
Leitungsstelle
Stellengliederungsparadigma
Modell
Ausführungsstelle
Stellengliederungsparadigma
Modell
Assistenz
Stellengliederungsparadigma
Modell
Leitung
Stellengliederungsparadigma
Modell
Ausführungsstelle
Stellengliederungsparadigma
Modell
Organigramm
Organigramm
Ausführungsstelle
Ausführungsstelle
name
kapazität
untergeordnete leitung
Organigramm
Leitungsstelle
Leitungsstelle
name
kapazität
übergeordnete leitung
untergeordnete leitung
assistenz
Organigramm
Assistenz
Assistent
leitungsstelle
stabstelle
Organigramm
Leitung
Leitung
leitungsstelle
nichtstabstelle
Organigramm
Stabstelle
Stabstelle
name
kapazität
assistenz
Organigramm
Organigramm
Organigramm
Ausführungsstelle
Ausführungsstelle
name : Beschriftung
kapazität : Beschriftung
untergeordnete leitung : Verbindung
Organigramm
Leitungsstelle
Leitungsstelle
name : Beschriftung
kapazität : Beschriftung
übergeordnete leitung : Verbinder
untergeordnete leitung : Verbinder
assistenz : Verbinder
Organigramm
Assistenz
Assistenz
leitungsstelle : Verbinder
stabstelle : Verbinder
Organigramm
Leitung
Leitung
leitungsstelle : Verbinder
nichtstabstelle : Verbinder
Organigramm
Stabstelle
Stabstelle
name : Beschriftung
kapazität : Beschriftung
untergeordnete leitung : Verbindung

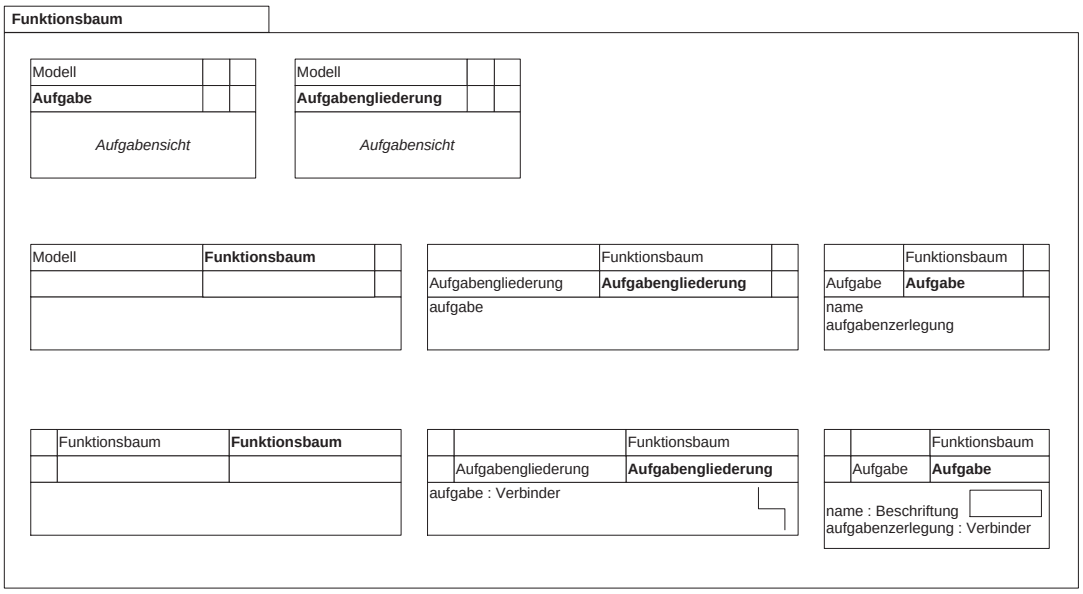


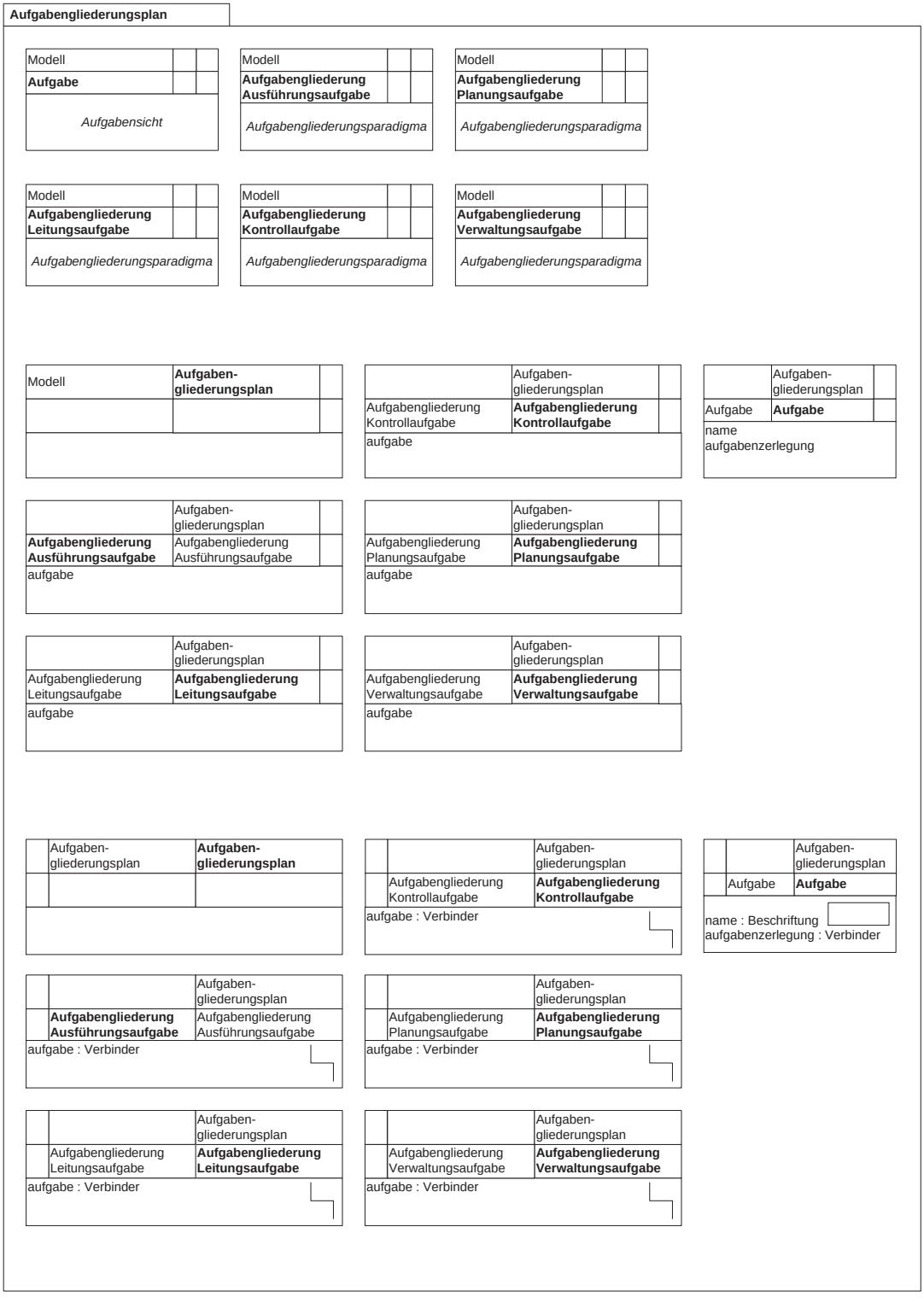

## 12.2.2 Aufgabensicht



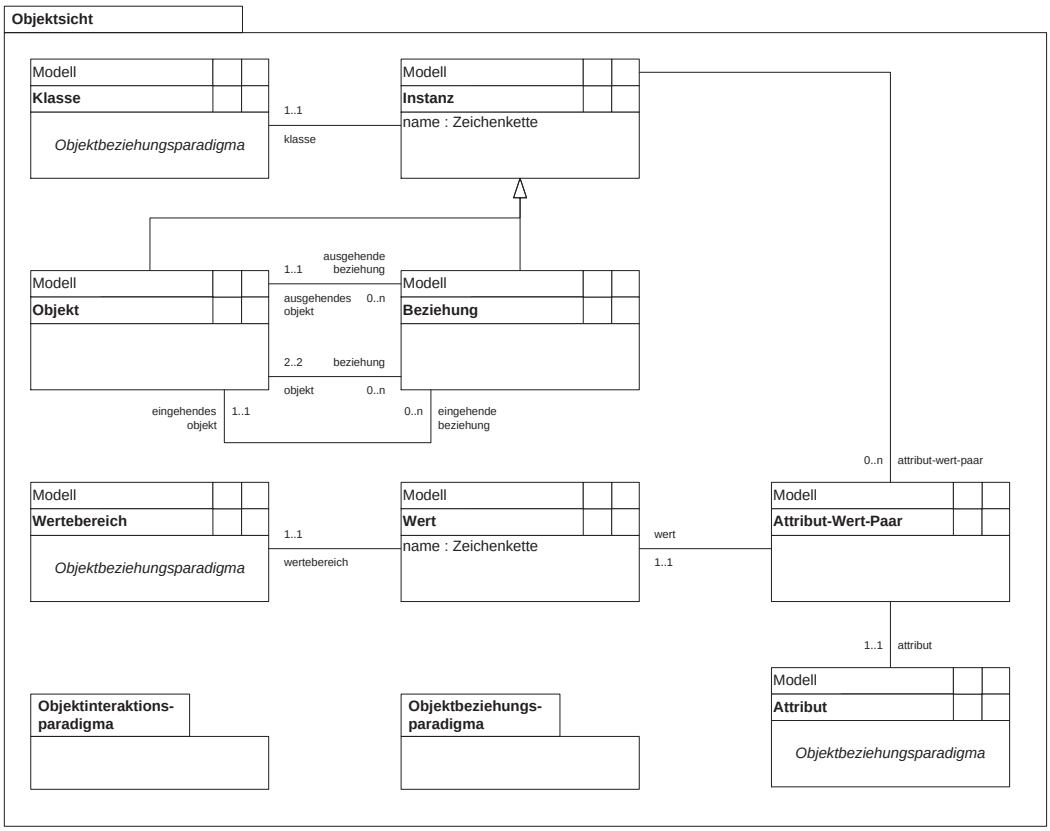
### 12.2.2.1 Aufgabengliederungsparadigma



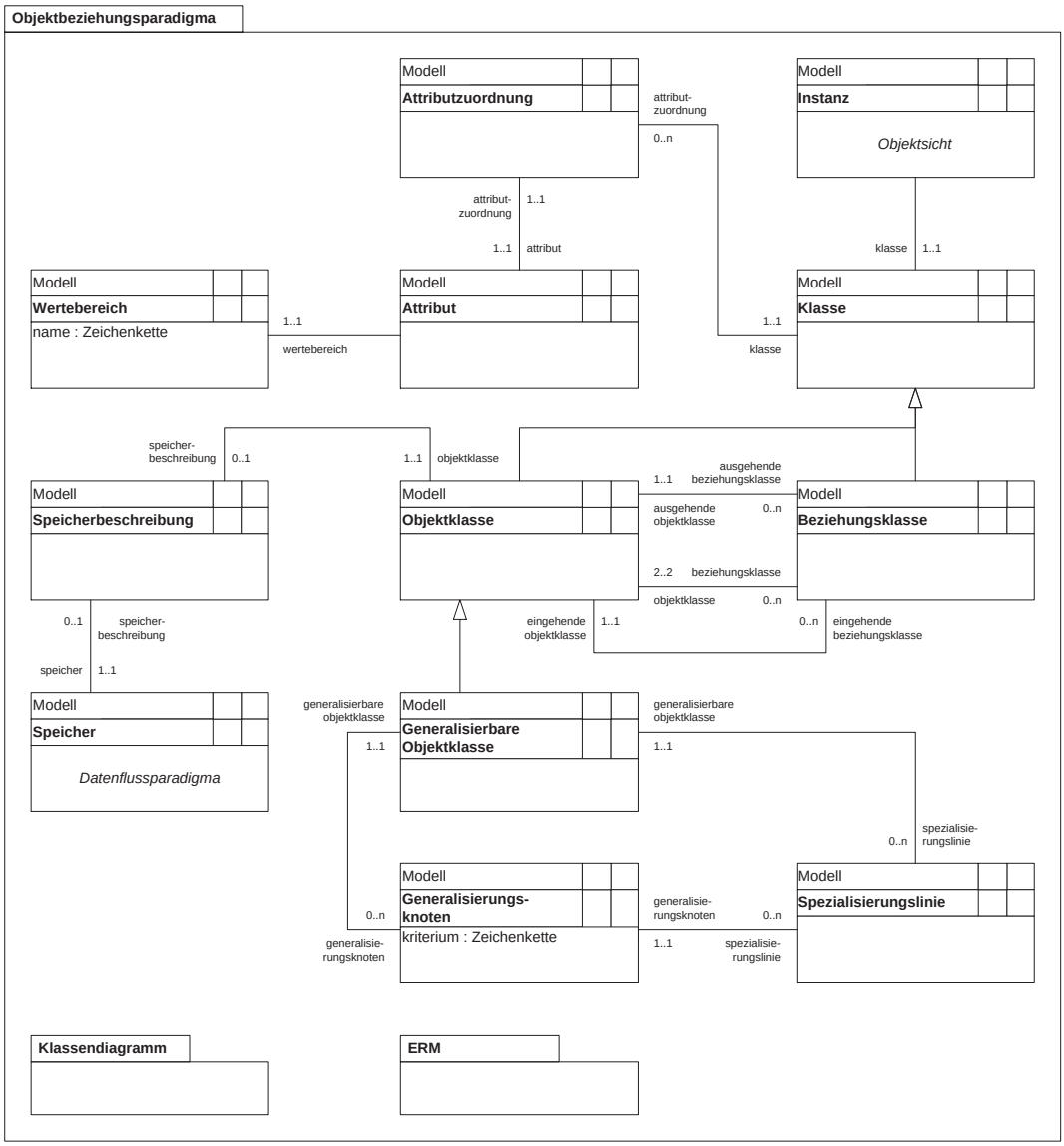



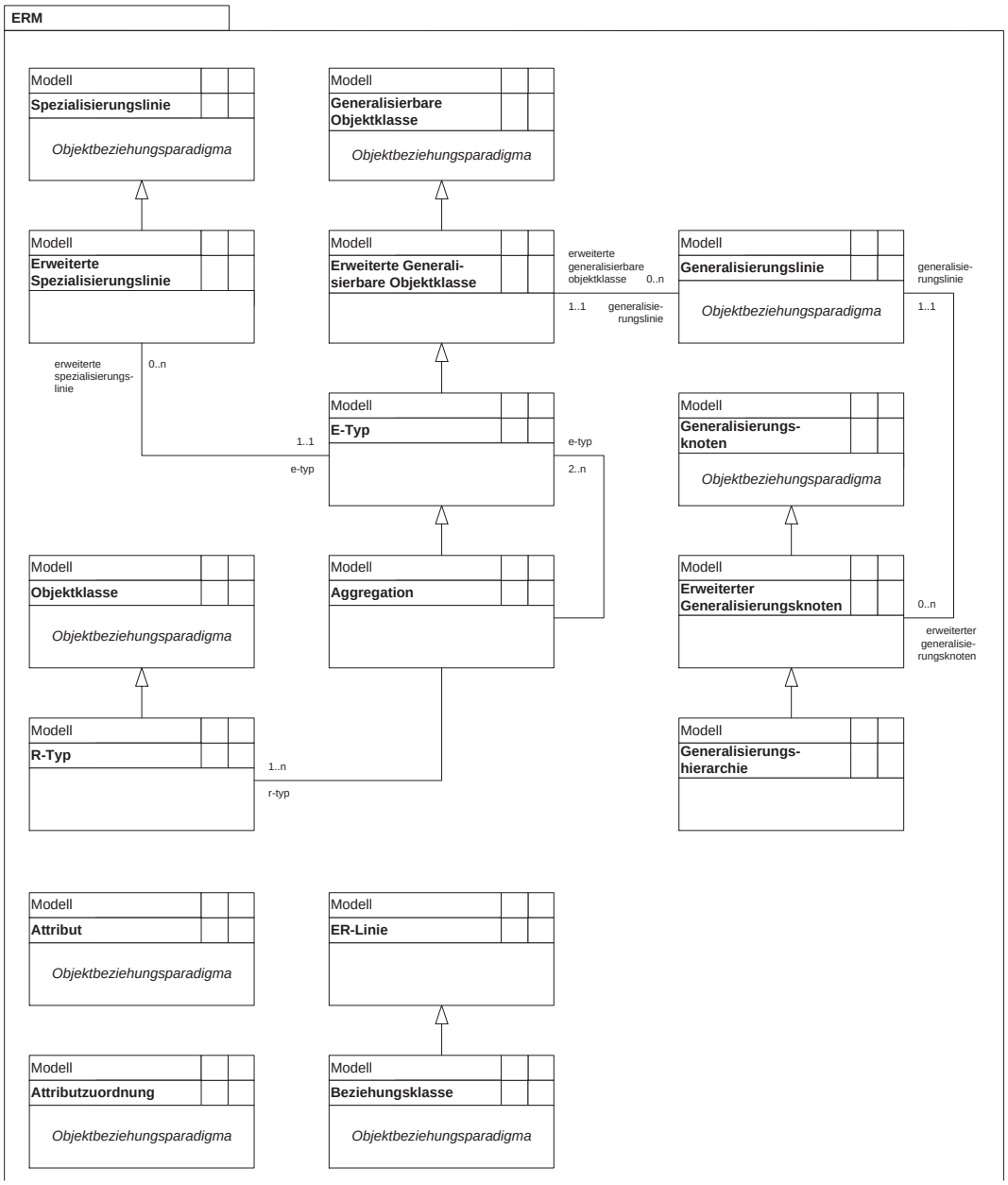

Aufgabengliederungsplan
Modell
Aufgabe
Aufgabensicht
Modell
Aufgabengliederung Ausführungsaufgabe
Aufgabengliederungsparadigma
Modell
Aufgabengliederung Planungsaufgabe
Aufgabengliederungsparadigma
Modell
Aufgabengliederung Leitungsaufgabe
Aufgabengliederungsparadigma
Modell
Aufgabengliederung Kontrollaufgabe
Aufgabengliederungsparadigma
Modell
Aufgabengliederung Verwaltungsaufgabe
Aufgabengliederungsparadigma
Modell
Aufgaben-gliederungsplan
Aufgaben-gliederungsplan
Aufgabengliederung Kontrollaufgabe
Aufgabengliederung Kontrollaufgabe
aufgabe
Aufgaben-gliederungsplan
Aufgabe
Aufgabe
name
aufgabenzerlegung
Aufgaben-gliederungsplan
Aufgabengliederung Ausführungsaufgabe
Aufgabengliederung Ausführungsaufgabe
aufgabe
Aufgaben-gliederungsplan
Aufgabengliederung Planungsaufgabe
Aufgabengliederung Planungsaufgabe
aufgabe
Aufgaben-gliederungsplan
Aufgabengliederung Leitungsaufgabe
Aufgabengliederung Leitungsaufgabe
aufgabe
Aufgaben-gliederungsplan
Aufgabengliederung Verwaltungsaufgabe
Aufgabengliederung Verwaltungsaufgabe
aufgabe
Aufgaben-gliederungsplan
Aufgaben-gliederungsplan
Aufgaben-gliederungsplan
Aufgabengliederung Kontrollaufgabe
Aufgabengliederung Kontrollaufgabe
aufgabe : Verbinder
Aufgaben-gliederungsplan
Aufgabe
Aufgabe
name : Beschriftung
aufgabenzerlegung : Verbinder
Aufgaben-gliederungsplan
Aufgabengliederung Ausführungsaufgabe
Aufgabengliederung Ausführungsaufgabe
aufgabe : Verbinder
Aufgaben-gliederungsplan
Aufgabengliederung Planungsaufgabe
Aufgabengliederung Planungsaufgabe
aufgabe : Verbinder
Aufgaben-gliederungsplan
Aufgabengliederung Leitungsaufgabe
Aufgabengliederung Leitungsaufgabe
aufgabe : Verbinder
Aufgaben-gliederungsplan
Aufgabengliederung Verwaltungsaufgabe
Aufgabengliederung Verwaltungsaufgabe
aufgabe : Verbinder

### 12.2.3 Objektsicht



### 12.2.3.1 Objektbeziehungsparadigma




ERM
Modell
Spezialisierungslinie
Objektbeziehungsparadigma
Modell
Generalisierbare Objektklasse
Objektbeziehungsparadigma
Modell
Erweiterte Spezialisierungslinie
Modell
Erweiterte Generalisierbare Objektklasse
erweiterte generalisierbare objektklasse
0..n
1..1
generalisierungslinie
Modell
Generalisierungslinie
Objektbeziehungsparadigma
generalisierungslinie
1..1
erweiterte spezialisierungslinie
0..n
Modell
E-Typ
1..1
e-typ
e-typ
2..n
Modell
Generalisierungsknoten
Objektbeziehungsparadigma
Modell
Objektklasse
Objektbeziehungsparadigma
Modell
Aggregation
Modell
Erweiterter Generalisierungsknoten
0..n
erweiterter generalisierungsknoten
Modell
R-Typ
1..n
r-typ
Modell
Generalisierungshierarchie
Modell
Attribut
Objektbeziehungsparadigma
Modell
ER-Linie
Modell
Attributzuordnung
Objektbeziehungsparadigma
Modell
Beziehungsklasse
Objektbeziehungsparadigma

| Modell | **ERM** |
|---|---|
| | |

| | ERM |
|---|---|
| R-Typ | **R-Typ** |

name
ausgegangene beziehungsklasse
attributzuordnung

| | ERM |
|---|---|
| E-Typ | **E-Typ** |

name
eingegangene beziehungsklasse
generalisierungslinie
erweiterte spezialisierungslinie
attributzuordnung

| | ERM |
|---|---|
| Aggregation | **Aggregation** |

e-typ
r-typ
name
eingegangene beziehungsklasse
generalisierungslinie
erweiterte spezialisierungslinie

| | ERM |
|---|---|
| ER-Linie | **ER-Linie** |

eingehende objektklasse
ausgehende objektklasse
kardinalität

| | ERM |
|---|---|
| Attributzuordnung | **Attributzuordnung** |

attribut
klasse

| | ERM |
|---|---|
| Attribut | **Attribut** |

name
attributzuordnung

| | ERM |
|---|---|
| Erweiterte Spezialisierungslinie | **Subtypen-hierarchie** |

e-typ
generalisierbare objektklasse

| | ERM |
|---|---|
| Erweiterte Spezialisierungslinie | **Erweiterte Spezialisierungslinie** |

generalisierungsknoten
generalisierbare objektklasse

| | ERM |
|---|---|
| Generalisierungshierarchie | **Generalisierungshierarchie** |

generalisierungskriterium
generalisierbare objektklasse
spezialisierungslinie

| | ERM |
|---|---|
| Generalisierungslinie | **Generalisierungslinie** |

erweiterter generalisierungsknoten
erweiterte generalisierbare objektklasse

| | ERM | **ERM** |
|---|---|---|
| | | |

| | | ERM |
|---|---|---|
| | E-Typ | **E-Typ** |

name : Beschriftung
eingegangene beziehungsklasse : Verbinder
generalisierungslinie : Verbinder
attributzuordnung : Verbinder
erweiterte spezialisierungslinie : Verbinder

| | | ERM |
|---|---|---|
| | R-Typ | **R-Typ** |

name : Beschriftung
attributzuordnung : Verbinder
ausgegangene beziehungsklasse : Verbinder

| | | ERM |
|---|---|---|
| | Aggregation | **Aggregation** |

e-typ : Verbinder
r-typ : Verbinder
name : Beschriftung
generalisierungslinie : Verbinder
eingegangene beziehungsklasse : Verbinder
erweiterte spezialisierungslinie : Verbinder

| | | ERM |
|---|---|---|
| | ER-Linie | **ER-Linie** |

kardinalität : Beschriftung
eingehende objektklasse : Verbinder
ausgehende objektklasse : Verbinder

| | | ERM |
|---|---|---|
| | Attributzuordnung | **Attributzuordnung** |

attribut : Verbinder
klasse : Verbinder

| | | ERM |
|---|---|---|
| | Attribut | **Attribut** |

name : Beschriftung
attributzuordnung : Verbinder

| | | ERM |
|---|---|---|
| | Subtypen-hierarchie | **Subtypen-hierarchie** |

e-typ : Verbinder
generalisierbare objektklasse : Verbinder

| | | ERM |
|---|---|---|
| | Erweiterte Spezialisierungslinie | **Erweiterte Spezialisierungslinie** |

