ECONSTOR
Make Your Publications Visible.

A Service of

ZBW Leibniz-Informationszentrum Wirtschaft
Leibniz Information Centre for Economics

Kirchner, Lutz

Research Report

Entwurf einer Modellierungssprache zur Unterstützung der Aufgaben des IT-Managements: Grundlagen, Anforderungen und Metamodell

ICB-Research Report, No. 11

Provided in Cooperation with:
University Duisburg-Essen, Institute for Computer Science and Business Information Systems (ICB)

Suggested Citation: Kirchner, Lutz (2007) : Entwurf einer Modellierungssprache zur Unterstützung der Aufgaben des IT-Managements: Grundlagen, Anforderungen und Metamodell, ICB-Research Report, No. 11, Universität Duisburg-Essen, Institut für Informatik und Wirtschaftsinformatik (ICB), Essen

This Version is available at:
https://hdl.handle.net/10419/58170

Standard-Nutzungsbedingungen:

Die Dokumente auf EconStor dürfen zu eigenen wissenschaftlichen Zwecken und zum Privatgebrauch gespeichert und kopiert werden.

Sie dürfen die Dokumente nicht für öffentliche oder kommerzielle Zwecke vervielfältigen, öffentlich ausstellen, öffentlich zugänglich machen, vertreiben oder anderweitig nutzen.

Sofern die Verfasser die Dokumente unter Open-Content-Lizenzen (insbesondere CC-Lizenzen) zur Verfügung gestellt haben sollten, gelten abweichend von diesen Nutzungsbedingungen die in der dort genannten Lizenz gewährten Nutzungsrechte.

Terms of use:

Documents in EconStor may be saved and copied for your personal and scholarly purposes.

You are not to copy documents for public or commercial purposes, to exhibit the documents publicly, to make them publicly available on the internet, or to distribute or otherwise use the documents in public.

If the documents have been made available under an Open Content Licence (especially Creative Commons Licences), you may exercise further usage rights as specified in the indicated licence.

WWW.ECONSTOR.EU

Mitglied der Leibniz-Gemeinschaft

ICB
Institut für Informatik und Wirtschaftsinformatik

Lutz Kirchner

Entwurf einer Modellierungssprache zur Unterstützung der Aufgaben des IT-Managements

ICB-RESEARCH REPORT

Grundlagen, Anforderungen und Metamodell

ICB-Research Report No.11
February 2007
Universität Duisburg-Essen

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

The ICB Research Reports comprise preliminary results which will usually be revised for subsequent publications. Critical comments would be appreciated by the authors.

Alle Rechte vorbehalten. Insbesondere die der Übersetzung, des Nachdruckes, des Vortrags, der Entnahme von Abbildungen und Tabellen – auch bei nur auszugsweiser Verwertung.

All rights reserved. No part of this report may be reproduced by any means, or translated.

Author's Address:

Lutz Kirchner

Institut für Informatik und
Wirtschaftsinformatik (ICB)
Universität Duisburg-Essen
D-45117 Essen

lutz.kirchner@uni-due.de

ICB Research Reports

Edited by:

Prof. Dr. Heimo Adelsberger
Prof. Dr. Peter Chamoni
Prof. Dr. Frank Dorloff
Prof. Dr. Klaus Echtle
Prof. Dr. Stefan Eicker
Prof. Dr. Ulrich Frank
Prof. Dr. Michael Goedicke
Prof. Dr. Tobias Kollmann
Prof. Dr. Bruno Müller-Clostermann
Prof. Dr. Klaus Pohl
Prof. Dr. Erwin P. Rathgeb
Prof. Dr. Rainer Unland
Prof. Dr. Stephan Zelewski

Managing Assistant and Contact:

Jürgen Jung

Institut für Informatik und
Wirtschaftsinformatik (ICB)
Universität Duisburg-Essen
Universitätsstr. 9
45141 Essen
Germany

Email: icb@uni-duisburg-essen.de

ISSN 1860-2770

Abstract

Die Bewältigung der Aufgaben im Rahmen des Managements betrieblicher IT-Landschaften erweist sich vor dem Hintergrund der stetig zunehmenden Heterogenität und Komplexität der in Unternehmen eingesetzten IT-Lösungen als eine immer größere Herausforderung. Bestehende Ansätze aus Wissenschaft und Praxis werden den hieraus resultierenden Anforderungen an das IT-Management in der Regel nur in unbefriedigendem Umfang gerecht. Nicht zuletzt bedingt durch den Mangel an geeigneten Beschreibungskonzepten für IT-Landschaften sind die eingesetzten IT-Komponenten in ihrer Gesamtheit oftmals nicht mehr zu überblicken.

In dem vorliegenden Arbeitsbericht wird die Spezifikation einer Sprache für die Modellierung von IT-Landschaften vorgestellt. Diese Sprache ist Teil einer Methode für das IT-Management, welche sich zurzeit in Entwicklung befindet. Das primäre Entwurfsziel ist die Bereitstellung dedizierter Sprachkonzepte für die Darstellung von IT-Komponenten und deren Beziehung zu weiteren betrieblichen Konzepten, um eine Grundlage für ein effektives und effizientes IT-Management zu schaffen.

Inhaltsverzeichnis

1 EINFÜHRUNG 1

2 ZENTRALE BEGRIFFE DES IT-MANAGEMENTS 4

2.1 INFORMATIONSTECHNIK UND INFORMATIONSSYSTEME 4
2.2 IT-MANAGEMENT 5
2.3 IT-GOVERNANCE 7
2.4 IT-SERVICE 8
2.5 IT-KOSTENRECHNUNG 11
2.6 NUTZEN VON IT 13
2.7 ZUSAMMENFASSUNG 15

3 ANFORDERUNGEN AN EINE MODELLIERUNGSSPRACHE FÜR DAS IT-MANAGEMENT 20

3.1 SPRACHSPEZIFIKATION 20
3.2 WIEDERVERWENDUNG 20
3.3 ANWENDUNGSNAHE KONZEPTE 21
3.4 OPERATIONALISIERBARKEIT 25
3.5 SCHNITTSTELLEN ZU ANDEREN MODELLIERUNGSSPRACHEN 27
3.6 INTEGRITÄTSBEDINGUNGEN 28

4 DIE INFORMATION TECHNOLOGY MODELLING LANGUAGE 29

4.1 KERNKONZEPTE DER ITML 31
4.2 EINORDNUNG DER ITML IN DAS MEMO-FRAMEWORK 33
4.2.1 Einführung in MEMO 33
4.2.2 Die MEMO-Sprachen 34
4.3 IT-SPEZIFISCHE SPRACHKONZEPTE DER ITML 41
4.3.1 Hardwarespezifische Konzepte 41
4.3.2 Softwarespezifische Konzepte 49
4.3.3 Integrations- und ergänzende Konzepte für Hard- und Software 55
4.3.4 Organisation und Services 57
4.3.5 Kosten- und Nutzenspezifische Konzepte 63
4.3.6 Informationssysteme 66
4.4 KONZEPTE ZUR WIEDERVERWENDUNG 70

5 ZUSAMMENFASSUNG UND AUSBLICK 74

6 LITERATUR 75

Abbildungsverzeichnis

Abbildung 1: Idealtypischer Lebenszyklus von IT-Landschaften 16
Abbildung 2: Handlungsrahmen für das IT-Management 18
Abbildung 3: Ausschnitt aus dem MEMO-Metametamodell (aus [Fran98a], Fig. 2, S. 8) 30
Abbildung 4: Basiselemente der Modellierungssprache 32
Abbildung 5: Die Kernkonzepte der ITML 32
Abbildung 6: MEMO Perspektiven, Aspekte und Subjekte (aus [Fran02], Fig. 1) 33
Abbildung 7: Integration der MEMO-Sprachen (in Anlehnung an [Fran99], Fig. 3) 35
Abbildung 8: Exemplarisches Strategienetz (aus [FrLa06], Fig. 4) 36
Abbildung 9: Konzepte der SML und ITML 37
Abbildung 10: Beispiel für einen Geschäftsprozess mit der MEMO-OrgML 37
Abbildung 11: Konzepte der OrgML und ITML 38
Abbildung 12: Beispiel für ein OML-Objektmodell (aus [Fran98c], Fig. 8) 39
Abbildung 13: Beispiel für ein ResML-Modell (aus [Jung07]) 40
Abbildung 14: Konzepte der ResML und ITML 40
Abbildung 15: Hardwarespezifische Konzepte der ITML 41
Abbildung 16: Softwarespezifische Konzepte der ITML 51
Abbildung 17: Hard- und softwarespezifische Konzepte 56
Abbildung 18: Standard und assoziierte Konzepte 57
Abbildung 19: Integration von Aufbau- und Ablauforganisationskonzepten 58
Abbildung 20: Rollenträger einer Organisationsrolle 60
Abbildung 21: IT-spezifische Kostenkonzepte 64
Abbildung 22: Zuordnung der speziellen Kostenarten 66
Abbildung 23: Informationssystem und technische Konzepte 66
Abbildung 24: Informationssystem und strategie- bzw. organisationsrelevante Konzepte 67
Abbildung 25: Risikokontext von Informationssystemen 69
Abbildung 26: Spezialisierung im Metamodell der ITML 71

Tabellenverzeichnis

Tabelle 1: Schreibweise der Kardinalitäten 30

1 Einführung

Die Bewältigung der Aufgaben im Rahmen des Managements betrieblicher IT-Landschaften erweist sich vor dem Hintergrund der stetig zunehmenden Heterogenität und Komplexität der in Unternehmen eingesetzten IT-Lösungen als eine immer größere Herausforderung (vgl. [Horv03], S. 719). Die Planung und Implementierung von IT-Lösungen bzw. von IT-Landschaften zur Realisierung von Services, welche wiederum die Geschäftsprozesse eines Unternehmens unterstützen, sowie die Analyse existierender Infrastrukturen hinsichtlich ihrer Leistungsfähigkeit und Wirtschaftlichkeit stehen hierbei zunächst im Vordergrund der Managementaktivitäten. Da es für den nachhaltigen Erfolg eines Unternehmens von fundamentaler Bedeutung ist, im Ergebnis eine möglichst effektive und effiziente Informationstechnik zur Verfügung stehen zu haben, ist an dieser Stelle ein dringender Bedarf an methodischer Unterstützung erkennbar. In einer Vielzahl von Unternehmen werden jedoch die mit dem IT-Management verbundenen Aufgaben noch immer häufig ohne erkennbare Methode angegangen, was in der Folge zu unbefriedigenden, suboptimalen Ergebnissen führt.