generalisierbare objektklasse : Verbinder
generalisierungsknoten : Verbinder

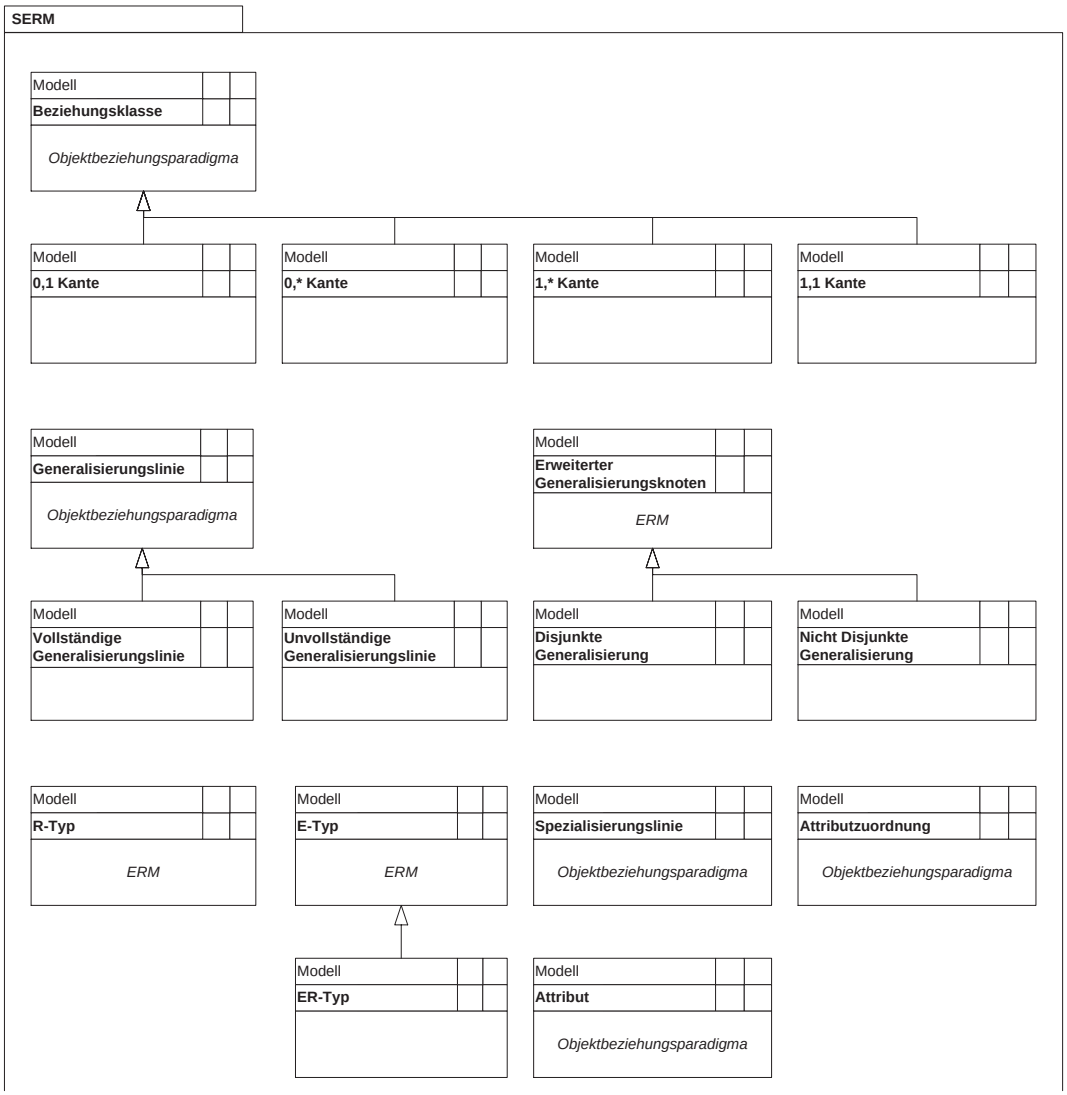
| | | ERM |
|---|---|---|
| | Generalisierungshierarchie | **Generalisierungshierarchie** |

spezialisierungslinie : Verbinder
generalisierungskriterium : Verbinder
generalisierbare objektklasse : Verbinder

| | | ERM |
|---|---|---|
| | Generalisierungslinie | **Generalisierungslinie** |

erweiterte generalisierbare objektklasse : Verbinder
erweiterter generalisierungsknoten : Verbinder

| **SERM** |
|---|
| |


SERM
Modell
Beziehungsklasse
Objektbeziehungsparadigma
Modell
0,1 Kante
Modell
0,* Kante
Modell
1,* Kante
Modell
1,1 Kante
Modell
Generalisierungslinie
Objektbeziehungsparadigma
Modell
Erweiterter Generalisierungsknoten
ERM
Modell
Vollständige Generalisierungslinie
Modell
Unvollständige Generalisierungslinie
Modell
Disjunkte Generalisierung
Modell
Nicht Disjunkte Generalisierung
Modell
R-Typ
ERM
Modell
E-Typ
ERM
Modell
Spezialisierungslinie
Objektbeziehungsparadigma
Modell
Attributzuordnung
Objektbeziehungsparadigma
Modell
ER-Typ
Modell
Attribut
Objektbeziehungsparadigma

| Modell | **SERM** |
|---|---|

| | SERM |
|---|---|
| R-Typ | **R-Typ** |

name
ausgegangene beziehungsklasse
attributzuordnung

| | SERM |
|---|---|
| E-Typ | **E-Typ** |

name
eingegangene beziehungsklasse
generalisierungslinie
erweiterte spezialisierungslinie
attributzuordnung

| | SERM |
|---|---|
| 0,1 Kante | **0,1 Kante** |

eingehende objektklasse
ausgehende objektklasse

| | ERM |
|---|---|
| E-Typ | **ER-Typ** |

name
eingegangene beziehungsklasse
ausgegangene beziehungsklasse
generalisierungslinie
erweiterte spezialisierungslinie
attributzuordnung

| | SERM |
|---|---|
| Attribut | **Attribut** |

name
attributzuordnung

| | SERM |
|---|---|
| Attributzuordnung | **Attributzuordnung** |

attribut
klasse

| | SERM |
|---|---|
| 1,1 Kante | **1,1 Kante** |

eingehende objektklasse
ausgehende objektklasse

| | SERM |
|---|---|
| Unvollständige Generalisierungslinie | **Unvollständige Generalisierungslinie** |

erweiterter generalisierungsknoten
erweiterte generalisierbare objektklasse

| | SERM |
|---|---|
| Disjunkte Generalisierung | **Disjunkte Generalisierung** |

spezialisierungslinie
generalisierungslinie
generalisierungskriterium

| | SERM |
|---|---|
| Spezialisierungslinie | **Spezialisierungslinie** |

genererealisierungsknoten
generalisierbare objektklasse

| | SERM |
|---|---|
| 1,* Kante | **1,*Kante** |

eingehende objektklasse
ausgehende objektklasse

| | SERM |
|---|---|
| Vollständige Generalisierungslinie | **Vollständige Generalisierungslinie** |

erweiterter generalisierungsknoten
erweiterte generalisierbare objektklasse

| | SERM |
|---|---|
| Nicht Disjunkte Generalisierung | **Nicht Disjunkte Generalisierung** |

spezialisierungslinie
generalisierungslinie
generalisierungskriterium

| | SERM |
|---|---|
| 0,* Kante | **0,*Kante** |

eingehende objektklasse
ausgehende objektklasse

| | SERM | **SERM** |
|---|---|---|

| | | SERM |
|---|---|---|
| | R-Typ | **R-Typ** |

name : Beschriftung
attributzuordnung : Verbinder
ausgegangene beziehungsklasse : Verbinder

| | | SERM |
|---|---|---|
| | E-Typ | **E-Typ** |

name : Beschriftung
eingegangene beziehungsklasse : Verbinder
generalisierungslinie : Verbinder
attributzuordnung : Verbinder
erweiterte spezialisierungslinie : Verbinder

| | | SERM |
|---|---|---|
| | 0,1 Kante | **0,1 Kante** |

eingehende objektklasse : Verbinder
ausgehende objektklasse : Verbinder

| | | SERM |
|---|---|---|
| | ER-Typ | **ER-Typ** |

name : Beschriftung
eingegangene beziehungsklasse : Verbinder

ausgegangene beziehungsklasse : Verbinder
generalisierungslinie : Verbinder
attributzuordnung : Verbinder
erweiterte spezialisierungslinie : Verbinder

| | | SERM |
|---|---|---|
| | Attribut | **Attribut** |

name : Beschriftung
attributzuordnung : Verbinder

| | | SERM |
|---|---|---|
| | Attributzuordnung | **Attributzuordnung** |

attribut : Verbinder
klasse : Verbinder



| | | SERM |
|---|---|---|
| | 1,1 Kante | **1,1 Kante** |

eingehende objektklasse : Verbinder
ausgehende objektklasse : Verbinder

| | | SERM |
|---|---|---|
| | Unvollständige Generalisierungslinie | **Unvollständige Generalisierungslinie** |

erweiterter generalisierungsknoten : Verbinder



erweiterte generalisierbare objektklasse : Verbinder

| | | SERM |
|---|---|---|
| | Disjunkte Generalisierung | **Disjunkte Generalisierung** |

spezialisierungslinie : Verbinder
generalisierungslinie : Verbinder
generalisierungskriterium : Beschriftung

| | | SERM |
|---|---|---|
| | Spezialisierungslinie | **Spezialisierungslinie** |

generalisierbare objektklasse : Verbinder
generalisierungsknoten : Verbinder



| | | SERM |
|---|---|---|
| | 1,* Kante | **1,*Kante** |

eingehende objektklasse : Verbinder
ausgehende objektklasse : Verbinder

| | | SERM |
|---|---|---|
| | Vollständige Generalisierungslinie | **Vollständige Generalisierungslinie** |

erweiterter generalisierungsknoten : Verbinder



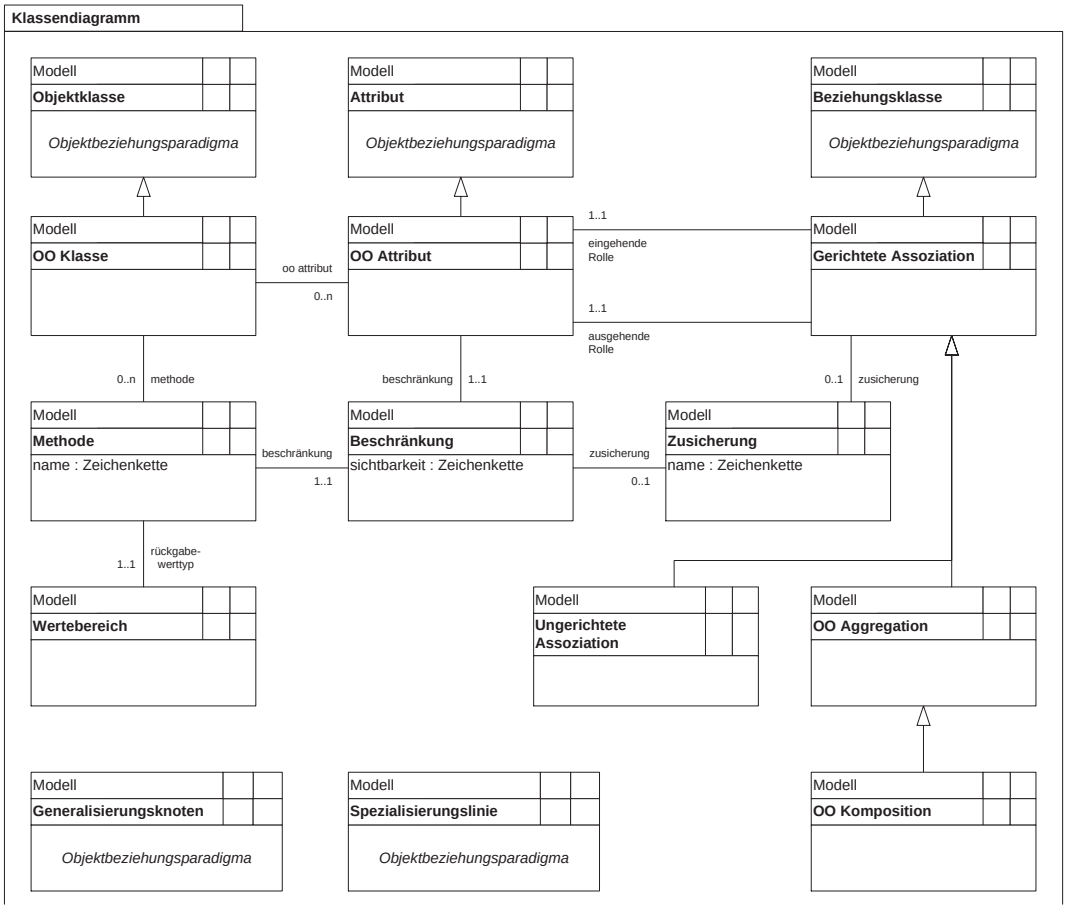
erweiterte generalisierbare objektklasse : Verbinder

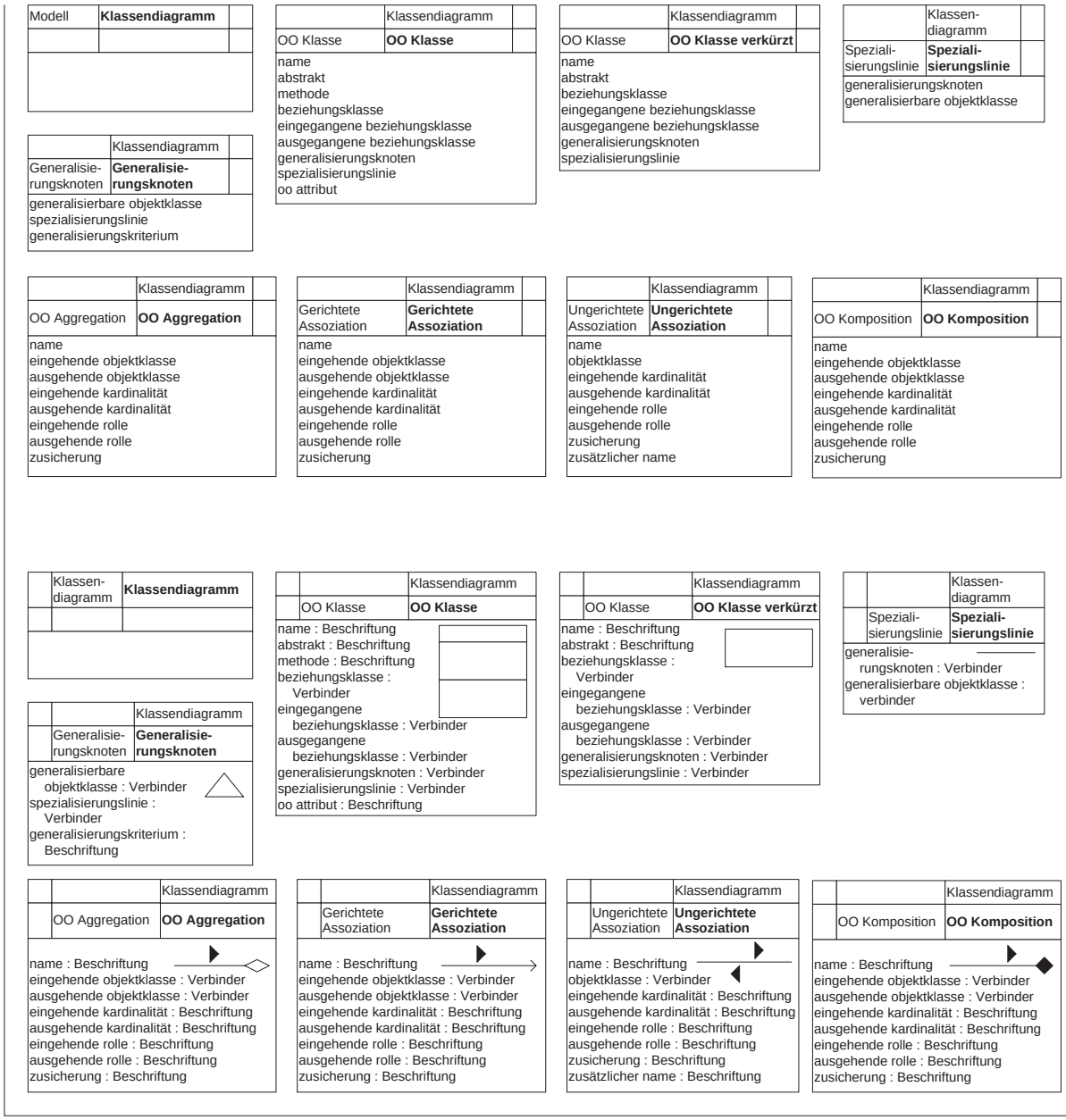
| | | SERM |
|---|---|---|
| | Nicht Disjunkte Generalisierung | **Nicht Disjunkte Generalisierung** |

spezialisierungslinie : Verbinder
generalisierungslinie : Verbinder
generalisierungskriterium : Beschriftung

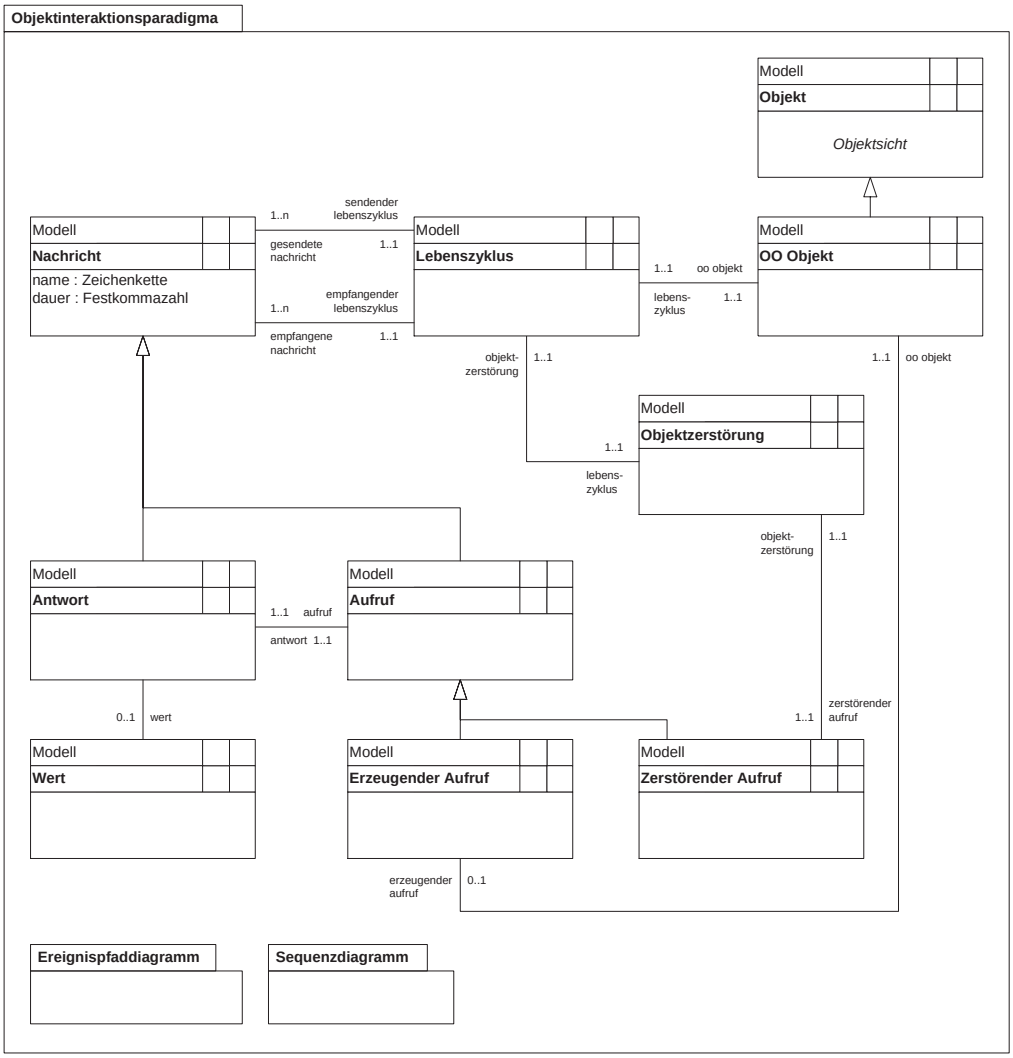
| | | SERM |
|---|---|---|
| | 0,* Kante | **0,*Kante** |

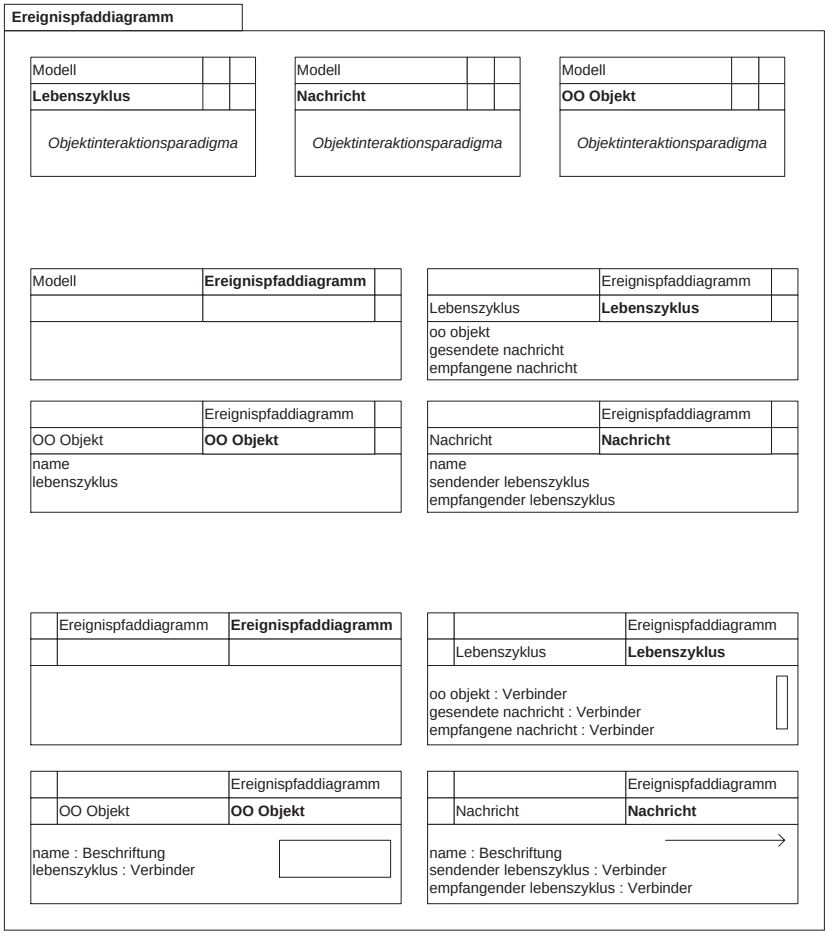
eingehende objektklasse : Verbinder
ausgehende objektklasse : Verbinder