Eine in der Praxis häufig anzutreffende Herangehensweise an die o.g. Problemstellung ist neben dem Einsatz von proprietären Methoden die Adaption von Best Practice Ansätzen des IT-Managements sowie IT-Governance Frameworks. Hier sind an erster Stelle ITIL[1] und CobiT[2] zu nennen. Diese beinhalten jeweils eine Menge von Referenzprozessen niedrigen Detailgrads, die dokumentieren, welche Aktivitäten und Verantwortlichkeiten in einer IT-Organisation typischerweise vorzufinden sind. Beschreibungskonzepte für IT sowie detaillierte Prozessbeschreibungen sind allerdings nicht enthalten.

In der Wissenschaft existieren bisher kaum elaborierte Ansätze, die dediziert die vielfältigen Facetten des IT-Managements abdecken. Architekturframeworks, z.B. DoDAF[3] oder TOGAF[4], sowie Modellierungssprachen, bspw. 3LGM[5,] ARCUS[6] oder der ARIS-basierte Ansatz von Kirsch[7], bieten größtenteils keine durchgängig angemessenen Beschreibungskonzepte für IT-Landschaften. Die vorhandenen Beziehungen zwischen IT, Geschäftsprozessen, Strategien oder Umweltfaktoren finden hinsichtlich einer nachhaltigen Unterstützung der Tätigkeiten im Rahmen des IT-Managements kaum ausreichend Berücksichtigung. Vorgehensmodelle, die anleiten, wie die Modellierungskonzepte im Rahmen des IT-Managements eingesetzt werden können, sind in der Regel ebenfalls lediglich rudimentär dokumentiert[8]. Diese Mängel erschweren eine erfolgreiche Anwendung der Ansätze erheblich.

Ähnlichen Herausforderungen, wie sie im Bereich des IT-Managements zu finden sind, müssen sich Unternehmensmodellierungsmethoden, wie bspw. MEMO[9], ARIS[10] oder SOM[11], stellen. Diese zielen zunächst auf die Analyse und den Entwurf von Informationssystemen und bedienen sich zu die-

---

[1] S. z.B. [Olbr04] oder [ViGu04].

[2] S. [Cobi06].

[3] S. [DoDA04].

[4] S. [TOGA04].

[5] S. [WBW03] und [WHB+04].

[6] S. [HMT02] und [Tens05].

[7] S. [Kirs99].

[8] TOGAF bildet hierbei mit seinem Phasenkonzept (*Architecture Development Method*) eine Ausnahme.

[9] S. Abschnitt 4.2.1.

[10] S. [Sche02].

[11] S. [FeSi98].

sem Zweck einer Menge integrierter Modellierungssprachen, hauptsächlich für die Beschreibung der Aufbau- und Ablauforganisation eines Unternehmens, sowie Referenz- und Vorgehensmodellen. Durch die Anwendung angemessener Beschreibungskonzepte wird die hohe Komplexität der Domäne in der Regel erfolgreich kompensiert und der Untersuchungsgegenstand zugänglich gemacht. Parallelen zum IT-Management sind vor allem in Form des Untersuchungsgegenstands – das betriebliche Informationssystem - sowie der Aufgabe der Beherrschung des Lebenszyklus von Informationssystemen bzw. IT-Landschaften zu sehen. Allerdings wird im Bereich der Unternehmensmodellierung von der technischen Basis der betrieblichen Informationssysteme zugunsten einer eher betriebswirtschaftlichen Sichtweise weitgehend abstrahiert.

Vor dem Hintergrund der Potenziale der Unternehmensmodellierung liegt es nahe, einen ähnlichen Ansatz auch im Rahmen der Entwicklung einer IT-Management-Methode zu verfolgen. So sind ergänzend zu den existierenden Beschreibungsmitteln anwendungsnahe Modellierungskonzepte einzuführen, welche eine Abbildung der Objekte des IT-Managements auf einem zunächst hohen Abstraktionsniveau erlauben. In der Folge können abhängig von der konkreten Zielsetzung und Perspektive detailliertere Sichten auf die betriebliche IT und deren Zusammenhänge mit Organisation und Umwelt erstellt werden.

In dem vorliegenden Bericht wird die Spezifikation einer domänenspezifischen Sprache (*Domain Specific Language*, DSL) für die Modellierung von IT-Landschaften vorgestellt. Die Sprache ist Teil einer Methode für das IT-Management, welche sich zurzeit in Entwicklung befindet und als ein integraler Bestandteil der Unternehmensmodellierungsmethode MEMO (Multi-Perspective Enterprise Modelling) konzipiert ist. Neben der Modellierungssprache sind exemplarische, idealtypische Sprachanwendungen (Referenzmodelle) sowie einige Vorgehensmodelle, welche die Verwendung der Sprache hinsichtlich unterschiedlicher Tätigkeiten während der Planung und Analyse von IT anleiten, Bestandteil der Methode. Die Einbettung der Sprache in das MEMO-Framework erfolgt über die Integration der Modellierungssprache mit den bereits existierenden Sprachen zur Darstellung von Strategien und Geschäftsprozessen. Die jeweils zentralen Abstraktionen der betriebswirtschaftlichen und informationstechnischen Perspektive auf ein Unternehmen werden zusammengeführt. Somit kann das IT-Management durch den Einsatz von wohl strukturierten und sorgfältig integrierten Konzepten nachhaltig angereichert und zum Erfolg geführt werden.

Die Entscheidung für den Entwurf einer DSL basiert auf einer Reihe von Vorteilen, die diese in unserem Anwendungskontext gegenüber einer generischen Sprache (*General Purpose Language*, GPL) aufweist. So definiert die Sprachspezifikation einer DSL in Form einer Rekonstruktion der Fachterminologie die zentralen Konzepte einer Domäne sowie deren Beziehungen untereinander. Diese Konzepte enthalten eine hohe, anwendungsnahe Semantik, wodurch ihre Anwendung allerdings auf eine spezifische Zieldomäne – in unserem Falle IT-Landschaften - beschränkt wird. GPLs, wie z.B. die UML[1], eignen sich hingegen generell für eine Anwendung auf eine große Anzahl von Domänen. Als Konsequenz enthalten die Konzepte der Sprachen wenig anwendungsnahe Semantik, da dies die Wiederverwendungsreichweite der Sprache nachhaltig einschränken würde. Hieraus ergibt sich ein in unserem Kontext zentraler Vorteil von DSLs: die Reduktion der Fehlerrate bei der Modellierung von IT-Landschaften durch spezifische Vorgaben der Sprachspezifikation. Die Qualität sowie die Anschaulichkeit der Modelle werden in der Folge tendenziell gefördert und deren Operationalisierbarkeit, d.h. bspw. ihre Verwendung im Rahmen von Analysen, vereinfacht (s. z.B. [EsJa01] und [LKT04]).

[1] S. [OMG04] und [OMG05].

Die Spezifikation einer Modellierungssprache kann entweder informal, formal oder semiformal erfolgen. Metamodelle[1] als Vertreter der semiformalen Spezifikationsansätze weisen verglichen mit anderen Herangehensweisen den Vorzug auf, eine werkzeugbasierte Überprüfung der Korrektheit und Integrität von Modellen effizient zu unterstützen (vgl. [FrPr97], S. 24) sowie einen Paradigmenbruch zwischen der Sprachdefinition und der Sprachanwendung zu vermeiden (vgl. [FrLA03], S. 95). Aufgrund der hier skizzierten Vorzüge wurde die Entscheidung getroffen, der Herausforderung des Entwurfs von Beschreibungskonzepten für IT-Landschaften im Kontext des IT-Managements durch die Spezifikation einer DSL in Form eines Metamodells zu begegnen.

In den verbleibenden Kapiteln des Berichts werden zunächst die grundlegenden Aufgaben und Ziele des IT-Managements und verwandter Themenbereiche aufgearbeitet. Im Anschluss werden darauf aufbauend Anforderungen an eine Sprache zur Modellierung von IT-Landschaften formuliert, bevor die aktuelle Spezifikation der Sprache vorgestellt wird. Abschließend erfolgt eine Zusammenfassung sowie ein Ausblick auf zukünftige Forschungstätigkeiten.

[1] Der Begriff *Metamodellierung* bezeichnet den Vorgang der Erstellung eines Modells einer Modellierungssprache. Somit ist ein *Metamodell* ein Modell aller Modelle, die mit einer Sprache dargestellt werden können. Es definiert abstrakte Syntax und Semantik der Sprache (vgl. [KiJu06], S. 307, [GKP98], S. 1 und [GiHi05], S. 1).

2 Zentrale Begriffe des IT-Managements

IT-Management als vornehmlich in der Praxis entstandene Begrifflichkeit bedarf einer sorgfältigen inhaltlichen Eingrenzung, damit die hiermit verbundenen Aufgaben und Ziele deutlich werden. Dies erfolgt grundlegend in den nachfolgenden Abschnitten des aktuellen Kapitels. Zunächst werden die Grundbegriffe der Informationstechnik hergeleitet. Hier sind insbesondere die Termini Informationssystem, Software und Hardware, aber auch Daten und Information zu nennen. Im Anschluss wird das IT-Management anhand der einschlägigen Literatur betrachtet und gegen IT-Governance abgegrenzt. Weiter wird der IT-Service-Begriff erläutert sowie ein Überblick über mögliche Ansätze zur Ermittlung von Kosten und Nutzen von IT gegeben. Abschließend erfolgt eine Zusammenfassung mit der Vorstellung eines Handlungsrahmens für das IT-Management, in dem alle grundlegenden Aufgaben und Konzepte zusammengeführt werden.