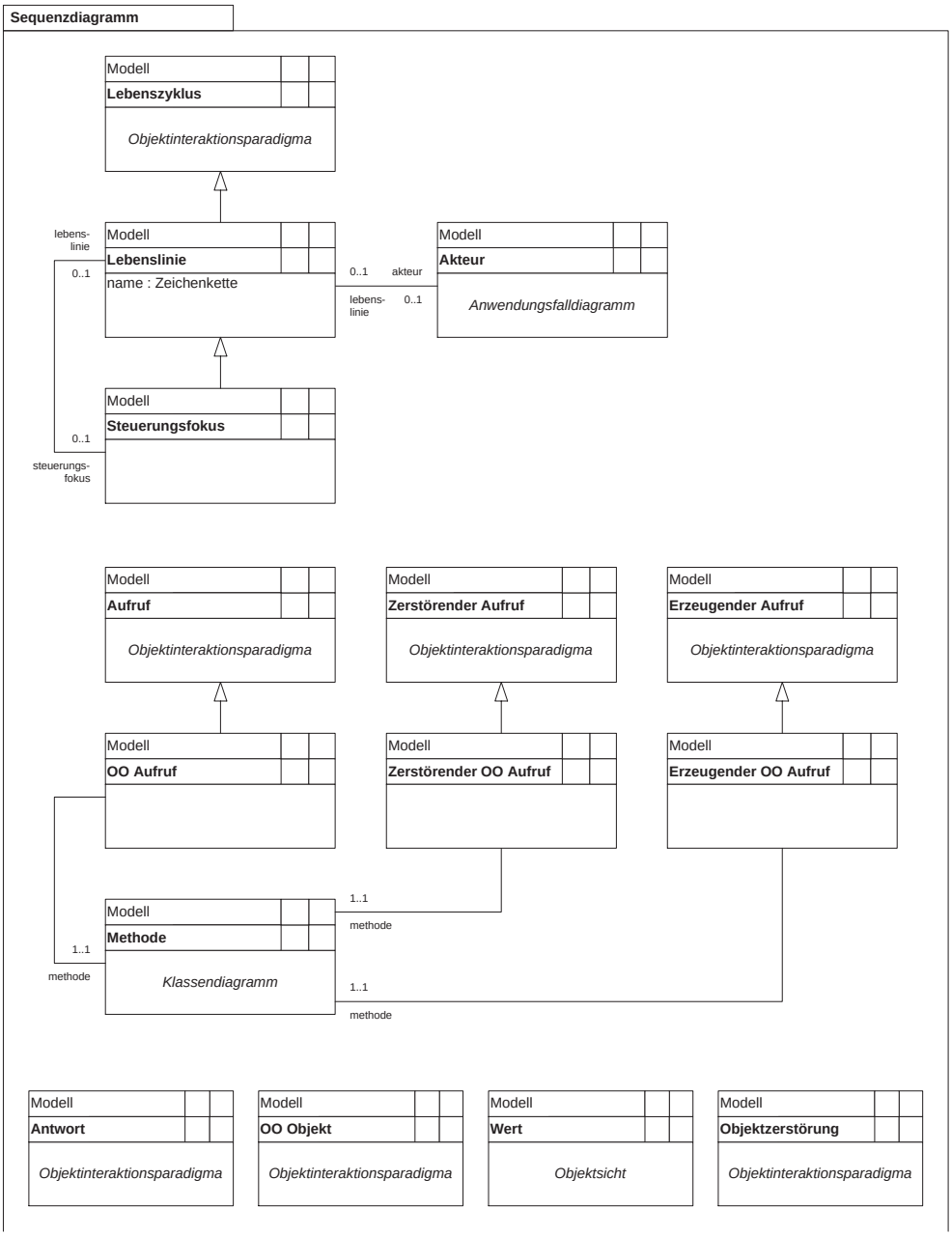

Klassendiagramm
Modell
Objektklasse
Objektbeziehungsparadigma
Modell
Attribut
Objektbeziehungsparadigma
Modell
Beziehungsklasse
Objektbeziehungsparadigma
Modell
OO Klasse
Modell
OO Attribut
Modell
Gerichtete Assoziation
oo attribut
0..n
1..1
eingehende Rolle
1..1
ausgehende Rolle
0..n
methode
beschränkung
1..1
0..1
zusicherung
Modell
Methode
name : Zeichenkette
Modell
Beschränkung
sichtbarkeit : Zeichenkette
Modell
Zusicherung
name : Zeichenkette
beschränkung
1..1
zusicherung
0..1
1..1
rückgabe-werttyp
Modell
Wertebereich
Modell
Ungerichtete Assoziation
Modell
OO Aggregation
Modell
Generalisierungsknoten
Objektbeziehungsparadigma
Modell
Spezialisierungslinie
Objektbeziehungsparadigma
Modell
OO Komposition


Modell
Klassendiagramm
Klassendiagramm
OO Klasse
OO Klasse
name
abstrakt
methode
beziehungsklasse
eingegangene beziehungsklasse
ausgegangene beziehungsklasse
generalisierungsknoten
spezialisierungslinie
oo attribut
Klassendiagramm
OO Klasse
OO Klasse verkürzt
name
abstrakt
beziehungsklasse
eingegangene beziehungsklasse
ausgegangene beziehungsklasse
generalisierungsknoten
spezialisierungslinie
Klassen-diagramm
Speziali-sierungslinie
Speziali-sierungslinie
generalisierungsknoten
generalisierbare objektklasse
Klassendiagramm
Generalisie-rungsknoten
Generalisie-rungsknoten
generalisierbare objektklasse
spezialisierungslinie
generalisierungskriterium
Klassendiagramm
OO Aggregation
OO Aggregation
name
eingehende objektklasse
ausgehende objektklasse
eingehende kardinalität
ausgehende kardinalität
eingehende rolle
ausgehende rolle
zusicherung
Klassendiagramm
Gerichtete Assoziation
Gerichtete Assoziation
name
eingehende objektklasse
ausgehende objektklasse
eingehende kardinalität
ausgehende kardinalität
eingehende rolle
ausgehende rolle
zusicherung
Klassendiagramm
Ungerichtete Assoziation
Ungerichtete Assoziation
name
objektklasse
eingehende kardinalität
ausgehende kardinalität
eingehende rolle
ausgehende rolle
zusicherung
zusätzlicher name
Klassendiagramm
OO Komposition
OO Komposition
name
eingehende objektklasse
ausgehende objektklasse
eingehende kardinalität
ausgehende kardinalität
eingehende rolle
ausgehende rolle
zusicherung
Klassen-diagramm
Klassendiagramm
Klassendiagramm
Klassen-diagramm
OO Klasse
OO Klasse verkürzt
Speziali-sierungslinie
Klassendiagramm
Generalisie-rungsknoten
Generalisie-rungsknoten
OO Aggregation
OO Aggregation
Gerichtete Assoziation
Gerichtete Assoziation
Ungerichtete Assoziation
Ungerichtete Assoziation
OO Komposition
OO Komposition

### 12.2.3.2 Objektinteraktionsparadigma




Ereignispfaddiagramm
Modell
Lebenszyklus
Objektinteraktionsparadigma
Modell
Nachricht
Objektinteraktionsparadigma
Modell
OO Objekt
Objektinteraktionsparadigma
Modell
Ereignispfaddiagramm
Ereignispfaddiagramm
Lebenszyklus
Lebenszyklus
oo objekt
gesendete nachricht
empfangene nachricht
Ereignispfaddiagramm
OO Objekt
OO Objekt
name
lebenszyklus
Ereignispfaddiagramm
Nachricht
Nachricht
name
sendender lebenszyklus
empfangender lebenszyklus
Ereignispfaddiagramm
Ereignispfaddiagramm
Ereignispfaddiagramm
Lebenszyklus
Lebenszyklus
oo objekt : Verbinder
gesendete nachricht : Verbinder
empfangene nachricht : Verbinder
Ereignispfaddiagramm
OO Objekt
OO Objekt
name : Beschriftung
lebenszyklus : Verbinder
Ereignispfaddiagramm
Nachricht
Nachricht
name : Beschriftung
sendender lebenszyklus : Verbinder
empfangender lebenszyklus : Verbinder


Sequenzdiagramm
Modell
Lebenszyklus
Objektinteraktionsparadigma
lebens-
linie
0..1
Modell
Lebenslinie
name : Zeichenkette
0..1
akteur
lebens-
linie
0..1
Modell
Akteur
Anwendungsfalldiagramm
Modell
Steuerungsfokus
0..1
steuerungs-
fokus
Modell
Aufruf
Objektinteraktionsparadigma
Modell
Zerstörender Aufruf
Objektinteraktionsparadigma
Modell
Erzeugender Aufruf
Objektinteraktionsparadigma
Modell
OO Aufruf
Modell
Zerstörender OO Aufruf
Modell
Erzeugender OO Aufruf
1..1
methode
Modell
Methode
1..1
methode
Klassendiagramm
1..1
methode
Modell
Antwort
Objektinteraktionsparadigma
Modell
OO Objekt
Objektinteraktionsparadigma
Modell
Wert
Objektsicht
Modell
Objektzerstörung
Objektinteraktionsparadigma

| Modell | **Sequenzdiagramm** | |
|---|---|---|
| | | |

| | Sequenzdiagramm | |
|---|---|---|
| Lebenslinie | **Lebenslinie** | |
| oo objekt<br>akteur<br>steuerungsfokus | | |

| | Sequenzdiagramm | |
|---|---|---|
| Steuerungsfokus | **Steuerungsfokus** | |
| lebenslinie<br>steuerungsfokus<br>gesendete nachricht<br>empfangene nachricht | | |

| | Sequenzdiagramm | |
|---|---|---|
| OO Objekt | **OO Objekt** | |
| lebenslinie | | |

| | Sequenzdiagramm | |
|---|---|---|
| Akteur | **Akteur** | |
| lebenslinie | | |

| | Sequenzdiagramm | |
|---|---|---|
| Objektzerstörung | **Objektzerstörung** | |
| lebenslinie<br>zerstörender aufruf | | |

| | Sequenzdiagramm | |
|---|---|---|
| Zerstörender OO Aufruf | **Zerstörender OO Aufruf** | |
| objektzerstörung<br>sendender lebenszyklus<br>methode | | |

| | Sequenzdiagramm | |
|---|---|---|
| Erzeugender OO Aufruf | **Erzeugender OO Aufruf** | |
| objektzerstörung<br>sendender lebenszyklus<br>methode | | |

| | Sequenzdiagramm | |
|---|---|---|
| OO Aufruf | **OO Aufruf** | |
| empfangender lebenszyklus<br>sendender lebenszyklus<br>methode | | |

| | Sequenzdiagramm | |
|---|---|---|
| Antwort | **Antwort** | |
| empfangender lebenszyklus<br>sendender lebenszyklus<br>wert | | |

| | Sequenzdiagramm | **Sequenzdiagramm** |
|---|---|---|
| | | |

| | | Sequenzdiagramm |
|---|---|---|
| | Lebenslinie | **Lebenslinie** |
| oo objekt : Verbinder<br>akteur : Verbinder<br>steuerungsfokus : Verbinder | | |

| | | Sequenzdiagramm |
|---|---|---|
| | Steuerungsfokus | **Steuerungsfokus** |
| lebenslinie : Verbinder<br>steuerungsfokus : Verbinder<br>gesendete nachricht : Verbinder<br>empfangene nachricht : Verbinder | | |

| | | Sequenzdiagramm |
|---|---|---|
| | OO Objekt | **OO Objekt** |
| lebenslinie : Verbinder | | |

| | | Sequenzdiagramm |
|---|---|---|
| | Akteur | **Akteur** |
| lebenslinie : Verbinder | | |

| | | Sequenzdiagramm |
|---|---|---|
| | Objektzerstörung | **Objektzerstörung** |
| lebenslinie : Verbinder<br>zerstörender aufruf : Verbinder | | |

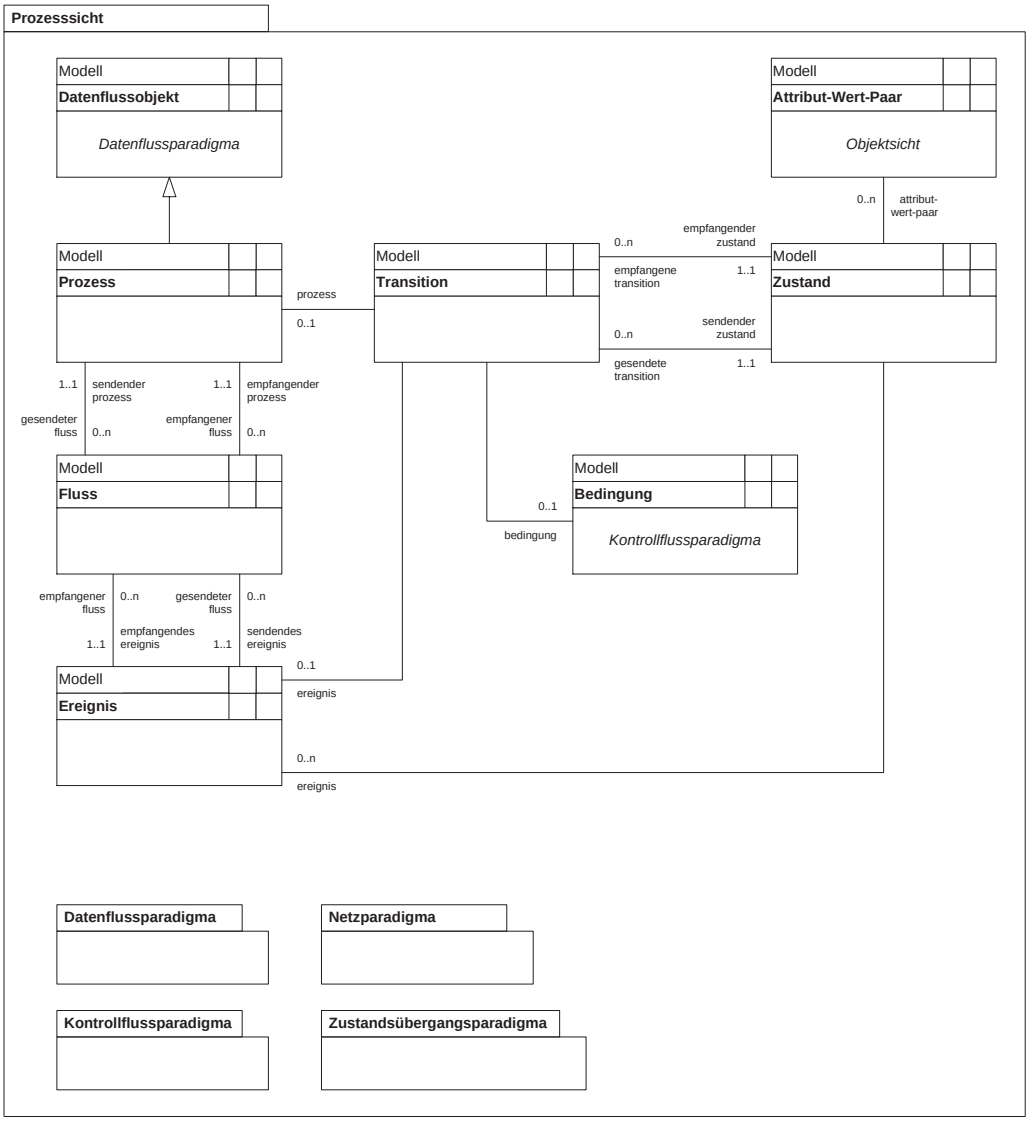
| | | Sequenzdiagramm |
|---|---|---|
| | Zerstörender OO Aufruf | **Zerstörender OO Aufruf** |
| methode : Verbinder<br>objektzerstörung : Verbinder<br>sendender lebenszyklus : Verbinder | | |

| | | Sequenzdiagramm |
|---|---|---|
| | Erzeugender OO Aufruf | **Erzeugender OO Aufruf** |
| methode : Verbinder<br>objektzerstörung : Verbinder<br>sendender lebenszyklus : Verbinder | | |

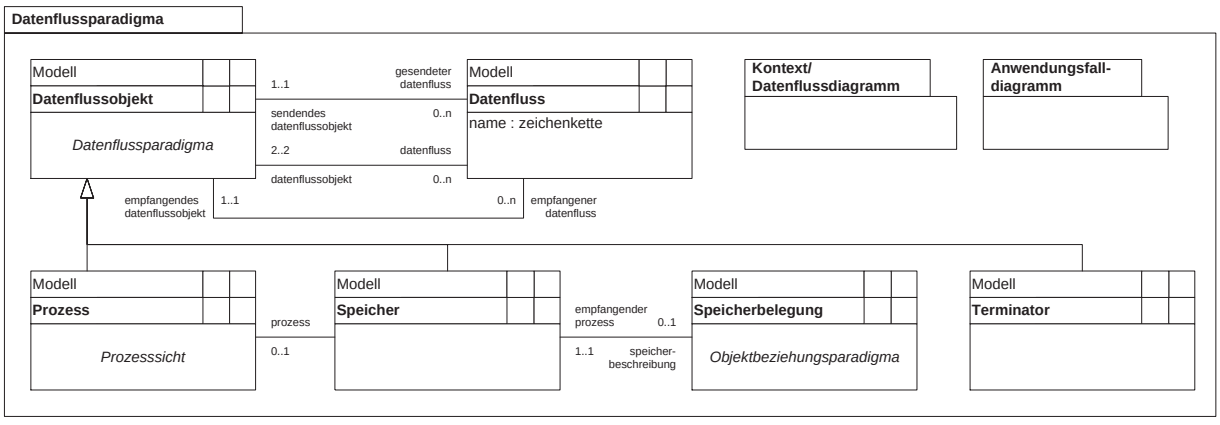
| | | Sequenzdiagramm |
|---|---|---|
| | OO Aufruf | **OO Aufruf** |
| methode : Verbinder<br>sendender lebenszyklus : Verbinder<br>empfangender lebenszyklus : Verbinder | | |

| | | Sequenzdiagramm |
|---|---|---|
| | Antwort | **Antwort** |
| wert : Beschriftung<br>sendender lebenszyklus : Verbinder<br>empfangender lebenszyklus : Verbinder | | |

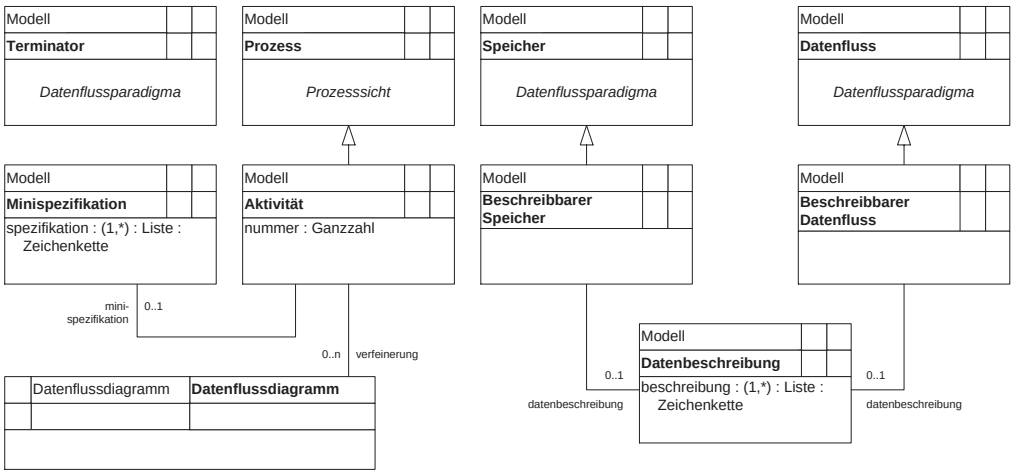
### 12.2.4 Prozesssicht

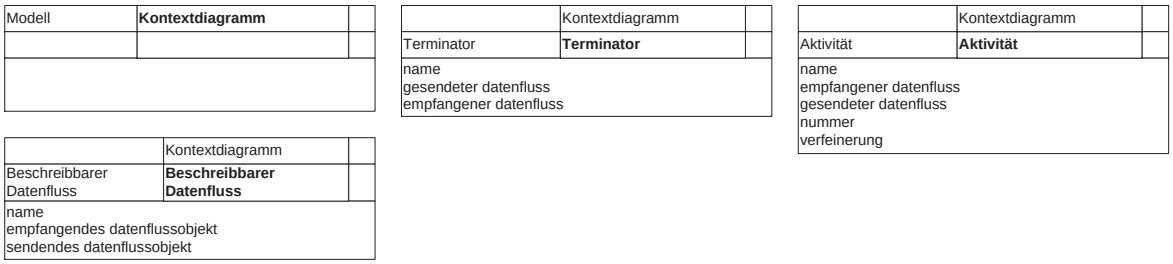


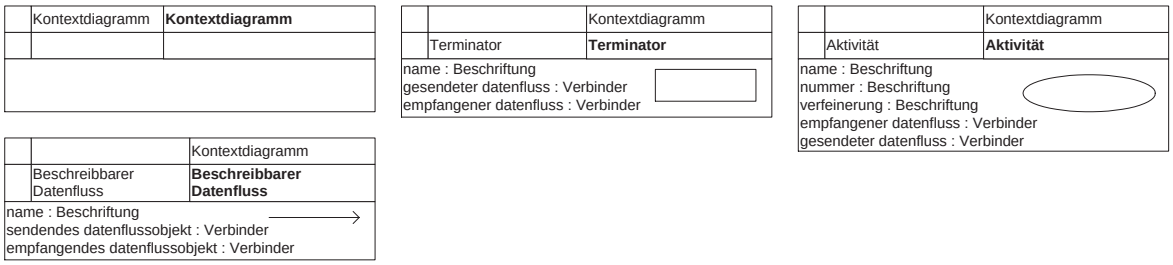

#### 12.2.4.1 Datenflussparadigma

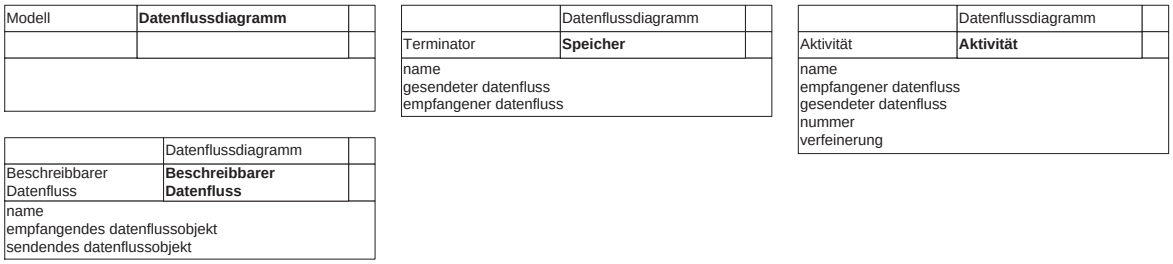


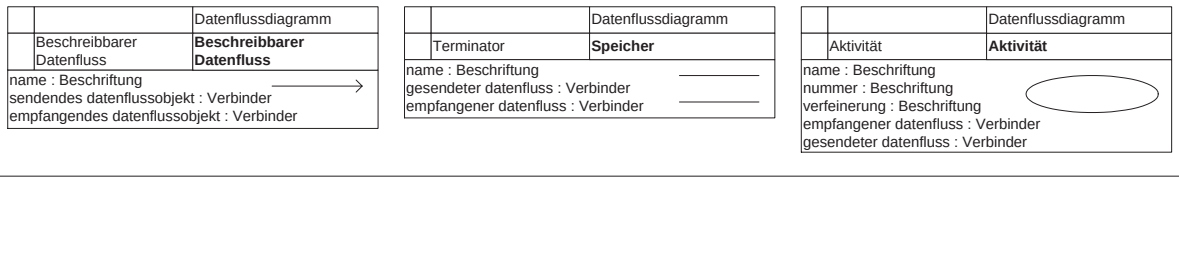
**Kontext-/Datenflussdiagramm**

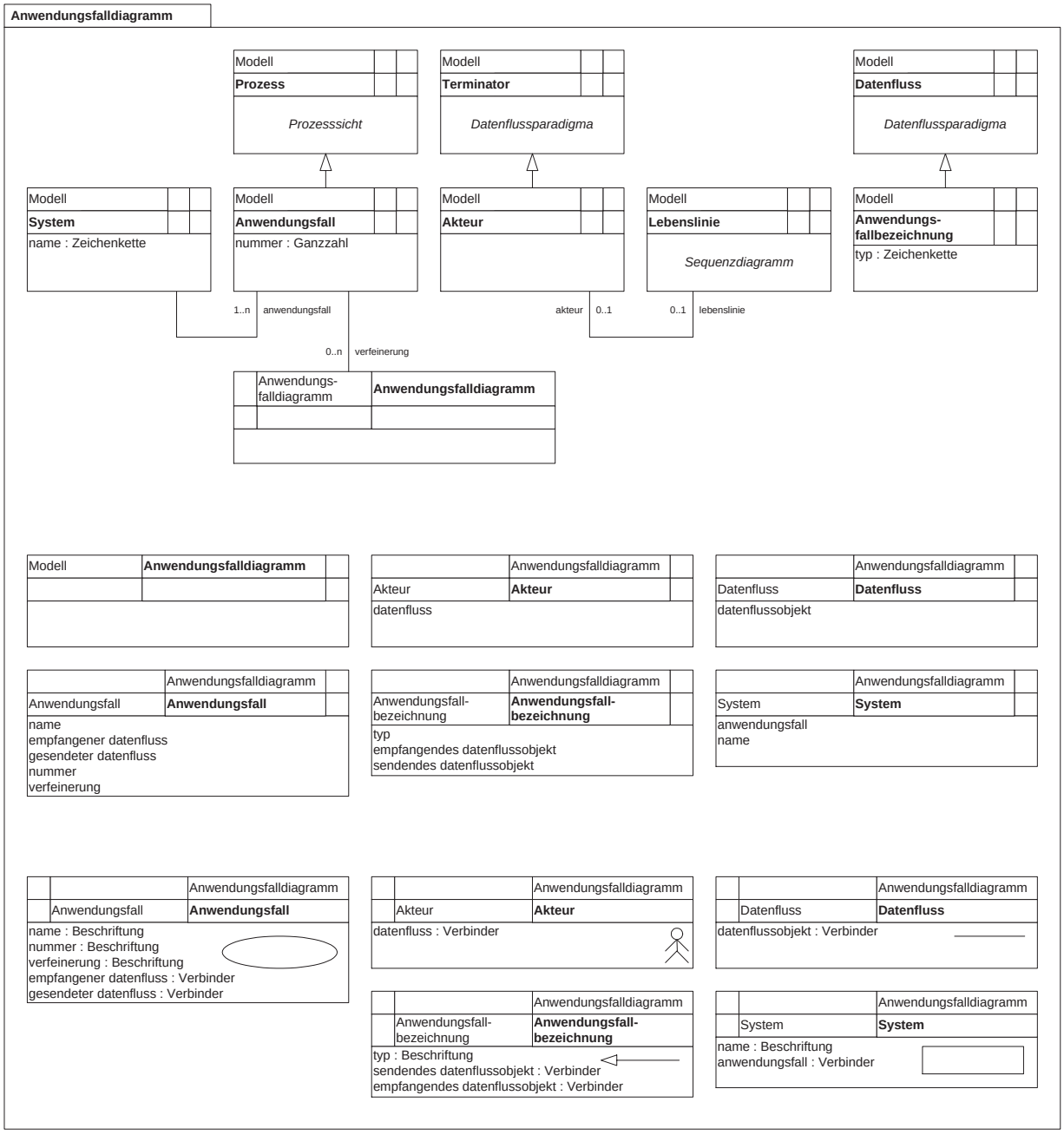


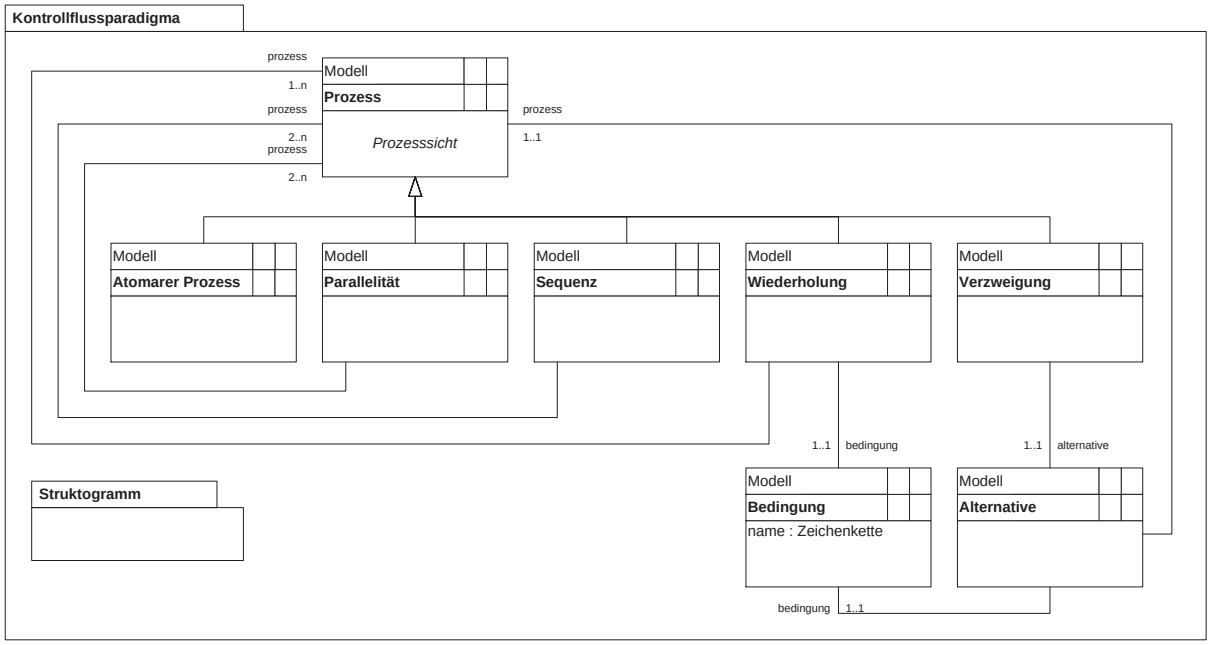



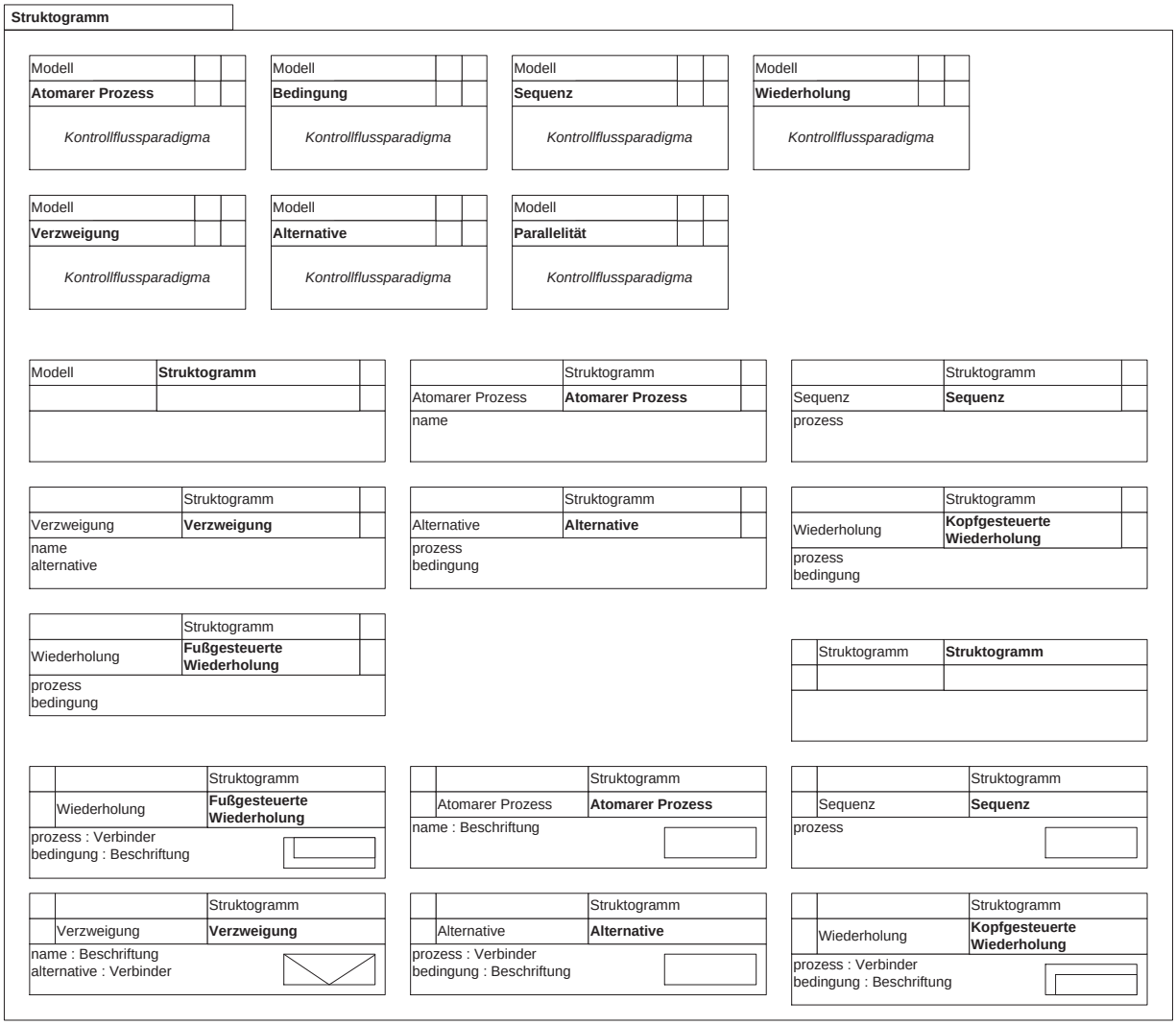



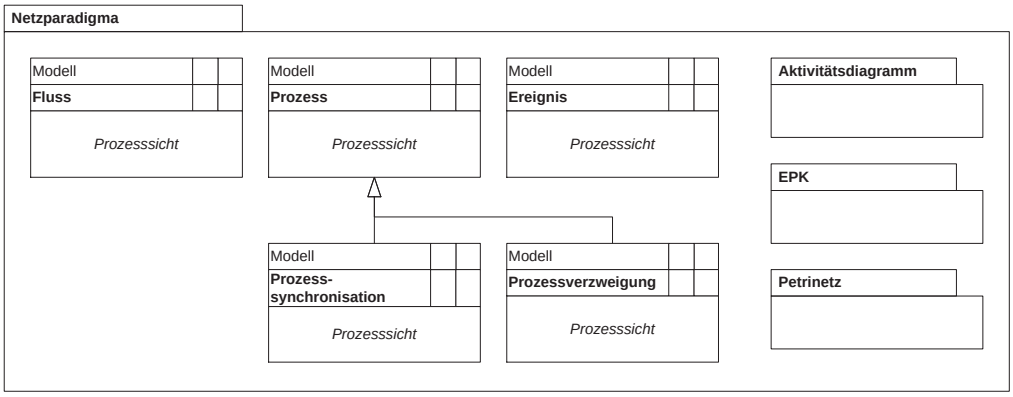

Anwendungsfalldiagramm
Modell
Prozess
Prozesssicht
Modell
Terminator
Datenflussparadigma
Modell
Datenfluss
Datenflussparadigma
Modell
System
name : Zeichenkette
Modell
Anwendungsfall
nummer : Ganzzahl
Modell
Akteur
Modell
Lebenslinie
Sequenzdiagramm
Modell
Anwendungs-
fallbezeichnung
typ : Zeichenkette
1..n
anwendungsfall
akteur
0..1
0..1
lebenslinie
0..n
verfeinerung
Anwendungs-
falldiagramm
Anwendungsfalldiagramm
Modell
Anwendungsfalldiagramm
Akteur
Akteur
datenfluss
Anwendungsfalldiagramm
Datenfluss
Datenfluss
datenflussobjekt
Anwendungsfalldiagramm
Anwendungsfall
Anwendungsfall
name
empfangener datenfluss
gesendeter datenfluss
nummer
verfeinerung
Anwendungsfalldiagramm
Anwendungsfall-
bezeichnung
Anwendungsfall-
bezeichnung
typ
empfangendes datenflussobjekt
sendendes datenflussobjekt
Anwendungsfalldiagramm
System
System
anwendungsfall
name
Anwendungsfalldiagramm
Anwendungsfall
Anwendungsfall
name : Beschriftung
nummer : Beschriftung
verfeinerung : Beschriftung
empfangener datenfluss : Verbinder
gesendeter datenfluss : Verbinder
Anwendungsfalldiagramm
Akteur
Akteur
datenfluss : Verbinder
Anwendungsfalldiagramm
Datenfluss
Datenfluss
datenflussobjekt : Verbinder
Anwendungsfalldiagramm
Anwendungsfall-
bezeichnung
Anwendungsfall-
bezeichnung
typ : Beschriftung
sendendes datenflussobjekt : Verbinder
empfangendes datenflussobjekt : Verbinder
Anwendungsfalldiagramm
System
System
name : Beschriftung
anwendungsfall : Verbinder

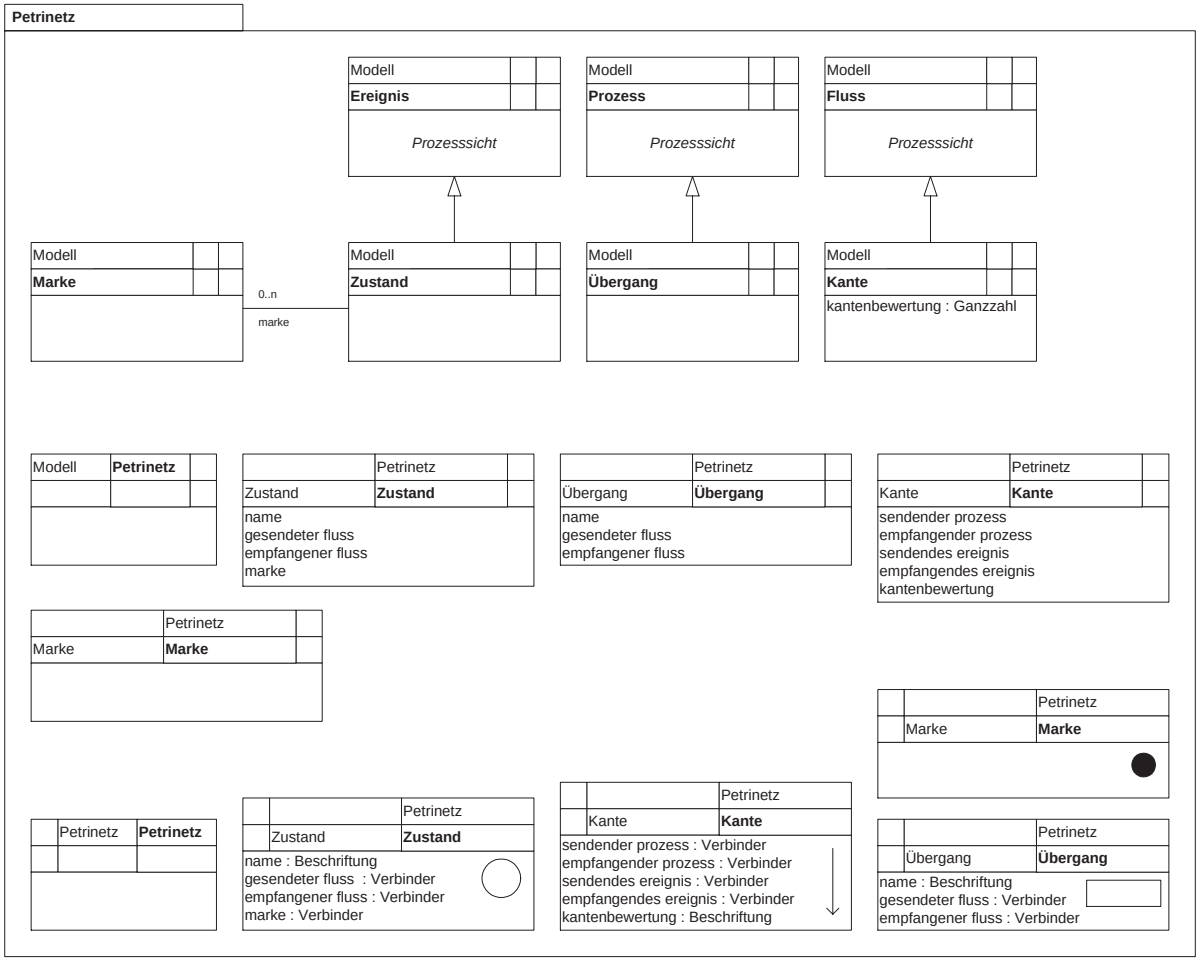
### 12.2.4.2 Kontrollflussparadigma



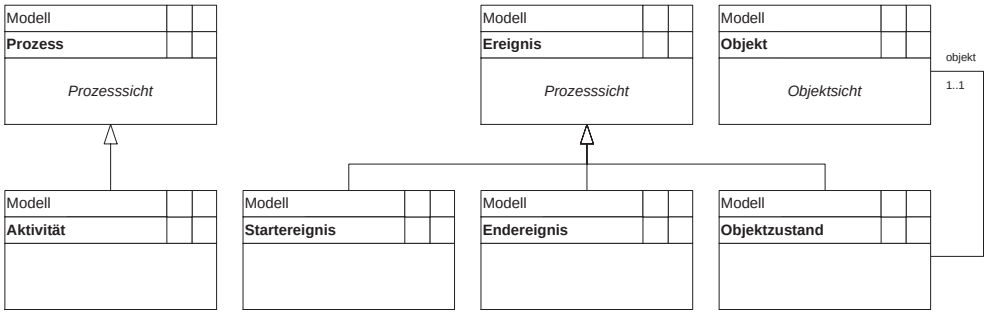

Struktogramm
Modell
Atomarer Prozess
Kontrollflussparadigma
Modell
Bedingung
Kontrollflussparadigma
Modell
Sequenz
Kontrollflussparadigma
Modell
Wiederholung
Kontrollflussparadigma
Modell
Verzweigung
Kontrollflussparadigma
Modell
Alternative
Kontrollflussparadigma
Modell
Parallelität
Kontrollflussparadigma
Modell
Struktogramm
Struktogramm
Atomarer Prozess
Atomarer Prozess
name
Struktogramm
Sequenz
Sequenz
prozess
Struktogramm
Verzweigung
Verzweigung
name
alternative
Struktogramm
Alternative
Alternative
prozess
bedingung
Struktogramm
Wiederholung
Kopfgesteuerte Wiederholung
prozess
bedingung
Struktogramm
Wiederholung
Fußgesteuerte Wiederholung
prozess
bedingung
Struktogramm
Struktogramm
Struktogramm
Wiederholung
Fußgesteuerte Wiederholung
prozess : Verbinder
bedingung : Beschriftung
Struktogramm
Atomarer Prozess
Atomarer Prozess
name : Beschriftung
Struktogramm
Sequenz
Sequenz
prozess
Struktogramm
Verzweigung
Verzweigung
name : Beschriftung
alternative : Verbinder
Struktogramm
Alternative
Alternative
prozess : Verbinder
bedingung : Beschriftung
Struktogramm
Wiederholung
Kopfgesteuerte Wiederholung
prozess : Verbinder
bedingung : Beschriftung

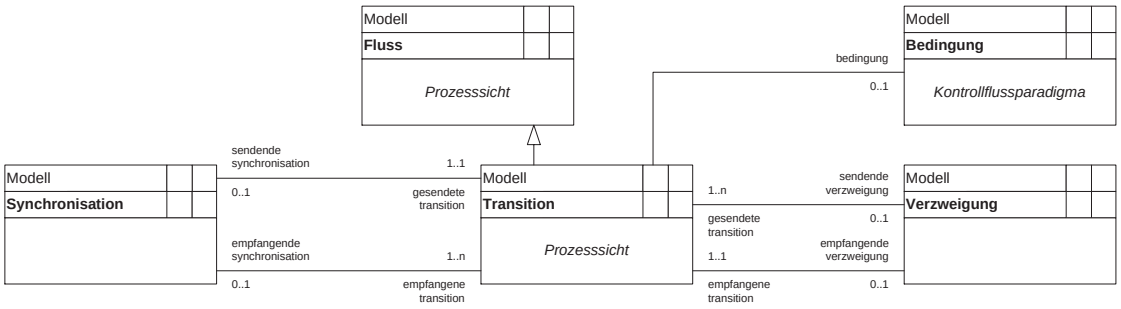
### 12.2.4.3 Netzparadigma



**Aktivitätsdiagramm**




| Modell | **Aktivitätsdiagramm** |
|---|---|

| | Aktivitätsdiagramm |
|---|---|
| Aktivität | **Aktivität** |
| name<br>gesendeter fluss<br>empfangener fluss | |

| | Aktivitätsdiagramm |
|---|---|
| Startereignis | **Startereignis** |
| gesendeter fluss | |

| | Aktivitätsdiagramm |
|---|---|
| Endereignis | **Endereignis** |
| empfangener fluss | |

| | Aktivitätsdiagramm |
|---|---|
| Verzweigung | **Verzweigung** |
| gesendete transition<br>empfangene transition | |

| | Aktivitätsdiagramm |
|---|---|
| Objektzustand | **Objektzustand** |
| name<br>objekt<br>gesendeter fluss<br>empfangener fluss | |

| | Aktivitätsdiagramm |
|---|---|
| Transition | **Transition** |
| bedingung<br>sendender prozess<br>empfangender prozess<br>sendende synchronisation<br>empfangende synchronisation<br>sendende verzweigung<br>empfangende verzweigung | |

| | Aktivitätsdiagramm |
|---|---|
| Transition | **Objekttransition** |
| bedingung<br>sendender prozess<br>empfangender prozess<br>sendende synchronisation<br>empfangende synchronisation | |

| | Aktivitätsdiagramm |
|---|---|
| Synchronisation | **Synchronisation** |
| gesendete transition<br>empfangene transition | |

| | Aktivitätsdiagramm |
|---|---|
| | **Aktivitätsdiagramm** |

| | Aktivitätsdiagramm |
|---|---|
| Aktivität | **Aktivität** |
| name : Beschriftung<br>gesendeter fluss : Verbinder<br>empfangener fluss : Verbinder | |

| | Aktivitätsdiagramm |
|---|---|
| Startereignis | **Startereignis** |
| gesendeter fluss : Verbinder | |

| | Aktivitätsdiagramm |
|---|---|
| Endereignis | **Endereignis** |
| empfangener fluss : Verbinder | |

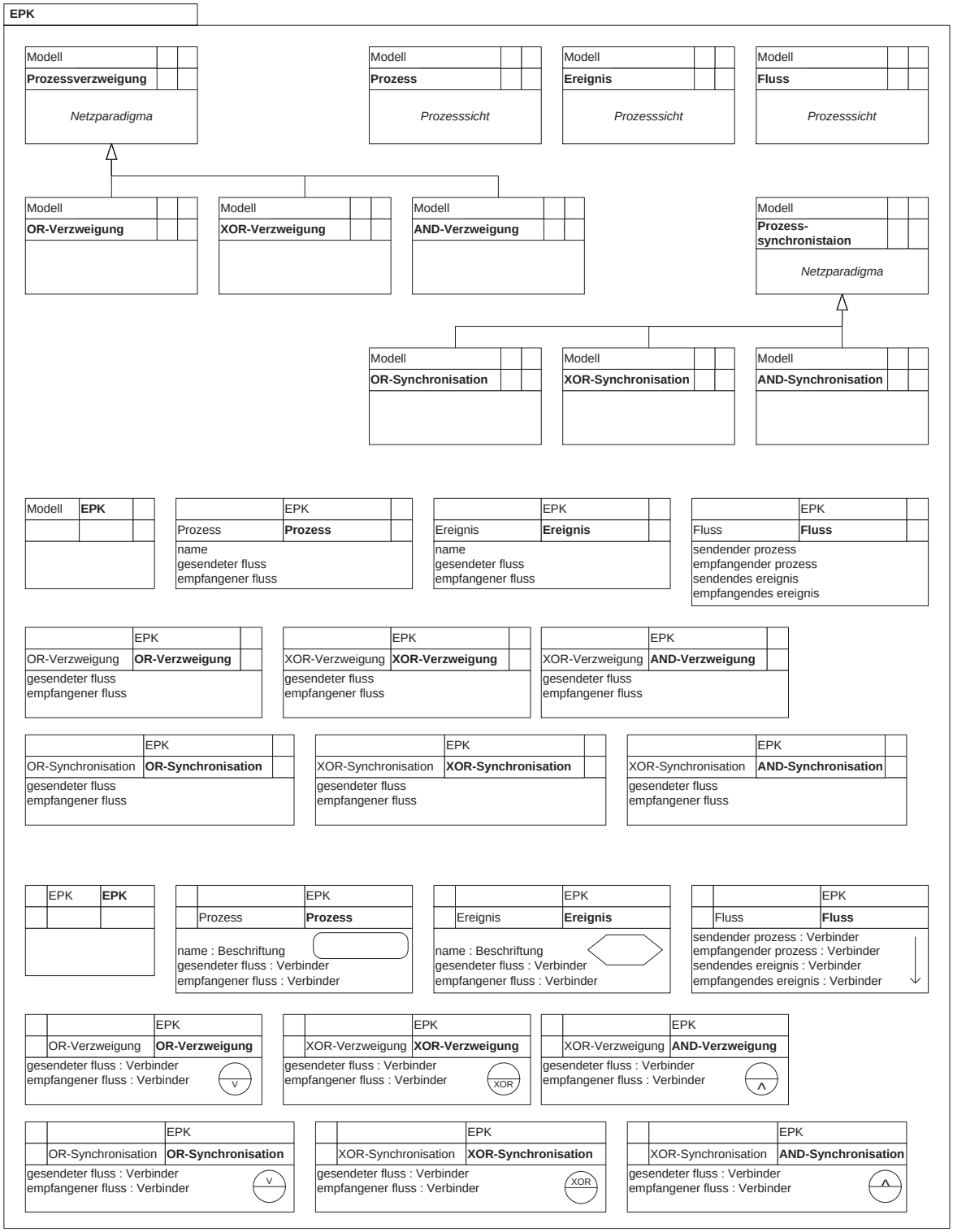
| | Aktivitätsdiagramm |
|---|---|
| Verzweigung | **Verzweigung** |
| gesendete transition : Verbinder<br>empfangene transition : Verbinder | |

| | Aktivitätsdiagramm |
|---|---|
| Objektzustand | **Objektzustand** |
| name : Beschriftung<br>objekt : Verbinder<br>gesendeter fluss : Verbinder<br>empfangener fluss : Verbinder | |