2.1 Informationstechnik und Informationssysteme

Der Fokus der Aktivitäten des IT-Managements ist auf die *Information Technology* bzw. *Informationstechnik* (IT) eines Unternehmens gerichtet (s. Abschnitt 2.2). Daher ist es von grundlegender Wichtigkeit, diesen Untersuchungsgegenstand begrifflich abzugrenzen. IT ist innerhalb eines Unternehmens üblicherweise in Systemen organisiert, welche jeweils einem dedizierten Zweck zugeordnet werden können. Man spricht in diesem Zusammenhang von IT-Systemen, aber auch synonym von Elektronischen Datenverarbeitungssystemen (EDV bzw. DV-Systeme), Informationsverarbeitungssystemen (IV-Systeme), Informationssystemen (IS) oder Informations- und Kommunikationstechnik (IKT), um nur einige der verbreiteten Begrifflichkeiten zu nennen. In dieser Arbeit verwenden wir vornehmlich die Begriffe *IT-Landschaft* sowie *Informationssystem*, um die in unserem Kontext relevanten Erkenntnisobjekte terminologisch einzugrenzen. Unter einer IT-Landschaft verstehen wir die Gesamtheit der betrieblichen Datenverarbeitungs- und Kommunikationssysteme; d.h. sowohl Hardware als auch Software inkl. der Beziehungen und Schnittstellen zur betreibenden Organisation sowie zur betrieblichen Umwelt. Das Informationssystem differenziert sich von einer IT-Landschaft durch einen eher anwendungsorientierten als technischen Fokus. Es bildet als Teil einer IT-Landschaft ebenfalls eine Abstraktion über Hard- und Software sowie organisatorische Aspekte, wird aber hauptsächlich hinsichtlich seiner betrieblichen Funktion betrachtet. Stellvertretend für eine Menge von inhaltlich ähnlich ausgerichteten Definitionen[1] stellen Laudon et al. diese Sichtweise folgendermaßen dar: „Ein Informationssystem enthält […] Anwendungssoftware und Daten und ist in die Organisations-, Personal- und Technikstrukturen des Unternehmens eingebettet." ([LLS06], S. 31). Oftmals wird von einem Anwendungssystem gesprochen, wenn lediglich Bezug zur technischen Basis eines Informationssystems genommen wird (s. z.B. [LLA], S. 31 und [Mert01], S. 46-46). Der organisatorische Kontext wird hier nur teilweise in der Form der Prozesse und Aufgaben des Systems berücksichtigt. Aspekte der Aufbauorganisation und des System-Managements fehlen. Die vom System zu verarbeitenden Daten hingegen werden von beiden Begriffen berücksichtigt.

Scheer schlägt eine Kategorisierung von Informationssystemen gemäß ihrer primären betrieblichen Funktion in Administrations-, Dispositions-, Management- und Planungssysteme vor (s. [Sche95], S.4 ff.). Weiterhin unterscheidet er zwischen Systemen, welche die operativen Aktivitäten im Unternehmen unterstützen, und solchen, die eher wertorientiert sind (Abrechnungs- und Controllingsysteme sowie Systeme zur Unterstützung der langfristigen Planung). Allerdings ist diese Art der Kategorisierung gemäß Scheer eher als didaktische Hilfe zu verstehen, denn als ernsthafter Versuch einer Definition.

[1] Z.B. [GRB04], S. 1, [Mert01], S. 499.

Damit Informationssysteme ihre dedizierten Aufgaben ausführen können, müssen sie zunächst in der Lage sein, Daten zu verarbeiten. Diese Daten liegen in der Regel elektronisch vor, können aber auch in der Form von Akten oder anderen Schriftsätzen abgelegt sein. Daten lassen sich von Informationen definitorisch dahingehend abgrenzen, dass sie zwar gemäß einer festgelegten *Syntax* strukturiert sind, aber nicht ohne weiteres vom Menschen interpretierbar bzw. lesbar sein müssen (vgl. [LLS06], S. 32). Die Erweiterung des Datenbegriffs zum Informationsbegriff inkludiert hingegen eine potenzielle Nützlichkeit für und Interpretationsmöglichkeit durch den menschlichen Betrachter (vgl. [LLS06], S. 32.), d.h. ein Bezug der Daten zu existierenden oder realweltlichen Objekten wird hergestellt. Formal wird innerhalb eines Begriffssystems, der *Semiotik*, neben der Syntax auch die Eigenschaft der *Semantik* und *Pragmatik* von Zeichenketten festgelegt (vgl. [GRB04], S. 31-32). Die Semantik erweitert die Syntax um eine Bedeutung, welche sich für den Betrachter der Daten ergibt. Die Ebene der Pragmatik berücksichtigt zusätzlich noch den Hintergrund des Empfängers. Es wird angenommen, dass er aufgrund seiner Kenntnisse und Perspektiven eine bestimmte Interpretation der Daten vornimmt. Die Aspekte der Semiotik zusammen betrachtet führen zu einer möglichen Interpretation des Informationsbegriffs. *Wissen* verfügt im Vergleich zu Informationen über weiterführende Merkmale. Hierzu zählen z.B. der Handlungsbezug, d.h. Wissen entsteht durch aktive Auseinandersetzung eines Individuums mit der Umwelt, oder der Sozialbezug (Wissen entsteht innerhalb sozialer Beziehungen) des Individuums, welche die Information aufnimmt (vgl. [LHM95], S. 208). Dies soll aber an dieser Stelle nicht weiter ausgeführt werden, da es für den Kontext der Arbeit nicht von direktem Interesse ist. Stattdessen sei der Leser bspw. an [LHM95], S. 165 ff. verwiesen. Dort findet sich eine ausführliche Diskussion der Begriffe Daten, Information und Wissen, welche jeweils aus verschiedenen wissenschaftlichen Perspektiven heraus betrachtet werden.