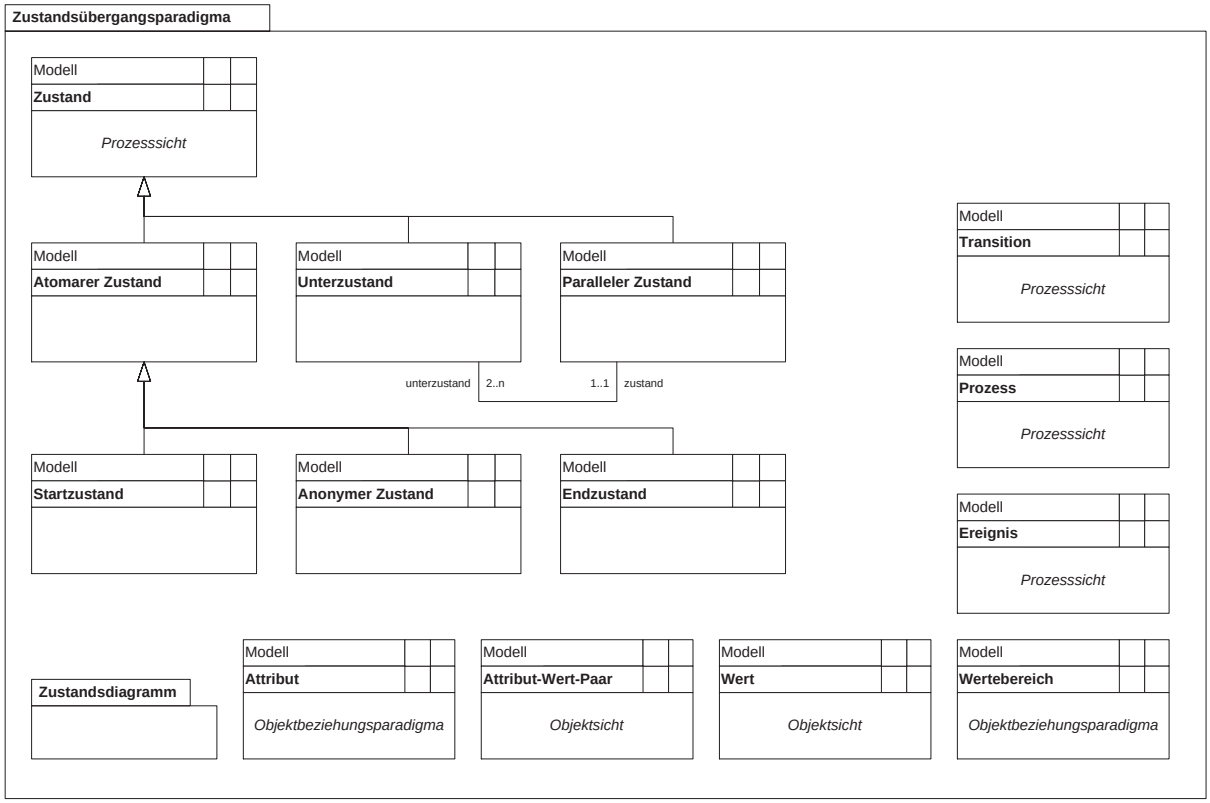
| | Aktivitätsdiagramm |
|---|---|
| Transition | **Transition** |
| bedingung: Beschriftung<br>sendender prozess : Verbinder<br>empfangender prozess : Verbinder<br>sendende synchronisation : Verbinder<br>empfangende synchronisation : Verbinder<br>sendende verzweigung : Verbinder<br>empfangende verzweigung : Verbinder | |

| | Aktivitätsdiagramm |
|---|---|
| Objekttransition | **Objekttransition** |
| bedingung : Beschriftung<br>sendender prozess : Verbinder<br>empfangender prozess : Verbinder<br>sendende synchronisation : Verbinder<br>empfangende synchronisation : Verbinder | |

| | Aktivitätsdiagramm |
|---|---|
| Synchronisation | **Synchronisation** |
| gesendete transition : Verbinder<br>empfangene transition : Verbinder | |


EPK
Modell
Prozessverzweigung
Netzparadigma
Modell
Prozess
Prozesssicht
Modell
Ereignis
Prozesssicht
Modell
Fluss
Prozesssicht
Modell
OR-Verzweigung
Modell
XOR-Verzweigung
Modell
AND-Verzweigung
Modell
Prozess-synchronistaion
Netzparadigma
Modell
OR-Synchronisation
Modell
XOR-Synchronisation
Modell
AND-Synchronisation
Modell EPK
Prozess Prozess
name
gesendeter fluss
empfangener fluss
Ereignis Ereignis
name
gesendeter fluss
empfangener fluss
Fluss Fluss
sendender prozess
empfangender prozess
sendendes ereignis
empfangendes ereignis
OR-Verzweigung OR-Verzweigung
gesendeter fluss
empfangener fluss
XOR-Verzweigung XOR-Verzweigung
gesendeter fluss
empfangener fluss
XOR-Verzweigung AND-Verzweigung
gesendeter fluss
empfangener fluss
OR-Synchronisation OR-Synchronisation
gesendeter fluss
empfangener fluss
XOR-Synchronisation XOR-Synchronisation
gesendeter fluss
empfangener fluss
XOR-Synchronisation AND-Synchronisation
gesendeter fluss
empfangener fluss
EPK EPK
Prozess Prozess
name : Beschriftung
gesendeter fluss : Verbinder
empfangener fluss : Verbinder
Ereignis Ereignis
name : Beschriftung
gesendeter fluss : Verbinder
empfangener fluss : Verbinder
Fluss Fluss
sendender prozess : Verbinder
empfangender prozess : Verbinder
sendendes ereignis : Verbinder
empfangendes ereignis : Verbinder
OR-Verzweigung OR-Verzweigung
gesendeter fluss : Verbinder
empfangener fluss : Verbinder
V
XOR-Verzweigung XOR-Verzweigung
gesendeter fluss : Verbinder
empfangener fluss : Verbinder
XOR
XOR-Verzweigung AND-Verzweigung
gesendeter fluss : Verbinder
empfangener fluss : Verbinder
OR-Synchronisation OR-Synchronisation
gesendeter fluss : Verbinder
empfangener fluss : Verbinder
V
XOR-Synchronisation XOR-Synchronisation
gesendeter fluss : Verbinder
empfangener fluss : Verbinder
XOR
XOR-Synchronisation AND-Synchronisation
gesendeter fluss : Verbinder
empfangener fluss : Verbinder

### 12.2.4.4 Zustandsübergangsparadigma



**Zustandsdiagramm**

| Modell |
|---|
| **Startzustand** |
| *Zustandsübergangsparadigma* |

| Modell |
|---|
| **Endzustand** |
| *Zustandsübergangsparadigma* |

| Modell |
|---|
| **Zustand** |
| *Prozesssicht* |

| Modell |
|---|
| **Anonymer Zustand** |
| *Zustandsübergangsparadigma* |

| Modell |
|---|
| **Unterzustand** |
| *Zustandsübergangsparadigma* |

| Modell |
|---|
| **Paralleler Zustand** |
| *Zustandsübergangsparadigma* |

| Modell |
|---|
| **Transition** |
| *Prozesssicht* |

| Modell |
|---|
| **Prozess** |
| *Prozesssicht* |

| Modell |
|---|
| **Attribut-Wert-Paar** |
| *Objektsicht* |

| Modell |
|---|
| **Attribut** |
| *Objektbeziehungsparadigma* |

| Modell |
|---|
| **Ereignis** |
| *Prozesssicht* |

| Modell |
|---|
| **Wertebereich** |
| *Objektbeziehungsparadigma* |

| Modell | **Zustandsdiagramm** |
|---|---|

| | Zustandsdiagramm |
|---|---|
| Startzustand | **Startzustand** |

gesendete transition

| | Zustandsdiagramm |
|---|---|
| Endzustand | **Endzustand** |

empfangene transition

| | Zustandsdiagramm |
|---|---|
| Zustand | **Zustand** |

empfangene transition
gesendete transition
name
attribut-wert-paar

| | Zustandsdiagramm |
|---|---|
| Unterzustand | **Unterzustand** |

empfangene transition
gesendete transition
name
attribut-wert-paar
zustand

| | Zustandsdiagramm |
|---|---|
| Zustand | **Erweiterter Zustand** |

empfangene transition
gesendete transition
name
attribut-wert-paar
ereignis

| | Zustandsdiagramm |
|---|---|
| Anonymer Zustand | **Anonymer Zustand** |

empfangene transition
gesendete transition

| | Zustandsdiagramm |
|---|---|
| Paralleler Zustand | **Paralleler Zustand** |

empfangene transition
gesendete transition
name
attribut-wert-paar
unterzustand

| | Zustandsdiagramm |
|---|---|
| Paralleler Zustand | **Erweiterter Paralleler Zustand** |

empfangene transition
gesendete transition
name
attribut-wert-paar
unterzustand
ereignis

| | Zustandsdiagramm |
|---|---|
| Unterzustand | **Erweiterter Unterzustand** |

empfangene transition
gesendete transition
name
attribut-wert-paar
zustand
ereignis

| | Zustandsdiagramm | **Zustandsdiagramm** |
|---|---|---|

| | Zustandsdiagramm |
|---|---|
| Startzustand | **Startzustand** |

gesendete transition : Verbinder

| | Zustandsdiagramm |
|---|---|
| Zustand | **Zustand** |

name : Beschriftung
attribut-wert-paar : Verbinder
gesendete transition : Verbinder
empfangene transition : Verbinder

| | Zustandsdiagramm |
|---|---|
| Unterzustand | **Unterzustand** |

name : Beschriftung
zustand : Verbinder
attribut-wert-paar : Verbinder
gesendete transition : Verbinder
empfangene transition : Verbinder

| | Zustandsdiagramm |
|---|---|
| Anonymer Zustand | **Anonymer Zustand** |

gesendete transition : Verbinder
empfangene transition : Verbinder

| | Zustandsdiagramm |
|---|---|
| Paralleler Zustand | **Paralleler Zustand** |

name : Beschriftung
unterzustand : Verbinder
attribut-wert-paar : Verbinder
empfangene transition : Verbinder
gesendete transition : Verbinder

| | Zustandsdiagramm |
|---|---|
| Endzustand | **Endzustand** |

empfangene transition : Verbinder

| | Zustandsdiagramm |
|---|---|
| Erweiterter Paralleler Zustand | **Erweiterter Paralleler Zustand** |

name : Beschriftung
ereignis : Beschriftung
unterzustand : Verbinder
attribut-wert-paar : Verbinder
gesendete transition : Verbinder
empfangene transition : Verbinder

| | Zustandsdiagramm |
|---|---|
| Erweiterter Unterzustand | **Erweiterter Unterzustand** |

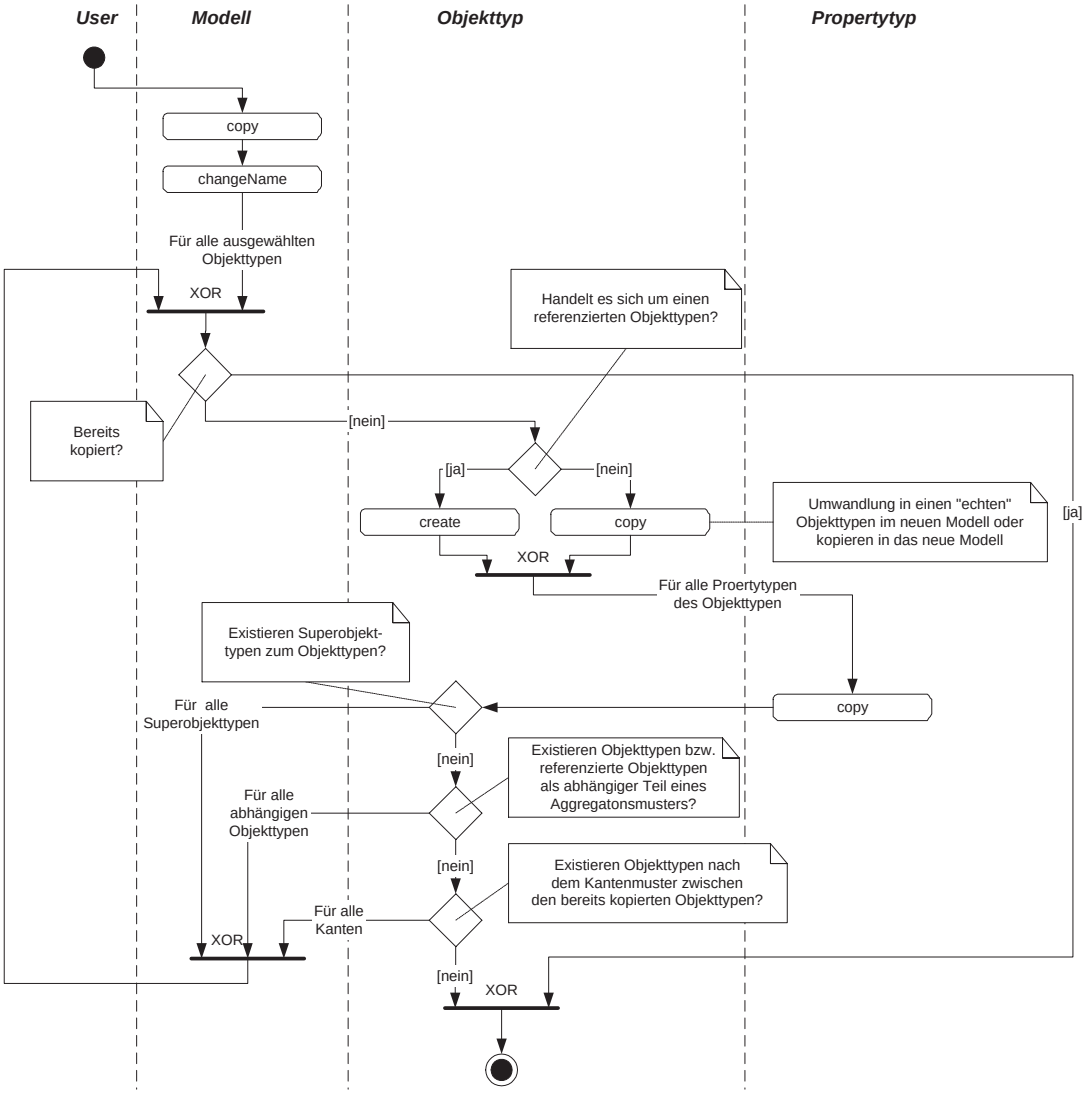
name : Beschriftung
zustand : Verbinder
ereignis : Beschriftung
empfangene transition : Verbinder
gesendete transition : Verbinder
attribut-wert-paar : Beschriftung

| | Zustandsdiagramm |
|---|---|
| Erweiterter Zustand | **Erweiterter Zustand** |

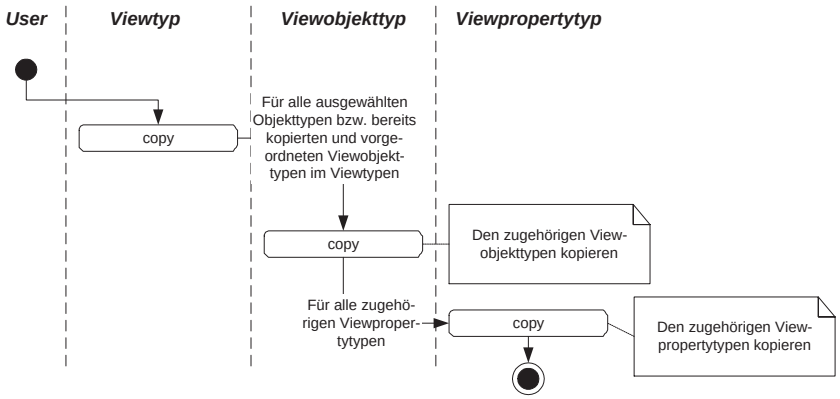
name : Beschriftung
ereignis : Beschriftung
attribut-wert-paar : Verbinder
gesendete transition : Verbinder
empfangene transition : Verbinder

## 12.3 Dynamische Sicht

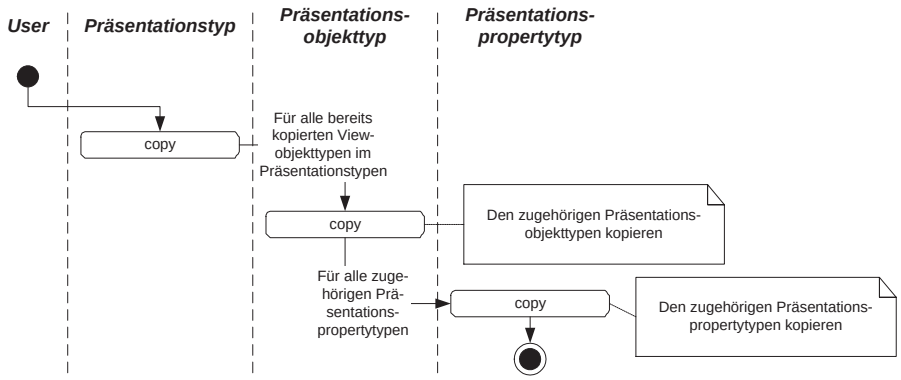
### Aktivitätsdiagramm zum Kopieren eines Modells



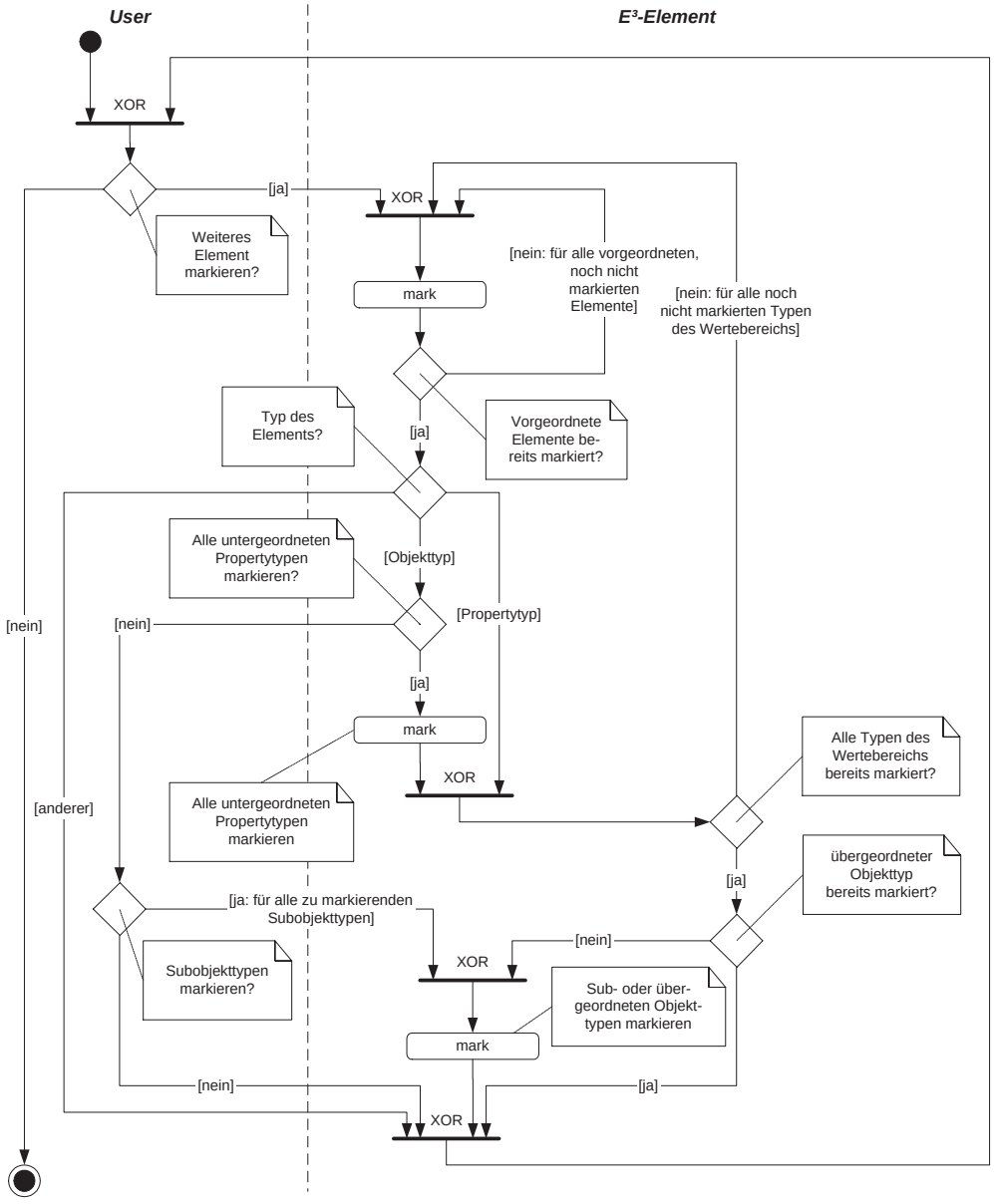

### Aktivitätsdiagramm zum Kopieren eines Viewtypen



## Aktivitätsdiagramm zum Kopieren eines Präsentationstypen




## Aktivitätsdiagramm zum Markieren der Elemente



## 12.4 Fachbegriffsmodell des Referenzmetamodells der E$^3$-Methode

| Objekttyp | Beschreibung |
|---|---|
| 0,* Kante | Spezielle **Beziehungsklasse** mit der Multiplizität (0,*). |
| 0,1 Kante | Spezielle **Beziehungsklasse** mit der Multiplizität (0,1). |
| 1,* Kante | Spezielle **Beziehungsklasse** mit der Multiplizität (1,*). |
| 1,1 Kante | Spezielle **Beziehungsklasse** mit der Multiplizität (1,1). |
| Abteilung | Vereinigung mehrerer **Stellen** unter einheitlicher Leitung (vgl. [Wei$^+$92], S. 52). |
| Aggregation | Zusammenfassung von untereinander in Beziehung stehenden **E-Typen** zu einem aggregierten **E-Typ** (vgl. [FeSi01], S. 139). |
| Akteur | Spezialform des **Terminators**. Ein Akteur kann Ausgangspunkt einer **Lebenslinie** sein. |
| Aktivität[1] | Spezialform eines **Prozesses**. Transformiert eingehende in ausgehende **Datenflüsse**. Eine Aktivität[1] kann verfeinert werden. Nicht mehr zerlegbare Aktivitäten[1] werden in einer **Minispezifikation** beschrieben. |
| Aktivität[2] | Spezialform eines **Prozesses**. Einzelner Schritt in einem Ablauf, Zustand mit einer internen Aktion und einer oder mehreren ausgehenden **Transitionen** (vgl. [Oest01], S. 319). |
| Alternative | Eine Alternative besteht genau aus einem **Prozess**, der immer dann abgearbeitet wird, wenn die zugeordnete **Bedingung** erfüllt ist. |
| AND-Synchronisation | Die nachfolgende Prozessabarbeitung kann erst dann beginnen, wenn alle eingehenden **Prozesse** abgearbeitet sind. |
| AND-Verzweigung | Nach einer UND Prozessverzweigung werden alle folgenden **Prozesse** abgearbeitet. |
| Anonymer Zustand | **Zustand**[1] ohne Namen (vgl. [Oest01], S. 306). |
| Antwort | Spezialform einer **Nachricht**, die die Reaktion auf einen **Aufruf** darstellt. Eine Antwort kann mit einem **Wert** verknüpft werden. |
| Anwendungsfall | Spezialform des **Prozesses**, der in einem Anwendungsfalldiagramm verfeinert werden kann. Anwendungsfälle werden zu einem **System** zusammengefasst. |
| Anwendungsfallbeziehung | Spezialform eines **Datenflusses**, die eine Beziehung zwischen **Anwendungsfällen** darstellt. Diese Beziehung kann die Ausprägungen ,,erweitert" oder ,,genutzt" annehmen (vgl. [Oest01], S. 207). |
| Assistenz | Zuordnung einer **Stabstelle** zu einer **Leitungsstelle** |
| Atomarer Prozess | Nicht weiter zerlegbarer **Prozess**. |
| Atomarer Zustand | Nicht weiter zerlegbarer **Zustand**[1]. |
| Attribut | Eigenschaft einer **Klasse**, die dieser über eine **Attributzuordnung** zugeordnet wird. Jedes Attribut hat einen **Wertebereich**. |
| Attribut-Wert-Paar | Aggregation aus einem **Wert** und einem **Attribut**. |
| Attributzuordnung | Zuordnung eines **Attributs** zu einer **Klasse** |