Um eine Daten- bzw. Informationsverarbeitung realisieren zu können, bauen Informationssysteme bzw. IT-Landschaften auf verschiedenen IT-Komponenten auf. Diese können prinzipiell in *Hardware* und in *Software* untereilt werden. Unter Hardware verstehen wir tangible Geräte, wie Computer oder Drucker sowie deren Komponenten (z.B. Grafikkarte, Speicher). Netzwerke als Aggregation über eine Menge von Hardwaregeräten fallen ebenfalls unter diesen Begriff. Hardware bildet die Basis zum Betrieb von Software. Software bezeichnet intangible (immaterielle) Komponenten, welche zum Betrieb von Informationssystemen nötig sind (vgl. [GRB04], S. 6-7). Unter einem *Softwareartefakt* verstehen wir einen identifizierbaren Teil einer Software, der einer Hardwarekomponente zugeordnet werden kann. Weitere Definitionen zu dem Begriff Software finden z.B. sich in [Balz00], S.23).

Nach der Abgrenzung von mit dem Begriff der Informationstechnik eng verknüpfter Konzepte, wie Informationssystem, Daten, Information und Wissen sowie Hard- und Software, folgt im nächsten Abschnitt eine Betrachtung und Einordnung existierender Definitionen und Aufgabenbeschreibungen für das IT-Management.

2.2 IT-Management

Der Begriff *IT-Management* wird in der Literatur nicht einheitlich definiert. Jedoch lassen sich bei den jeweils beschriebenen Aufgaben des IT-Managements inhaltliche Überschneidungen feststellen. Einige dieser Aufgaben decken sich mit denjenigen, die auch dem *Informationsmanagement* (IM) zugeordnet werden. Dies zeigt sich z.B. in der folgenden Aussage von Zarnekow und Brenner: „Das IT-Management, häufig auch als Informationsmanagement bezeichnet, beschäftigt sich als Teil der Unternehmensführung mit der Erkennung und Umsetzung der Potenziale der Informations- und Kommunikationstechnologie in Lösungen." ([ZaBr03], S.7). Hier wird die Anforderung an das IT-Management deutlich, strategische Potenziale der IT nutzbar zu machen. Stahlknecht und Hasenkamp definieren IT-Management als eine dem Informationsmanagement untergeordnete Aufgabe, die sich mit der Planung, der Beschaffung und der Implementierung von IT-Infrastrukturen befasst, welche die Informationsbeschaffung unterstützen sollen (s. [StHa05], S. 37). Strategisches IT-

Management wird als die Menge von Aufgaben beschrieben, welche die organisatorische und technologische Planung sowie die effiziente und effektive Implementierung von IT-Infrastrukturen unter Berücksichtigung der Unternehmensstrategien zum Ziel hat (vgl. auch [Krcm00], S. 155). Dem IT-Management wird teilweise eine Rolle zugeschrieben, die einerseits in enger Kooperation mit der Geschäftsleitung wahrgenommen werden muss, andererseits aber auch die Ausführung von Aufgaben wie Administration und Implementierung beinhaltet.

Oftmals erfolgt eine Untergliederung der Aktivitäten des IT-Managements in die Phasen Planung, Entwicklung und Produktion von IS und IT-Infrastrukturen. Zusätzlich wird das *IT-Controlling* als Querschnittsaufgabe identifiziert (s. [ZaBr03], S. 8). Horváth gibt hier zu bedenken, dass der Aufgabenbereich des IT-Controllings nicht klar definiert und schwierig gegen das IM abzugrenzen ist. Als Kernaufgaben des IT-Controllings werden gleichermaßen die IT-Planung, aber auch Kontrolle und Informationsversorgung genannt (vgl. [Horv03], S. 720). Grob et al., Pietsch et al. und Stickel rechnen das IT-Controlling ebenfalls zum IT-Management und ordnen ihm ähnliche Aufgaben zu (s. [GRB04], S. 392, [PMK04], S. 82 ff. und [Stic01], S. 18). Kütz beschreibt als zentrale Aufgabe des IT-Controllings[1] die Beschaffung der Informationen, welche es dem Management ermöglichen, Entscheidungen in Bezug auf die IT fundiert und mit größtmöglicher Sicherheit zu treffen (s. [Kuet03], S. 51). Vor dem Hintergrund dieser Ausführungen interpretieren wir im Folgenden das IT-Controlling als integralen Teil des Aufgabenportfolios des IT-Managements.