| **Objekttyp** | **Beschreibung** |
|---|---|
| Aufgabe | Zielsetzung einer zweckbezogenen menschlichen Handlung (vgl. [Wei $^+$92], S. 287). Eine Aufgabe wird einem **Prozess** zugeordnet, der diese Aufgabe ausführt. Über eine **Aufgabenzuordnung** werden Aufgaben mit **Organisationseinheiten** verknüpft. Eine Aufgabe ist ebenfalls einer **Objektklasse** zugewiesen. |
| Aufgabengliederung | Zerlegung von **Aufgaben** in Unteraufgaben. |
| Aufgabengliederung Ausführungsaufgabe | **Aufgabengliederung** nach der (Ausführungs-) Phase. |
| Aufgabengliederung Kontrollaufgabe | **Aufgabengliederung** nach der (Kontroll-) Phase. |
| Aufgabengliederung Leitungsaufgabe | **Aufgabengliederung** nach dem Rang (Leistung). |
| Aufgabengliederung Planungsaufgabe | **Aufgabengliederung** nach der (Planungs-) Phase. |
| Aufgabengliederung Verwaltungsaufgabe | **Aufgabengliederung** nach dem Zweck (Verwaltung). |
| Aufgabenzuordnung | Zuordnung einer **Aufgabe** zu einer **Organisationseinheit**. |
| Aufruf | Spezialform einer **Nachricht**, die ein **OO Objekt** anweist, eine bestimmte Aktion auszuführen. Einem Aufruf wird eine **Methode** zugeordnet. Ein Aufruf wird mit einer **Antwort** abgeschlossen. |
| Ausführungsstelle | Eine **Stelle** ohne Leitungsbefugnisse. |
| Bedingung | Ausdruck der einen Wahrheitswert (wahr oder falsch) liefert. |
| Beschränkung | Durch eine Beschränkung wird die Sichtbarkeit von **OO Attributen** und **Methoden** festgelegt (private, protected, public). Zusätzlich kann eine **Zusicherung** zugewiesen werden. |
| Beschreibbarer Datenfluss | Spezialform eines **Datenflusses**, der sowohl zerlegt als auch durch eine **Datenbeschreibung** beschrieben werden kann. |
| Beschreibbarer Speicher | Spezialform eines **Speichers**, der zusätzlich durch eine **Datenbeschreibung** näher erläutert wird. |
| Beziehung | Spezialisierung einer **Instanz**. Eine Beziehung stellt Verbindungen zwischen **Objekten** her. |
| Beziehungsklasse | Spezialform einer **Klasse**, die gleichartige **Beziehungen** beschreibt und in Beziehung zu einer **Objektklasse** stehen kann. |
| Datenbeschreibung | Synonym: Data Dictionary (vgl. [DeMa79], S. 125ff). Beschreibung eines **beschreibbaren Datenflusses**, **beschreibbaren Speichers** und von **Aktivitäten**[1]. |
| Datenfluss | Pipeline, durch welche Pakete von Informationen fließen (vgl. [DeMa79], S. 52). |
| Datenflussobjekt | Generalisierung eines **Speichers**, eines **Terminators** und eines **Prozesses**. |
| Disjunkte Generalisierung | Spezialform einer **Generalisierung**, bei der keine Überlappung der Objektmengen auftritt (vgl. [FeSi01], S. 151). |
| Dokument | Spezielles **Kommunikationsmedium**. Die Kommunikation erfolgt in Papierform, wobei das Dokument nicht elektronisch zugestellt wird. |

| Objekttyp | Beschreibung |
|---|---|
| E-Typ | Spezialform der **erweiterten generalisierbaren Objektklasse**, die innerhalb des ERM gleichartige Entitäten zusammenfasst (vgl. [FeSi01], S. 130). |
| eMail | Spezialform eines **Kommunikationsmediums**, bei dem die Nachricht ausschließlich auf elektronischer Basis übermittelt wird. |
| Endereignis | Spezialform eines **Ereignisses**, welches nur **Flüsse** empfangen kann. |
| Endzustand | Spezialform eines **atomaren Zustandes**, wobei ausschließlich eingehende **Transitionen**[1] zugelassen sind. |
| ER-Linie | Spezielle Beziehungsklasse, die die Zuordnung eines **E-Typs** zu einem **R-Typen** erlaubt. |
| ER-Typ | Spezieller **E-Typ**, der aus einer Kombination eines **R-Typs** und eines **E-Typs** entsteht, die durch eine (1,1)-Beziehung verbunden sind (vgl. [FeSi01], S. 145f). |
| Ereignis | Auslöser und Ergebnis eines **Prozesses**. Ereignisse werden durch **Flüsse** mit **Prozessen** verknüpft. |
| Erweiterte generalisierbare Objektklasse | Spezialform der **generalisierten Objektklasse**, wobei die Zuordnung eines **Generalisierungsknotens** ebenfalls durch ein Kantenmuster abgebildet wird. |
| erweiterte Kommunikation | Spezialform einer **Kommunikation**, der zusätzlich das **Kommunikationsmedium** zugeordnet wird. |
| Erweiterte Spezialisierungslinie | Spezialform einer **Spezialisierungslinie**, die eine (zusätzliche) Verbindung mit einem **E-Typen** ermöglicht. Damit können Subtypenhierarchien aufgebaut werden. |
| Erweiterter Generalisierungsknoten | Spezialform eines **Generalisierungsknotens**, wobei der Teil der Generalisierung ebenfalls durch ein Kantenmuster (Kante **Generalisierungslinie**) abgebildet wird. |
| Erzeugender Aufruf | Spezialform eines **Aufrufs**, mit dem ein **OO Objekt** angelegt wird. |
| Erzeugender OO Aufruf | Spezialform des **erzeugenden Aufrufs**. Ein erzeugender OO Aufruf korrespondiert zusätzlich mit einer **Methode**. |
| Fax | Spezielles **Kommunikationsmedium**. Die **Kommunikation** erfolgt in Papierform, wobei die Übermittlung des **Dokumentes** elektronisch verläuft und dabei der Empfänger eine Kopie des Originals erhält. |
| Fluss | Verknüpfung von **Prozessen** und **Ereignissen**. |
| Generalisierbare Objektklasse | Spezialform einer **Objektklasse**. Eine generalisierbare Objektklasse kann zusätzlich generalisiert bzw. spezialisiert werden. |
| Generalisierungshierarchie | Ein **E-Typ** E wird als Generalisierung von $E_1$–$E_n$ bezeichnet, falls jedes Entity von E genau ein Entity der **E-Typen** $E_1$–$E_n$ ist (vgl. [FeSi01], S. 151f). Spezialform eines **erweiterten Generalisierungsknotens**. |
| Generalisierungsknoten | Zusammen mit der **Spezialisierungslinie** bildet ein Generalisierungsknoten eine Generalisierungs- bzw. Spezialisierungsstruktur, die einer **generalisierbaren Objektklasse** zugeordnet werden kann. |
| Generalisierungslinie | Zuordnung einer **erweiterten generalisierbaren Objektklasse** zu einem erweiterten Generalisierungsknoten. |

| **Objekttyp** | **Beschreibung** |
|---|---|
| Gerichtete Assoziation | Spezialform einer **Beziehungsklasse**, bei der die Beziehung nur in eine Richtung navigierbar ist. |
| Geschäftspartner | Externer Kommunikationspartner. |
| Gliederbare Abteilung | Spezialform einer **Abteilung**. Die Beziehung zwischen **Abteilungen** wird dabei als Kantenmuster zusammen mit dem Objekttypen **Gliederung** abgebildet. |
| Gliederung | Zerlegung von **gliederbaren Abteilungen** in Form eines Kantenmusters. |
| Gruppe | Stellenmehrheit, die dann entsteht, wenn mehrere Stellen zusammengefasst werden (vgl. [Wei$^+$92], S. 52). Spezialform einer **Stelle**. |
| Instanz | Ausprägung einer **Klasse**. |
| Kante | Spezialform eines **Flusses**, dem eine Kantenbewertung hinzugefügt wird. |
| Klasse | Zusammenfassung gleichartiger **Instanzen**, die in **Beziehungsklassen** und **Objektklassen** verfeinert werden. Einer Klasse können mit einer **Attributzuordnung Attribute** (Eigenschaften) zugeordnet werden. |
| Kommunikation | Beschreibung des Nachrichtenaustausches zwischen **Organisationseinheiten** und/oder **Geschäftspartnern**. |
| Kommunikationsmedium | Art und Weise der Übertragung einer **Kommunikation**. |
| Lebenslinie | Spezialform des **Lebenszykluses**. Eine Lebenslinie kann zusätzlich einem **Akteur** zugeordnet werden. |
| Lebenszyklus | Der Lebenszyklus zeigt die Dauer der Existenz eines **OO Objektes** an. Nur innerhalb dieses Lebenszykluses kann ein **OO Objekt Nachrichten** senden und empfangen. |
| Leitung | Zuordnung einer **Leitungsstelle** zu einer **Nichtstabstelle**. |
| Leitungsstelle | Spezialform einer **Stelle**. Eine Leitungsstelle ist gegenüber anderen **Nichtstabstellen** weisungsbefugt. |
| Marke | Marken werden innerhalb eines Petrinetzes zu **Zuständen**[2] zugeordnet. |
| Methode | Eine Methode ist eine Sequenz von Anweisungen. Sie definiert eine Operation einer **OO Klasse**. |
| Minispezifikation | Beschreibung einer nicht weiter zerlegbaren **Aktivität**[1]. |
| Nachricht | Aufforderung eines Senders an einen Empfänger eine Dienstleistung zu erbringen. Nachrichten dienen der **Kommunikation**. Sie werden über ein **Kommunikationsmedium** übertragen (vgl. [Balz99], S. 536). |
| Nicht disjunkte Generalisierung | Spezialform einer **Generalisierung**, bei der eine Überlappung der Objektmengen zugelassen wird (vgl. [FeSi01], S. 151). |
| Nichtstabstelle | Spezialform einer **Stelle**, Synonym: Linienstelle. |
| Objekt | Spezialform einer **Instanz**, die eine gedankliche Abstraktion oder reale Entität darstellt. |
| Objektklasse | Spezialform einer **Klasse**, die gleichartige Objekte beschreibt. Eine Objektklasse kann mit einer **Beziehungsklasse** in Verbindung treten. |
| Objektzerstörung | Zuordnung eines **zerstörenden Aufrufs** zu einem **Lebenszyklus**. |
| Objektzustand | Spezialform eines **Ereignisses**, welches mit einem **Objekt** verknüpft ist. |
| OR-Synchronisation | Spezialisierung einer **Prozesssynchronisation**, bei der der **Prozess** weiterläuft, sobald Prozess 1 und/oder Prozess 2 beendet ist. |

| **Objekttyp** | **Beschreibung** |
|---|---|
| OR-Verzweigung | Spezialisierung einer **Prozessverzweigung**, bei der entweder der nachfolgende Prozess 1 und/oder Prozess 2 abgearbeitet wird. |
| OO Aggregation | Spezialfall einer **gerichteten Assoziation**, bei der die beteiligten Elemente Bestandteil einer Teil-Ganzes Beziehung sind (vgl. [Oest01], S. 279). |
| OO Attribut | Spezielles **Attribut**, welchem eine **Beschränkung** zugeordnet wird und innerhalb einer **gerichteten Assoziation** als Rolle verwendet werden kann. |
| OO Aufruf | Spezieller **Aufruf**, dem eine **Methode** zugeordnet wird. |
| OO Klasse | Spezielle **Objektklasse**. Eine OO Klasse ist die Definition der **OO Attribute**, **Methoden** und Semantik für eine Menge von **OO Objekten** (vgl. [Oest01], S. 209). |
| OO Komposition | Spezialfall einer **OO Aggregation**, bei der die Teile vom Ganzen existenzabhängig sind (vgl. [Oest01], S. 250). |
| OO Objekt | Spezialform eines **Objektes**, welches einen **Lebenszyklus** hat und durch einen **erzeugenden Aufruf** geschaffen wird. |
| Organisationseinheit | Innerbetrieblicher Kommunikationspartner. |
| Paralleler Zustand | Zusammenfassung von mindestens zwei **Unterzuständen**, die gleichzeitig abgearbeitet werden. |
| Parallelität | Spezialisierung eines **Prozesses**. Mindestens zwei **Prozesse**, die gleichzeitig ausgeführt werden. |
| Prozess | Zeitlich und sachlogisch strukturierte Kette von Aktivitäten zur Bearbeitung eines betriebswirtschaftlich relevanten Objektes. |
| Prozesssynchronisation | Zusammenführung von einem oder mehreren **Prozessen**. |
| Prozessverzweigung | Aufteilung eines **Prozesses** in einen oder mehrere andere **Prozesse**. |
| R-Typ | Spezielle **Objektklasse**, die gleichartige Beziehungen zusammenfasst (vgl. [FeSi01], S. 130). |
| Sequenz | Anreihung von mindestens zwei **Prozessen**. |
| Speicher | Spezielles **Datenflussobjekt**; Zwischenzeitliches Datenrepository (vgl. [DeMa79], S. 57), welches durch eine **Speicherbeschreibung** näher beschrieben wird. |
| Speicherbeschreibung | Zuordnung einer **Objektklasse** zu einem **Speicher**. |
| Spezialisierungslinie | Zusammen mit dem **Generalisierungsknoten** bildet eine Spezialisierungslinie eine Generalisierungs- bzw. Spezialisierungsstruktur, die einer **generalisierbaren Objektklasse** zugeordnet werden kann. |
| Stabstelle | Eine Stabstelle ist eine spezielle **Stelle**, die ihre Aufgaben von der ihr zugeordneten und sie unterstützenden Leitungsstelle ableitet (vgl. [Wei$^+$92], S. 79). |
| Startereignis | Spezialform eines **Ereignisses**, welches ausschließlich Flüsse aussenden kann. |
| Startzustand | Spezialform eines **atomaren Zustandes**, der lediglich ausgehende **Transitionen**[1] erlaubt. |
| Stelle | Vereinigung von (Teil)**Aufgaben** und ihre Übertragung auf einen Aufgabenträger (vgl. [Wei$^+$92], S. 43). |

| **Objekttyp** | **Beschreibung** |
|---|---|
| Steuerungsfokus | Der Steuerungsfokus gibt an, welches **OO Objekt** gerade die Programmkontrolle besitzt (vgl. [Oest01], S. 299). Er wird einer **Lebenslinie** zugeordnet. |
| Synchronisation | Zusammenführung von **Transitionen**$^2$ innerhalb eines Aktivitätsdiagramms. |
| System | Zusammenfassung mehrerer **Anwendungsfälle**. |
| Telefon | Spezielles **Kommunikationsmedium** bei dem die **Kommunikation** zwischen den Kommunikationspartnern mündlich unter Zuhilfenahme elektronischer Übertragungstechniken erfolgt. |
| Terminator | Spezielles **Datenflussobjekt**. Quelle oder Senke von **Datenflüssen**. |
| Transition$^1$ | Überführung zwischen zwei **Zuständen**$^1$. Eine Transition$^1$ kann an eine **Bedingung** gekoppelt sein. |
| Transition$^2$ | Spezialisierung von **Fluss**. Eine Transition$^2$ kann sowohl verzweigt (**Verzweigung**$^2$) als auch synchronisiert (**Synchronisation**) werden. Ihr kann eine **Bedingung** zugeordnet werden. |
| Übergang | Spezialform eines **Prozesses** innerhalb des Petrinetzes. |
| Ungerichtete Assoziation | Spezialform der **gerichteten Assoziation**, bei der die Assoziation in beide Richtungen navigierbar ist. |
| Unterzustand | Spezialform eines **Zustandes**$^1$. Es werden sequentielle **Zustände**$^1$ zusammengefasst. |
| Unvollständige Generalisierungslinie | Die Vereinigung der Objektmengen der Subtypen entspricht nicht der Objektmenge des Supertypen (vgl. [FeSi01], S. 151). |
| Verzweigung$^1$ | Spezialform eines **Prozesses**. Eine Verzweigung$^1$ spaltet einen **Prozess** in mindestens eine **Alternative** auf. Es wird nur derjenige **Prozess** ausgeführt, für den die der **Alternative** zugeordnete **Bedingung** erfüllt ist. |
| Verzweigung$^2$ | Aufspaltung von **Transitionen**$^2$. |
| Vollständige Generalisierungslinie | Die Vereinigung der Objektmengen der Subtypen ist identisch mit der Objektmenge des Supertypen (vgl. [FeSi01], S. 151). |
| Wert | Ausprägung eines **Wertebereichs**. |
| Wertebereich | Menge von **Werten** eines bestimmten Typs. |
| Wiederholung | Spezialisierung eines **Prozesses**, der abhängig von einer **Bedingung** mehrfach ausgeführt wird. |
| XOR-Synchronisation | Spezialform der **Prozesssynchronisation**, bei der der Prozess fortgesetzt wird, wenn entweder Prozess 1 oder Prozess 2 beendet ist. |
| XOR-Verzweigung | Spezialform der **Prozessverzweigung**, bei der entweder der nachfolgende Prozess 1 oder Prozess 2 abgearbeitet wird. |
| Zerstörender Aufruf | Spezialform eines **Aufrufs**. Ein zerstörender Aufruf beendet die Existenz eines **OO Objektes**. |
| Zerstörender OO Aufruf | Spezialform eines **zerstörenden Aufrufs**, der zusätzlich mit einer **Methode** korrespondiert. |
| Zusicherung | Ausdruck, der die möglichen Inhalte, Zustände oder die Semantik eines Elementes einschränkt (vgl. [Oest01], S. 233). |
| Zustand$^1$ | Zeitspanne zwischen dem Eintreten von zwei **Transitionen**$^1$. |
| Zustand$^2$ | Spezialisierung eines **Ereignisses**, dem **Marken** zugeordnet werden können. |

# 13 Bewertung von Methoden

## 13.1 Das Bunge-Wand-Weber-Modell ([GrRo00], S. 75f)

| **Ontological Construct** | **Explanation** |
|---|---|
| thing* | A thing is the elementary unit in the BWW ontological model. The real world is made up of things. Two or more things (composite or simple) can be associated into a composite thing. |
| property*:<br>in general<br>in particular<br>hereditary<br>emergent<br>intrinsic<br>non-binding mutual<br>binding mutual<br>attributes | Things possess properties. A property is modelled via a function that maps the thing into some value. For example, the attribute „weight“ represents a property that all humans possess. In this regard, weight is an attribute standing for a property in general. If we focus on the weight of a specific individual, however, we would be concerned with a property in particular. A property of a composite thing that belongs to a component thing is called an hereditary property. Otherwise it is called an emergent property. Some properties are inherent properties of individual things. Such properties are called intrinsic. Other properties are properties of pairs or many things. Such properties are called mutual. Non-binding mutual properties are those properties shared by two or more things that do not „make a difference“ to the things involved; for example, order relations or equivalence relations. By contrast, binding mutual properties are those properties shared by two or more things that do „make a difference“ to the things involved. Attributes are the names that we use to represent properties of things. |
| class | A class is a set of things that can be defined via their possessing a single property. |
| kind | A kind is a set of things that can be defined only via their possessing two or more common properties. |
| state* | The vector of values for all property functions of a thing is the state of the thing. |
| conceivable state space | The set of all states that the thing might ever assume is the conceivable state space of the thing. |
| state law:<br>stability condition<br>corrective action | A state law restricts the values of the properties of a thing to a subset that is deemed lawful because of natural laws or human laws. The stability condition specifies the states allowed by the state law. The corrective action specifies how the value of the property function must change to provide a state acceptable under the state law. |
| lawful state space | The lawful state space is the set of states of a thing that comply with the state laws of the thing. The lawful state space is usually a proper subset of the conceivable state space. |
| conceivable event space | The event space of a thing is the set of all possible events that can occur in the thing. |
| transformation* | A transformation is a mapping from one state to another state. |

| **Ontological Construct** | **Explanation** |
|---|---|
| lawful transformation:<br>stability condition<br>corrective action | A lawful transformation defines which events in a thing are lawful. The stability condition specifies the states that are allowable under the transformation law. The corrective action specifies how the values of the property function(s) must change to provide a state acceptable under the transformation law. |
| lawful event space | The lawful event space is the set of all events in a thing that are lawful. |
| history | The chronologically-ordered states that a thing traverses in time are the history of the thing. |
| acts on | A thing acts on another thing if its existence affects the history of the other thing. |
| coupling:<br>binding mutual property | Two things are said to be coupled (or interact) if one thing acts on the other. Furthermore, those two things are said to share a binding mutual property (or relation); that is, they participate in a relation that „makes a difference“ to the things. |
| system | A set of things is a system if, for any bi-partitioning of the set, couplings exist among things in the two subsets. |
| system composition | The things in the system are its composition. |
| system environment | Things that are not in the system but interact with things in the system are called the environment of the system. |
| system structure | The set of couplings that exist among things within the system, and among things in the environment of the system and things in the system is called the structure. |
| subsystem | A subsystem is a system whose composition and structure are subsets of the composition and structure of another system. |
| system decomposition | A decomposition of a system is a set of subsystems such that every component in the system is either one of the subsystems in the decomposition or is included in the composition of one of the subsystems in the decomposition. |
| level structure | A level structure defines a partial order over the subsystems in a decomposition to show which subsystems are components of other subsystems or the system itself. |
| external event | An external event is an event that arises in a thing, subsystem, or system by virtue of the action of some thing in the environment on the thing, subsystem, or system. |
| stable state* | A stable state is a state in which a thing, subsystem, or system will remain unless forced to change by virtue of the action of a thing in the environment (an external event). |
| unstable state | An unstable state is a state that will be changed into another state by virtue of the action of transformations in the system. |
| internal event | An internal event is an event that arises in a thing, subsystem, or system by virtue of lawful transformations in the thing, subsystem, or system. |
| well-defined event | A well-defined event is an event in which the subsequent state can always be predicted given that the prior state is known. |
| poorly defined event | A poorly-defined event is an event in which the subsequent state cannot be predicted given that the prior state is known. |

## 13.2 Qualität von Methoden

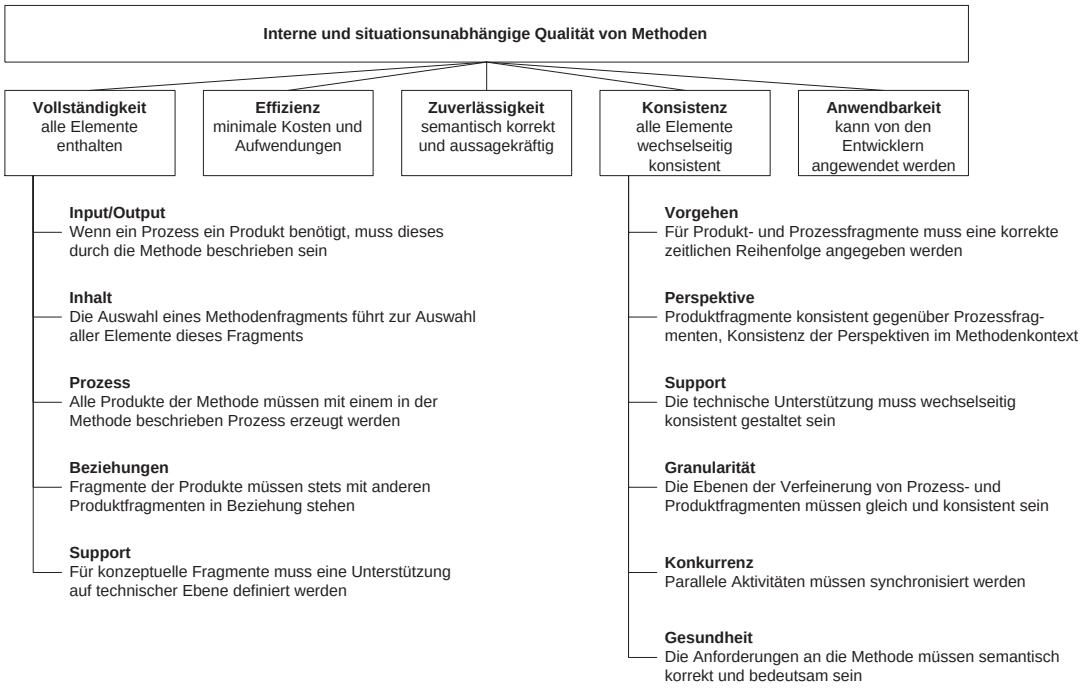



Abbildung 83: Interne und Situationsabhängige Qualität von Methoden nach Brinkkemper (zusammengefasst aus [Bri+99], S. 299ff)