Heinrich fasst Aufgaben, wie das Beobachten der Technologie, das Beeinflussen der Technologieentwicklung, das Bestimmen und Decken des Technologiebedarfs, die Evaluierung des Technologieeinsatzes (ex post) sowie das Verwalten des Technologiebestands eines Unternehmens mit dem Begriff *Technologiemanagement* zusammen (s. [Hein99], S. 156). Als Planungsaspekte werden die Infrastrukturplanung und die IS-Planung genannt. Mertens et al. verwenden als Zusammenfassung für die Aufgaben im Kontext des Managements von IT den Begriff *Management der Informationsverarbeitung* (vgl. [MBK+05], S. 179). Sie differenzieren zwischen strategischer Planung, der Organisation der Informationsverarbeitung und weiteren Managementaspekten. Zu den strategischen Aufgaben zählen die Definition einer IT-Strategie[2], das Festlegen einer IT-Architektur sowie die Auswahl von IT-Projekten. Organisatorische Aufgaben befassen sich mit der Einordnung der IT-Organisation ins Unternehmen sowie der damit verbundenen Entscheidung, auf welche Art anfallende IT-Kosten zu verrechnen sind. Sonstige Aufgaben drehen sich hauptsächlich um die Umsetzung gesetzlicher Rahmenbedingung (z.B. Planung und Implementierung von Lösungen zur Datensicherheit). Auch hier ist eine inhaltliche Deckung mit den bisher aufgeführten Wendungen des IT-Managements zu erkennen, sodass die Begriffe Technologiemanagement und Management der Informationsverarbeitung unter IT-Management subsumiert werden können.

In der Praxis sind aktuelle Ansätze des IT-Managements, von denen vor allem ITIL stetig an Bedeutung gewinnt, stark auf den Aspekt des *IT-Service-Managements* fokussiert. Ausgehend von dem Ziel einer IT-Organisation, die wichtigsten Geschäftsprozesse optimal zu unterstützen, leitet sich die zentrale Idee des IT-Service-Management ab (vgl. [Somm04], S. 27). Diese besteht darin, externe als auch interne IT-Organisationen als Dienstleister zu betrachten, deren Aufgabe die Lieferung von qualitativ hochwertigen und kostengünstigen IT-Services ist. In diesem Zusammenhang wird die permanente Verbesserung von Qualität und Wirtschaftlichkeit der angebotenen IT-Services durch ein entsprechend zielgerichtetes Management der IT-Infrastruktur verfolgt. Dies soll u.a. durch die Verwendung von Referenzprozessen in IT-Organisationen in der Form von Best Practices erreicht werden, welche das Vorgehen bei der Lösung von IT-spezifischen Aufgaben verbindlich vorschreiben.

[1] In der Quelle wird diese Definition zuerst für Controlling gegeben, bevor sie im Anschluss auch für IT-Controlling gültig erklärt wird.

[2] Hand definiert eine IT- bzw. IS-Strategie bspw. folgendermaßen: „An information system strategy is a statement of senior management commitment at all levels to a process of business change that will achieve specific business benefits, in which information systems and technology will play a major (possibly dominant) role." [Hand92].

Durch die Ausrichtung des Managementfokus auf die Dienstleistungen und Geschäftsprozesse im IT-Service-Managementbereich wird ein eher kunden- statt technikorientiertes IT-Management angestrebt (vgl. [BRS02], S.1). Hier ist der primäre Vorteil darin zu sehen, dass die Anforderungen durch die Geschäftsprozesse der Kunden in den Vordergrund gestellt werden und das IT-Management somit eine aktive Rolle bei der Erreichung einer hohen Kundenzufriedenheit spielen kann.

Ein weiterer Begriff, der im Kontext der vorliegenden Arbeit von Interesse ist, ist *Architekturmanagement*. Hier stehen die langfristige Planung und Weiterentwicklung von IT-Architekturen[1] im Fokus der Managementaktivitäten (vgl. [Bren94], S. 107). Die IT-Architektur wird in der Regel von strategischen Vorgaben diktiert und ist mit der Geschäftsarchitektur abzugleichen (vgl. [Foeg03], S. 63.). Hierzu werden Prozesse, Rollen und deren Verantwortlichkeiten im Unternehmen festgelegt, sodass die Dauerhaftigkeit der Wirkung gewährleistet ist. Das Architekturmanagement ist oftmals den Aktivitäten des IT-Managements zugeordnet. So wird es bspw. in ITIL im Rahmen des IT-Infrastrukturmanagements behandelt (vgl. [ITIL02]). Wir interpretieren das Architekturmanagement ebenfalls als Teil des Aufgabenportfolios des IT-Managements. Insbesondere die Gestaltung der IT-Infrastrukturen gemäß den Vorgaben der Strategien des Unternehmens sind Aspekte, die somit dem IT-Management als Schwerpunkt hinzugefügt werden.