## 13.3 Quality Function Deployment

### 13.3.1 Effizienz von Instrumenten der Produktentwicklung

| Effizienzkriterium / Instrument | Ermöglichung der Planung, Steuerung und Kontrolle von Kunden-zufriedenheit | Gewährleistung der Einhaltung der technischen Spezifikation | Förderung der differenzierten Kundenbedürfnis-befriedigung | Unterstützung der Prozeßqualität |
|---|---|---|---|---|
| Requirements Engineering Techniken | ○ | ● | ○ | ○ |
| Sieben Management- / Planungstechniken | ○ | ○ | ○ | ○ |
| Erhebungstechniken | ● | ○ | ● | ○ |
| Qualitätszirkel | ○ | ○ | | ○ |
| Kreativtechniken | | | ○ | ○ |
| Gewichtungs- / Priorisierungstechniken | ○ | ○ | ○ | |
| Wertanalyse | ○ | ● | ○ | ● |
| Quality Function Deployment (QFD) | ● | ● | ● | ● |
| Failure Mode and Effect Analysis (FMEA) | | ○ | | ○ |
| Statistische Prozeßkontrolle (SPC) | | ● | | ○ |
| Design-Review | | ● | | ○ |
| Design of Experiments (DoE) | | ● | | ● |
| Kundenzufriedenheits-messungen | ● | ○ | ● | ○ |
| Qualitäts-Audits | ○ | ● | | ● |
| Benchmarking | ● | | ● | ● |

Abbildung 84: Effizienz von Instrumenten der Produktentwicklung ([Herz00], S. 204)

### 13.3.2 Korrelationen im Matrix-Diagramm

Das folgende Beispiel ist [Her+00], S. 22f entnommen und zeigt eine Priorisierungsmatrix, wie sie z. B. im HoQ Anwendung findet. Die Zeilen $x_i(i = 1, ..., n)$ und Spalten $y_i(i = 1, ..., m)$ gestatten das Eintragen von Korrelationswerten $w_{ij}(i = 1, ..., n; j = 1, ..., m)$, die angeben, wie stark die Ausprägung von $y_j$ die Ausprägung von $x_i$ beeinflusst. Die absolute Gewichtung berechnet sich in jeder Zeile nach folgender Formel:

$$aw(y_i) = \sum_{i=1}^{n} rw(x_i) * w_{ij}$$

Die relative Gewichtung jeder Spalte berechnet sich wie folgt:

$$rw(y_i) = \frac{aw(y_i)}{\sum_{j=1}^{m} aw(y_j)}$$

| | $y_1$ | $y_2$ | $y_3$ | relative weight $rw(x_i)$ |
|---|---|---|---|---|
| $x_1$ | $w_{11}$=1 | $w_{12}$=9 | $w_{13}$=3 | 0,4 |
| $x_2$ | $w_{21}$=9 | $w_{22}$=0 | $w_{23}$=1 | 0,3 |
| $x_3$ | $w_{31}$=0 | $w_{32}$=3 | $w_{33}$=3 | 0,2 |
| $x_4$ | $w_{41}$=3 | $w_{42}$=0 | $w_{43}$=9 | 0,1 |
| absolute weight $aw(y_i)$ | 3,4 | 4,2 | 3,0 | $\sum$ 10,6 |
| relative weight $rw(y_i)$ | 0,32 | 0,4 | 0,28 | $\sum$ 1 |

### 13.3.3 Matrixdegeneration und Gegenmaßnahmen

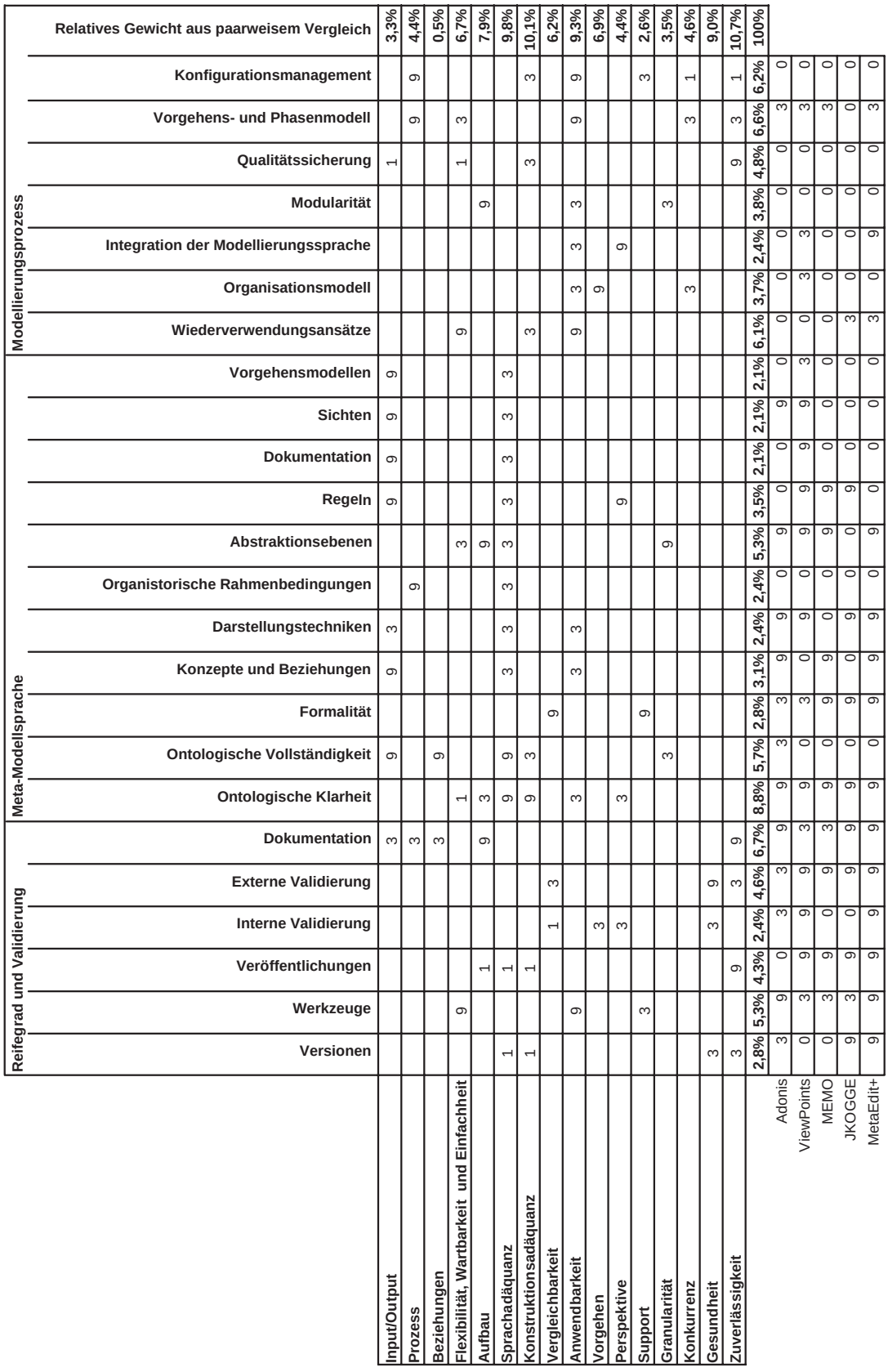
| Art der Matrixdegeneration | Gegenmaßnahmen |
|---|---|
| Leer bzw. im Verhältnis zu ihrer Bewertung zu schwache Zeilen | Merkmal zur Abdeckung der Anforderung entwickeln |
| Im Verhältnis zu ihrer Bewertung überproportional starke Zeile | Anforderung ggf. präzisieren |
| Leere Spalten | Merkmal überflüssig oder fehlende Anforderung |
| Gleiche Spalten | Eventuell zusammenlegen, wenn die Merkmale nur unterschiedlichen Erfüllungsgrad eines Merkmals widerspiegeln |
| Starke Spalten mit vielen Korrelationen | Merkmal präzisieren |
| Viele schwache Beziehungen bzw. weniger als 15% Korrelationen | Merkmale klarer formulieren |
| (Fast) Diagonalmatrix mit vielen starken (1:1) Beziehungen | Merkmale bzw. Anforderung überprüfen |

Tabelle 19: Matrixdegeneration und Gegenmaßnahmen (vgl. [Her$^+$00], S. 262)

## 13.4 Paarweiser Vergleich der Anforderungen an Methoden

| | Input/Output | Prozess | Beziehungen | Flexibilität, Wartbarkeit und Einfachheit | Aufbau | Sprachadäquanz | Konstruktionsadäquanz | Vergleichbarkeit | Anwendbarkeit | Vorgehen | Perspektive | Support | Granularität | Konkurrenz | Gesundheit | Zuverlässigkeit | Relatives Gewicht |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
| **Input/Output** | 1,00 | 0,33 | 9,00 | 0,33 | 0,11 | 0,11 | 0,11 | 0,11 | 0,11 | 0,11 | 0,11 | 0,33 | 1,00 | 0,33 | 0,11 | 0,11 | **3,3%** |
| **Prozess** | 3,00 | 1,00 | 9,00 | 0,33 | 1,00 | 0,33 | 0,11 | 0,33 | 0,11 | 1,00 | 3,00 | 3,00 | 1,00 | 0,33 | 0,33 | 0,11 | **4,4%** |
| **Beziehungen** | 0,11 | 0,11 | 1,00 | 0,11 | 0,11 | 0,11 | 0,11 | 0,11 | 0,11 | 0,11 | 0,33 | 0,33 | 0,11 | 0,11 | 0,11 | 0,11 | **0,5%** |
| **Flexibilität, Wartbarkeit und Einfachheit** | 3,00 | 3,00 | 9,00 | 1,00 | 3,00 | 1,00 | 0,33 | 1,00 | 0,33 | 1,00 | 1,00 | 3,00 | 9,00 | 3,00 | 3,00 | 1,00 | **6,7%** |
| **Aufbau** | 9,00 | 1,00 | 9,00 | 0,33 | 1,00 | 0,33 | 0,33 | 1,00 | 0,33 | 1,00 | 3,00 | 3,00 | 3,00 | 9,00 | 3,00 | 1,00 | **7,9%** |
| **Sprachadäquanz** | 9,00 | 3,00 | 9,00 | 1,00 | 3,00 | 1,00 | 1,00 | 3,00 | 0,33 | 0,33 | 3,00 | 3,00 | 9,00 | 9,00 | 9,00 | 1,00 | **9,8%** |
| **Konstruktionsadäquanz** | 9,00 | 9,00 | 9,00 | 3,00 | 3,00 | 1,00 | 1,00 | 3,00 | 1,00 | 3,00 | 1,00 | 3,00 | 9,00 | 9,00 | 3,00 | 1,00 | **10,1%** |
| **Vergleichbarkeit** | 9,00 | 3,00 | 9,00 | 1,00 | 1,00 | 0,33 | 0,33 | 1,00 | 0,33 | 0,33 | 1,00 | 1,00 | 3,00 | 1,00 | 0,33 | 0,33 | **6,2%** |
| **Anwendbarkeit** | 9,00 | 9,00 | 9,00 | 3,00 | 3,00 | 3,00 | 1,00 | 3,00 | 1,00 | 3,00 | 3,00 | 3,00 | 3,00 | 9,00 | 3,00 | 0,33 | **9,3%** |
| **Vorgehen** | 9,00 | 1,00 | 9,00 | 1,00 | 1,00 | 3,00 | 0,33 | 3,00 | 0,33 | 1,00 | 3,00 | 3,00 | 3,00 | 3,00 | 1,00 | 0,33 | **6,9%** |
| **Perspektive** | 9,00 | 0,33 | 3,00 | 1,00 | 0,33 | 0,33 | 1,00 | 1,00 | 0,33 | 0,33 | 1,00 | 1,00 | 3,00 | 3,00 | 0,33 | 0,33 | **4,4%** |
| **Support** | 3,00 | 0,33 | 3,00 | 0,33 | 0,33 | 0,33 | 0,33 | 1,00 | 0,33 | 0,33 | 1,00 | 1,00 | 1,00 | 3,00 | 0,33 | 0,11 | **2,6%** |
| **Granularität** | 1,00 | 1,00 | 9,00 | 0,11 | 0,33 | 0,11 | 0,11 | 0,33 | 0,33 | 0,33 | 0,33 | 1,00 | 1,00 | 0,33 | 0,11 | 0,11 | **3,5%** |
| **Konkurrenz** | 3,00 | 3,00 | 9,00 | 0,33 | 0,11 | 0,11 | 0,11 | 1,00 | 0,11 | 0,33 | 0,33 | 0,33 | 3,00 | 1,00 | 0,11 | 0,11 | **4,6%** |
| **Gesundheit** | 9,00 | 3,00 | 9,00 | 0,33 | 0,33 | 0,11 | 0,33 | 3,00 | 0,33 | 1,00 | 3,00 | 3,00 | 9,00 | 9,00 | 1,00 | 0,33 | **9,0%** |
| **Zuverlässigkeit** | 9,00 | 9,00 | 9,00 | 1,00 | 1,00 | 1,00 | 1,00 | 3,00 | 3,00 | 3,00 | 3,00 | 9,00 | 9,00 | 9,00 | 3,00 | 1,00 | **10,7%** |
| | 95,1 | 47,1 | 124,0 | 14,2 | 18,7 | 12,2 | 7,6 | 24,9 | 8,4 | 16,2 | 27,1 | 38,0 | 67,1 | 69,1 | 27,8 | 7,3 | **100%** |

## 13.5 Interdependenzmatrix



| | | Modellierungsprozess | | | | | | | | | | Meta-Modellsprache | | | | | | | | | | | | Reifegrad und Validierung | | | | | |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
| | | Relatives Gewicht aus paarweisem Vergleich | Konfigurationsmanagement | Vorgehens- und Phasenmodell | Qualitätssicherung | Modularität | Integration der Modellierungssprache | Organisationsmodell | Wiederverwendungsansätze | Vorgehensmodellen | Sichten | Dokumentation | Regeln | Abstraktionsebenen | Organistorische Rahmenbedingungen | Darstellungstechniken | Konzepte und Beziehungen | Formalität | Ontologische Vollständigkeit | Ontologische Klarheit | Dokumentation | Externe Validierung | Interne Validierung | Veröffentlichungen | Werkzeuge | Versionen | |
| | | 3,3% | | 4,4% | 0,5% | 6,7% | 7,9% | 9,8% | 10,1% | 6,2% | 9,3% | 6,9% | 4,4% | 2,6% | 3,5% | 4,6% | 9,0% | 10,7% | 100% | | | | | | | | | |
| Modellierungsprozess | Konfigurationsmanagement | | | 9 | | | | | 3 | | 9 | | | | | | | | 6,2% | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | |
| | Vorgehens- und Phasenmodell | | 9 | | | 3 | | | | | 9 | | | | | | | | 6,6% | 3 | 0 | 3 | 0 | 3 | 0 | 3 | 0 | 3 | |
| | Qualitätssicherung | 1 | | | | 1 | | | 3 | | | | | | | | | 9 | 4,8% | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | |
| | Modularität | | | | | 9 | | | | | 3 | | | | | | | | 3,8% | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | |
| | Integration der Modellierungssprache | | | | | | | | | | 3 | | 9 | | | | | | 2,4% | 0 | 0 | 3 | 0 | 0 | 0 | 0 | 0 | 9 | |
| | Organisationsmodell | | | | | | | | | | 3 | 9 | | | 3 | | | | 3,7% | 0 | 0 | 3 | 0 | 0 | 0 | 3 | 0 | 0 | |
| | Wiederverwendungsansätze | | | | | 9 | | | 3 | | 9 | | | | | | | | 6,1% | 0 | 0 | 0 | 0 | 0 | 0 | 3 | 0 | 3 | |
| Meta-Modellsprache | Vorgehensmodellen | 9 | | | | | | 3 | | | | | | | | | | | 2,1% | 9 | 0 | 3 | 9 | 0 | 0 | 0 | 0 | 0 | |
| | Sichten | 9 | | | | | | 3 | | | | | | | | | | | 2,1% | 9 | 9 | 9 | 0 | 0 | 0 | 0 | 0 | 0 | |
| | Dokumentation | 9 | | | | | | 3 | | | | | | | | | | | 2,1% | 0 | 0 | 9 | 0 | 0 | 0 | 0 | 0 | 0 | |
| | Regeln | 9 | | | | | | 3 | | | | | 9 | | | | | | 3,5% | 0 | 9 | 9 | 0 | 0 | 0 | 0 | 0 | 0 | |
| | Abstraktionsebenen | | | | | 3 | | 3 | | | | | | 9 | | | | | 5,3% | 9 | 9 | 9 | 0 | 0 | 0 | 0 | 0 | 0 | |
| | Organistorische Rahmenbedingungen | | | 9 | | | | 3 | | | | | | | 9 | | | | 2,4% | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | |
| | Darstellungstechniken | 3 | | | | | | 3 | | | 3 | | | | | 9 | | | 2,4% | 9 | 9 | 9 | 0 | 0 | 0 | 0 | 0 | 0 | |
| | Konzepte und Beziehungen | 9 | | | | | | 3 | | | 3 | | | | | 9 | 9 | | 3,1% | 9 | 9 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | |
| | Formalität | | | | | | | | | 9 | | | | | | | | 9 | 2,8% | 3 | 3 | 3 | 0 | 0 | 0 | 0 | 0 | 0 | |
| | Ontologische Vollständigkeit | 9 | | | 9 | | | 9 | 3 | | | | | | | | | 9 | 5,7% | 3 | 9 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | |
| | Ontologische Klarheit | | | | 1 | | | 9 | 9 | | 3 | | 3 | | | | | 9 | 8,8% | 9 | 9 | 3 | 0 | 0 | 0 | 0 | 0 | 0 | |
| Reifegrad und Validierung | Dokumentation | 3 | | 3 | 3 | | | 9 | | | | | | | | | | | 6,7% | 9 | 9 | 3 | 0 | 0 | 0 | 0 | 0 | 0 | |
| | Externe Validierung | | | | | | | | | 3 | | | | | | | | 9 | 4,6% | 3 | 3 | 9 | 0 | 0 | 0 | 0 | 0 | 0 | |
| | Interne Validierung | | | | | | | | | 1 | | 3 | | | | | | 3 | 2,4% | 3 | 9 | 9 | 0 | 0 | 0 | 0 | 0 | 0 | |
| | Veröffentlichungen | | | | | | | 1 | | | | | | | | | | 9 | 4,3% | 0 | 9 | 9 | 9 | 0 | 0 | 0 | 0 | 0 | |
| | Werkzeuge | | | | | 9 | | | | | 9 | | | | | | | | 5,3% | 9 | 3 | 9 | 3 | 9 | 0 | 0 | 0 | 0 | |
| | Versionen | | | | | | | | | | | | | | | | | 9 | 2,8% | 3 | 9 | 9 | 9 | 9 | 9 | 0 | 0 | 0 | |
| | Input/Output | | | | | | | | | | | | | | | | | | 2,8% | | | | | | | | | | |
| | Prozess | | | | | | | | | | | | | | | | | | 4,4% | | | | | | | | | | |
| | Beziehungen | | | | | | | | | | | | | | | | | | 0,5% | | | | | | | | | | |
| | Flexibilität, Wartbarkeit und Einfachheit | | | | | | | | | | | | | | | | | | 6,7% | | | | | | | | | | |
| | Aufbau | | | | | | | | | | | | | | | | | | 7,9% | | | | | | | | | | |
| | Sprachadäquanz | | | | | | | | | | | | | | | | | | 9,8% | | | | | | | | | | |
| | Konstruktionsadäquanz | | | | | | | | | | | | | | | | | | 10,1% | | | | | | | | | | |
| | Vergleichbarkeit | | | | | | | | | | | | | | | | | | 6,2% | | | | | | | | | | |
| | Anwendbarkeit | | | | | | | | | | | | | | | | | | 9,3% | | | | | | | | | | |
| | Vorgehen | | | | | | | | | | | | | | | | | | 6,9% | | | | | | | | | | |
| | Perspektive | | | | | | | | | | | | | | | | | | 4,4% | | | | | | | | | | |
| | Support | | | | | | | | | | | | | | | | | | 2,6% | | | | | | | | | | |
| | Granularität | | | | | | | | | | | | | | | | | | 3,5% | | | | | | | | | | |
| | Konkurrenz | | | | | | | | | | | | | | | | | | 4,6% | | | | | | | | | | |
| | Gesundheit | | | | | | | | | | | | | | | | | | 9,0% | | | | | | | | | | |
| | Zuverlässigkeit | | | | | | | | | | | | | | | | | | 10,7% | | | | | | | | | | |
| | | | | | | | | | | | | | | | | | | | | Adonis | 3 | 9 | 0 | 3 | 9 | 9 | 9 | 9 |
| | | | | | | | | | | | | | | | | | | | | ViewPoints | 0 | 3 | 9 | 9 | 9 | 3 | 9 | 9 |
| | | | | | | | | | | | | | | | | | | | | MEMO | 0 | 3 | 9 | 0 | 9 | 3 | 9 | 9 |
| | | | | | | | | | | | | | | | | | | | | JKOGGE | 9 | 3 | 9 | 0 | 9 | 9 | 9 | 9 |
| | | | | | | | | | | | | | | | | | | | | MetaEdit+ | 9 | 9 | 9 | 9 | 9 | 9 | 9 | 9 |

# 14 KMS der E$^3$-Methode

## 14.1 Beispiel der Versionierung eines Entity-Relationship Modells

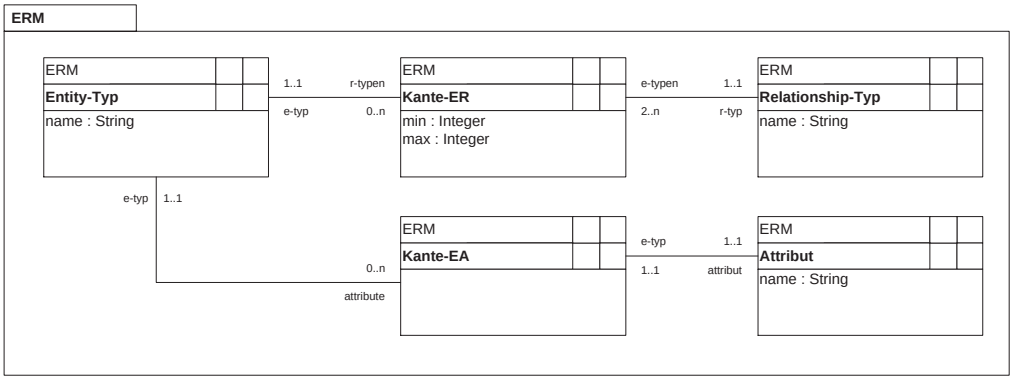



Abbildung 85: Typisierung des ERM

Um die Verwaltung von Modellversionen des E$^3$-Modells durch das Modell-KMS darzustellen, wird ein einfaches Entity-Relationship Modell zugrunde gelegt und schrittweise verändert. Es basiert auf der Typisierung in Abbildung 85 und hat den folgenden Aufbau:

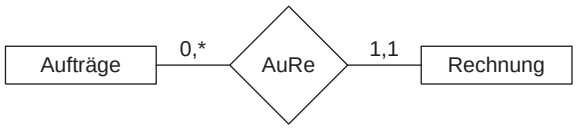



Abbildung 86: Einfaches Beispiel für ein ER-Modell

Unabhängig von der Tatsache, dass dieses Modell bereits in mehreren Arbeitsschritten entstanden sein muss, werden bei der Darstellung der zugehörigen Konfiguration und Konfigurationsstruktur die initialen Versionsnummern verwendet. Die Konfiguration enthält nicht die Elemente der Typebene und die Konfigurationsstruktur demzufolge auch nicht die Beziehung der Elemente zu ihren Typen. Schließlich wurde für dieses einfache Beispiel auch auf die Abbildung der Elemente aus der View- und Präsentationsebene verzichtet. Die Konfigurationsstrukturen verwalten Abhängigkeitsbeziehungen, innerhalb derer ein E$^3$-Elemente als abhängig oder unabhängig aufgeführt werden muss. Eine entsprechende Unterscheidung erfolgt im Beispiel jedoch nicht. Insgesamt hätte eine Beachtung all dieser Punkte nicht unbedingt zu mehr Klarheit beigetragen. Es ergeben sich also vereinfacht die folgende Konfiguration und Konfigurationsstruktur.

| Konfiguration - Version 1.0 | | |
|---|---|---|
| Identifikator | Version | Beschreibung |
| E1 | 1.0 | E-Typ Auftrag |
| E1-P1 | 1.0 | Property *name* des E1 |
| E1-P2 | 1.0 | Property *r-typen* des E1 |
| E2 | 1.0 | E-Typ Rechnung |
| E2-P1 | 1.0 | Property *name* des E2 |
| E2-P2 | 1.0 | Property *r-typen* des E1 |
| K1 | 1.0 | Kante, die E1 mit R1 verbindet |
| K1-P1 | 1.0 | Property *r-typen* der K1 |
| K1-P2 | 1.0 | Property *e-typen* der K1 |
| K2 | 1.0 | Kante, die E2 mit R1 verbindet |
| K2-P1 | 1.0 | Property *r-typen* der K2 |
| K2-P2 | 1.0 | Property *e-typen* der K2 |
| R1 | 1.0 | R-Typ AuRe |
| R1-P1 | 1.0 | Property *name* des R1 |
| R1-P2 | 1.0 | Property *e-typen* des R1 |
| CS1 | 1.0 | Konfigurationsstruktur |

| Konfigurationsstruktur CS1 - Version 1.0 | | |
|---|---|---|
| Konfigurationselement 1 | Konfigurationselement 2 | Art der Beziehung |
| E1 | E1-P1 | Property zu seinem Objekt |
| E1 | E1-P2 | Property zu seinem Objekt |
| E2 | E2-P1 | Property zu seinem Objekt |
| E2 | E2-P2 | Property zu seinem Objekt |
| K1 | K1-P1 | Property zu seinem Objekt |
| K1 | K1-P2 | Property zu seinem Objekt |
| K2 | K2-P1 | Property zu seinem Objekt |
| K2 | K2-P2 | Property zu seinem Objekt |
| R1 | R1-P1 | Property zu seinem Objekt |
| R1 | R1-P2 | Property zu seinem Objekt |
| R1-P2 | K1-P1 | Verbindung zweier Properties (Assoziationsmuster) |
| K1-P2 | E1-P2 | Verbindung zweier Properties (Assoziationsmuster) |
| R1-P2 | K2-P1 | Verbindung zweier Properties (Assoziationsmuster) |
| K2-P2 | E2-P2 | Verbindung zweier Properties (Assoziationsmuster) |

In einem ersten Schritt wird der E-Typ *Aufträge* in *Auftrag* umbenannt. Hierzu wird das Property *E1-P1* verändert und es entsteht eine neue Konfiguration. Die Konfigurationsstruktur *CS1* bleibt unverändert.

| Konfiguration - Version 1.1 | | |
|---|---|---|
| Identifikator | Version | Beschreibung |
| E1 | 1.0 | E-Typ Auftrag |
| E1-P1 | 1.1 | Property *name* des E1 |
| E1-P2 | 1.0 | Property *r-typen* des E1 |
| E2 | 1.0 | E-Typ Rechnung |
| E2-P1 | 1.0 | Property *name* des E2 |
| E2-P2 | 1.0 | Property *r-typen* des E1 |
| K1 | 1.0 | Kante, die E1 mit R1 verbindet |
| K1-P1 | 1.0 | Property *r-typen* der K1 |
| K1-P2 | 1.0 | Property *e-typen* der K1 |
| K2 | 1.0 | Kante, die E2 mit R1 verbindet |
| K2-P1 | 1.0 | Property *r-typen* der K2 |
| K2-P2 | 1.0 | Property *e-typen* der K2 |
| R1 | 1.0 | R-Typ AuRe |
| R1-P1 | 1.0 | Property *name* des R1 |
| R1-P2 | 1.0 | Property *e-typen* des R1 |
| CS1 | 1.0 | Konfigurationsstruktur |

Mit dem Löschen der Kante *K1* zwischen *Auftrag* und *AuRe* entstehen nacheinander zwei neue Versionen der Konfigurationsstruktur *CS1*, da bei diesem Vorgang insgesamt zwei Beziehungen entfernt werden. Zuerst wird also *CS1* schrittweise angepasst und anschließend die Kante gelöscht. In jedem der insgesamt drei Schritte entsteht eine neue Konfiguration, der Einfachheit halber werden die Ergebnisse der Schritte dargestellt. Das neue ER-Modell wird in Abbildung 87 dargestellt.

| Konfiguration - Version 1.4 | | |
|---|---|---|
| Identifikator | Version | Beschreibung |
| E1 | 1.0 | E-Typ Auftrag |
| E1-P1 | 1.1 | Property *name* des E1 |
| E1-P2 | 1.0 | Property *r-typen* des E1 |
| E2 | 1.0 | E-Typ Rechnung |
| E2-P1 | 1.0 | Property *name* des E2 |
| E2-P2 | 1.0 | Property *r-typen* des E1 |
| K2 | 1.0 | Kante, die E2 mit R1 verbindet |
| K2-P1 | 1.0 | Property *r-typen* der K2 |
| K2-P2 | 1.0 | Property *e-typen* der K2 |
| R1 | 1.0 | R-Typ AuRe |
| R1-P1 | 1.0 | Property *name* des R1 |
| R1-P2 | 1.0 | Property *e-typen* des R1 |
| CS1 | 1.2 | Konfigurationsstruktur |

| Konfigurationsstruktur CS1 - Version 1.2 | | |
|---|---|---|
| Konfigurationselement 1 | Konfigurationselement 2 | Art der Beziehung |
| E1 | E1-P1 | Property zu seinem Objekt |
| E1 | E1-P2 | Property zu seinem Objekt |
| E2 | E2-P1 | Property zu seinem Objekt |
| E2 | E2-P2 | Property zu seinem Objekt |
| K2 | K2-P1 | Property zu seinem Objekt |
| K2 | K2-P2 | Property zu seinem Objekt |
| R1 | R1-P1 | Property zu seinem Objekt |
| R1 | R1-P2 | Property zu seinem Objekt |
| R1-P2 | K2-P1 | Verbindung zweier Properties (Assoziationsmuster) |
| K2-P2 | E2-P2 | Verbindung zweier Properties (Assoziationsmuster) |


Abbildung 87: ER-Modell nach Löschen der Kante

In weiteren drei Schritten wird eine neue Kante *K3* angelegt, welche die getrennten Elemente des Modells erneut verbindet. Im ersten Schritt entsteht das neue Konfigurationselement *K3*, entsprechend auch eine neue Konfiguration.[147] Es entsteht die Version 1. 3 der Konfigurationsstruktur *CS1* und die Version 1. 5 der Konfiguration. Durch das Herstellen der Verbindungen zwischen Kante *K3*, dem E-Typen *E1* und dem R-Typen *R1* werden nacheinander zwei neue Beziehungen in die Konfigurationsstruktur *CS1* eingetragen, von dieser und der übergeordneten Konfiguration entstehen entsprechend auch zwei neue Versionen. Die abschließend entstehenden Tabellen beziehen sich auf die grafische Darstellung aus Abbildung 88.

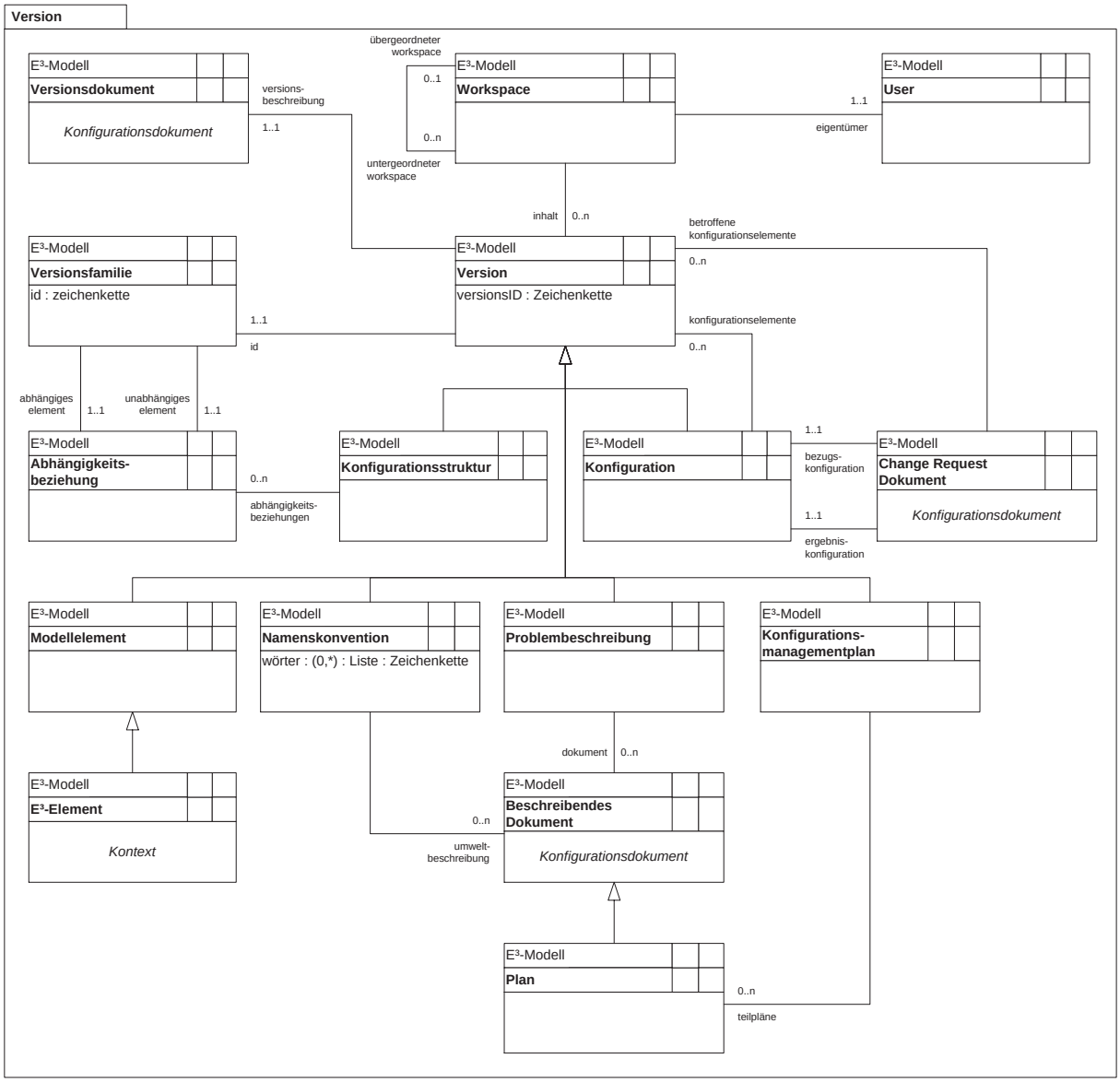
[147] Dabei wird davon ausgegangen, dass das mit dem Einfügen verbundene Herstellen der Beziehungen des Objekts *K3* Kante zu seinen Properties im gleichen Schritt geschieht, wie das Einfügen des Objekts und seiner Properties selbst. Die Schritte erfolgen sequentiell, sind jedoch als eine Modellieroperation zu betrachten. Die Konfiguration wird entsprechend nur einmal „hochgezählt“.

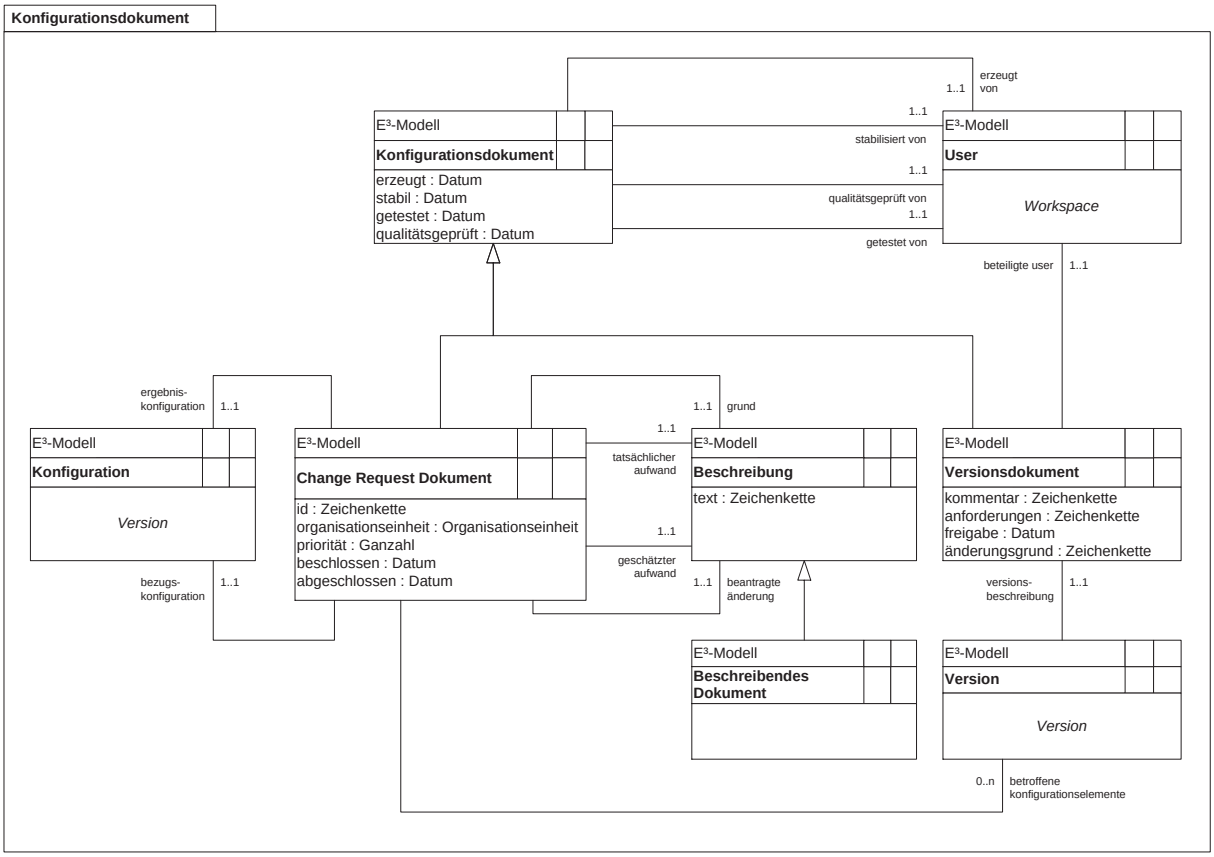
| Konfiguration - Version 1.7 | | |
|---|---|---|
| Identifikator | Version | Beschreibung |
| E1 | 1.0 | E-Typ Auftrag |
| E1-P1 | 1.1 | Property *name* des E1 |
| E1-P2 | 1.0 | Property *r-typen* des E1 |
| E2 | 1.0 | E-Typ Rechnung |
| E2-P1 | 1.0 | Property *name* des E2 |
| E2-P2 | 1.0 | Property *r-typen* des E1 |
| K2 | 1.0 | Kante, die E2 mit R1 verbindet |
| K2-P1 | 1.0 | Property *r-typen* der K2 |
| K2-P2 | 1.0 | Property *e-typen* der K2 |
| K3 | 1.0 | Kante, die E1 mit R1 verbindet |
| K3-P1 | 1.0 | Property *r-typen* der K1 |
| K3-P2 | 1.0 | Property *e-typen* der K1 |
| R1 | 1.0 | R-Typ AuRe |
| R1-P1 | 1.0 | Property *name* des R1 |
| R1-P2 | 1.0 | Property *e-typen* des R1 |
| CS1 | 1.2 | Konfigurationsstruktur |

| Konfigurationsstruktur CS1 - Version 1.5 | | |
|---|---|---|
| Konfigurationselement 1 | Konfigurationselement 2 | Art der Beziehung |
| E1 | E1-P1 | Property zu seinem Objekt |
| E1 | E1-P2 | Property zu seinem Objekt |
| E2 | E2-P1 | Property zu seinem Objekt |
| E2 | E2-P2 | Property zu seinem Objekt |
| K2 | K2-P1 | Property zu seinem Objekt |
| K2 | K2-P2 | Property zu seinem Objekt |
| K3 | K3-P1 | Property zu seinem Objekt |
| K3 | K3-P2 | Property zu seinem Objekt |
| R1 | R1-P1 | Property zu seinem Objekt |
| R1 | R1-P2 | Property zu seinem Objekt |
| R1-P2 | K2-P1 | Verbindung zweier Properties (Assoziationsmuster) |
| K2-P2 | E2-P2 | Verbindung zweier Properties (Assoziationsmuster) |
| R1-P2 | K3-P1 | Verbindung zweier Properties (Assoziationsmuster) |
| K3-P2 | E1-P2 | Verbindung zweier Properties (Assoziationsmuster) |


Abbildung 88: ER-Modell nach Wiedereinfügen einer Kante

## 14.2 Strukturelle Spezifikation des KMS der E$^3$-Methode




Konfigurationsdokument
E³-Modell
Konfigurationsdokument
erzeugt : Datum
stabil : Datum
getestet : Datum
qualitätsgeprüft : Datum
1..1
erzeugt von
1..1
E³-Modell
User
Workspace
stabilisiert von
1..1
qualitätsgeprüft von
1..1
getestet von
beteiligte user
1..1
ergebnis-konfiguration
1..1
E³-Modell
Konfiguration
Version
bezugs-konfiguration
1..1
E³-Modell
Change Request Dokument
id : Zeichenkette
organisationseinheit : Organisationseinheit
priorität : Ganzahl
beschlossen : Datum
abgeschlossen : Datum
1..1
tatsächlicher aufwand
1..1
geschätzter aufwand
1..1
grund
1..1
E³-Modell
Beschreibung
text : Zeichenkette
1..1
beantragte änderung
E³-Modell
Versionsdokument
kommentar : Zeichenkette
anforderungen : Zeichenkette
freigabe : Datum
änderungsgrund : Zeichenkette
versions-beschreibung
1..1
E³-Modell
Beschreibendes Dokument
E³-Modell
Version
Version
0..n
betroffene konfigurationselemente

## 14.3 Change Request Form des Method Engineering

| Element | Erläuterung |
|---|---|
| *Methodenidentifikator* | Bezeichnet die betroffene Methode eindeutig. |
| *Antragsnummer* | Laufende Nummer des Änderungsantrages für die betroffene Methode. |
| *Antragsteller* | Der Name von der Person, die die Änderung wünscht. |
| *Rolle des Antragstellers* | Die Angabe der Rolle ist relevant, damit der Methoden-Verantwortliche feststellen kann, ob der Antragsteller überhaupt eine Anforderungsänderung formulieren darf. |
| *Organisationseinheit des Antragstellers* | Dient in erster Linie als Kontaktanschrift für die Verifizierung der Anforderungsänderung (des Änderungsantrages) und damit der Methoden-Verantwortliche ggf. für die Analyse der Anforderungsänderungen nachfragen kann. |
| *Antragsdatum* | Datum der Antragstellung, d. h. das Datum, an dem der Methoden-Verantwortliche die Anforderungsänderung beginnt aufzunehmen. |
| *Methoden-Verantwortlicher* | Name des Methoden-Verantwortlichen, der den Änderungsantrag verfasst. |
| *Basis Methoden-Version* | Methoden-Version, an der die Änderung gewünscht wird (Versionsnummer). |
| *Anforderungsänderung* | Prosaische Beschreibung der beantragten Änderung. |
| *Gründe* | Angabe von Gründen, weshalb die Änderung gewünscht wird. |
| *Analysedatum* | Datum, an dem der Methoden-Verantwortliche beginnt, die Auswirkungen der Anforderungsänderung festzustellen. |
| *Methodeninterne Auswirkungen* | Beschreibung der methodeninternen Auswirkungen der Anforderungsänderung. |
| *Methodenexterne Auswirkungen* | Beschreibung der methodenexternen Auswirkungen der Anforderungsänderung. |
| *Priorisierung* | Bezeichnet die Priorität, die der Methoden-Verantwortliche für die Anforderungsänderung, nach seiner definierten Skala, in Abstimmung mit dem Antragsteller festlegt. |

| **Element** | **Erläuterung** |
|---|---|
| *Aufwandsschätzung* | Geschätzte Zeitdauer für die Umsetzung der Anforderungsänderung. Während der Methoden-Verantwortliche den Aufwand für die methodeninternen Auswirkungen im Laufe der Zeit wohl recht gut schätzen kann, hängt die Zeitdauer für die Umsetzung der methodenexternen Auswirkungen doch stark vom Umfang der Methodennutzung ab. Dazu muss er ggf. Erkundigungen einholen. |
| *CCB Datum* | Datum, an dem der Änderungsantrag an das Change Control Board weitergeleitet wurde. |
| *Datum CCB Entscheidung* | Datum, an dem das Change Control Board über den Antrag entschieden hat. |
| *CCB Verantwortlicher* | Hauptverantwortlicher Bearbeiter beim Change Control Board. |
| *CCB Entscheidung* | Sagt aus, ob der Änderungsantrag genehmigt wurde oder nicht. |
| *Methoden-Version* | Versionsnummer, die die Methode nach ihrer Anpassung erhalten soll und auf die sich die zu ergänzende Anforderungsspezifikation und die zu überarbeitende Dokumentation der Methode beziehen müssen. Sie wird vom Methoden-Verantwortlichen festgelegt. |
| *IA-Datum* | Datum, an dem die Methoden-Entwickler damit beginnen sollen, die Anforderungsänderung umzusetzen. Der Methoden-Verantwortliche legt dieses Datum, nach seinem definierten Konzept, auf Basis der Priorisierung fest. |
| *Methoden-Entwickler* | Name des Methoden-Entwicklers, der die Anforderungsänderung in der Anforderungsspezifikation ergänzt (bspw. Verweis auf diesen Änderungsantrag) und die Methode auf deren Basis anpasst. |
| *Dokumentationsstartdatum* | Datum, an dem mit der Überarbeitung der Dokumentation begonnen wird. |
| *Qualitätssicherer - Dokumentation* | Name des Qualitätssicherers, der die Dokumentation überarbeitet. |
| *Dokumentationsenddatum* | Datum, an dem die neue Version der Dokumentation vorliegt. |
| *Implementierungsverantwortlicher* | Name des Methoden-Entwicklers, der die Implementierung und den technischen Funktionstest der angepassten Methode übernimmt. |

| **Element** | **Erläuterung** |
|---|---|
| *Freigabe Implementierung* | Datum, an dem der Funktionstest erfolgreich abgeschlossen wurde und die Methode zur Validierung bezüglich der Anforderungsspezifikation freigegeben wird. |
| *Tester* | Name des Testers, der die Validierung der Methode bezüglich der Anforderungsspezifikation vornimmt. |
| *Freigabe Test* | Datum, an dem der Test erfolgreich absolviert wurde und die Methode zur Validierung bezüglich der qualitativen Ansprüche freigegeben wird. |
| *Qualitätssicherer - Validierung* | Name des Qualitätssicherers, der die Methode bezüglich der qualitativen Ansprüche validiert. |
| *Freigabe Qualitätssicherung* | Datum, an dem die Qualitätssicherung erfolgreich absolviert wurde und die Methode zur Nutzung bereit ist. |
| *Iterationsbemerkungen* | Feld für Bemerkungen, wenn ggf. Iterationen durch nicht erfolgreiche Prüfungen notwendig werden. |
| *Nutzungsfreigabe (EA-Datum)* | Datum, an dem der Methoden-Verantwortliche die Methode freigibt, damit die Methoden-Anwender die erstellten Modelle überarbeiten können. |
| *IST-Aufwand* | Tatsächlich benötigte Zeitdauer, bis die Anforderungsänderung komplett umgesetzt wurde. |
| *weitere Bemerkungen* | Feld für Bemerkungen, wenn bspw. verantwortliche Personen wechseln oder vertreten werden, dabei können entsprechende Verweise bei den betroffenen Feldern vorgenommen werden. |

# Index

## Ehrenwörtliche Erklärung

Hiermit versichere ich, die vorliegende Arbeit selbständig, ohne fremde Hilfe und ohne Benutzung anderer als der von mir angegebenen Quellen angefertigt zu haben. Alle aus fremden Quellen direkt oder indirekt übernommenen Gedanken sind als solche gekennzeichnet. Die Arbeit wurde noch keiner Prüfungsbehörde in gleicher oder ähnlicher Form vorgelegt.

Dresden, den 15. Januar 2003