Allen oben aufgeführten Beschreibungen der Aufgaben des IT-Managements bzw. verwandter Begriffe ist zu entnehmen, dass sich das Management der IT nicht nur auf die betriebliche Anwendungslandschaft bezieht, sondern auch explizit die zum Betrieb der Informationssysteme notwendige Hardware mit berücksichtigt. Dies steht im Kontrast zu früheren Ansätzen, wie z.B. dem *St. Galler Informationssystem-Management*, in dem von der hardwaretechnischen Basis explizit abstrahiert wird (vgl. [OBH92], S. 30). Weiterhin ist festzustellen, dass in der Praxis die Serviceorientierung und die Einführung der Dienstleister- und Kundenmetaphern eine mittlerweile etablierte Herangehensweise an die Aufgaben des IT-Managements sind. Dieser Aspekt wird daher im weiteren Verlauf der Arbeit weiterführend aufgegriffen.

2.3 IT-Governance

Ein Begriff, der in einem engen Bezug zum IT-Management steht, von diesem aber meist nicht klar angegrenzt wird, ist *IT-Governance*. Die Vielfalt der existierenden Definitionen für IT-Governance ist ähnlich der des IT-Managements, allerdings sind diese oftmals weniger präzise gehalten. Dies soll die folgende Auswahl von Definitionsversuchen illustrieren (vgl. [Pete04], S. 41).

- "IT Governance is the responsibility of the Board of Directors and executive management. It is an integral part of the enterprise governance and consists of the leadership and organisational structures and processes that ensure that the organisation´s IT sustains and extends the organisation´s strategy and objectives." IT Governance Institute, 2001
- "IT Governance is the organisational capacity exercised by the Board, executive management and IT management to control the formulation and implementation if IT strategy and in this way ensure the fusion of business and IT." van Grembergen, 2002
- „IT Governance describes a firm´s overall process for sharing decision rights about IT and monitoring the performance if IT investments." Weill & Vitale, 2002
- "IT Governance defines the locus of enterprise decision-making authority for core IT activities." Sambamurthy & Zmud, 2000

[1] Eine mögliche Definition des Begriffs *Architektur* bietet die IEEE an: Architecture is "The fundamental organization of a system, embodied in its components, their relationships to each other and the environment, and the principles governing its design and evolution" ([MEH01], S. 108).

- "IT Governance refers to the patterns of authority for key IT activities." Sambamurthy & Zmud, 1999
- "IT Governance is the degree to which authority for making IT decisions is defined and shared among management, and the processes managers in both IT and business organizations apply in setting IT priorities and the allocation of IT resources." Luftman, 1996
- "IT Governance describes the locus of responsibility for IT functions." Brown & Magill, 1994

Eine Schnittmenge zwischen den obigen Definitionen wird durch die Aussage gebildet, dass IT-Governance die Verantwortlichkeiten für die IT im Unternehmen festlegt. Diese Aufgabe wird oftmals als Aufgabe des führenden Managements bzw. des Vorstands gesehen. Inhaltlich steht die Gestaltung von Managementprozessen und der IT-Organisation zur Unterstützung der Strategien und Unternehmensziele im Mittelpunkt (s. hier auch [Cobi06b], S. 5).

Die Abgrenzung der Aufgaben, welche der IT-Governance bzw. dem IT-Management zugeordnet werden können, gestaltet sich schwierig. Es wird mitunter versucht dahingehend zu differenzieren, dass IT-Governance eher strategisch und kundenorientiert ausgerichtet ist (vgl. [GHG04], S. 4). Allerdings ist gerade die Kundenorientierung der zentrale Aspekt des Service-Managements als Bestandteil des IT-Managements, sodass dieses Kriterium nicht zur Abgrenzung der Begriffe geeignet scheint. Weill und Ross unternehmen einen weiteren Differenzierungsversuch, indem sie IT-Management als eine Menge von Aktivitäten innerhalb der IT-Governance charakterisieren (s. [WeRo04], S. 39). IT-Management-Services koordinieren demnach die integrierte Unternehmensinfrastruktur und sind für das Management der Beziehungen zwischen den Geschäftseinheiten zuständig. Typische Managementdienste beinhalten IS-Planung, Projektmanagement, Service Level Agreements (SLAs, s. auch Abschnitt 2.4) und Verhandlungen mit Herstellern und Outsourcern. IT-Governance wird hier als das „Spezifizieren von Entscheidungsbefugnissen und Verantwortlichkeiten zur Förderung erstrebenswerten Verhaltens in Bezug auf die Nutzung von IT." beschrieben (s. [WeRo04], S. 9). Diese intendiert unscharfe Definition soll die herausragende Funktion hervorheben, die IT-Governance in einem Unternehmen innehat. Eine echte Abgrenzung zum IT-Management leistet sie allerdings nicht.

Im Vergleich mit den Aufgabenbeschreibungen des IT-Managements in Abschnitt 2.2 ist keine durchgehend eindeutige Differenzierung zwischen IT-Management und IT-Governance festzustellen (vgl. [MZK03], p. 446). Jedoch ist die Frage nach der Verantwortung und den Zuständigkeiten beim IT-Governance deutlicher ausgeprägt als beim IT-Management, sodass ein eher institutioneller Charakter mit Fokus auf den formalen Aufbau einer IT-Organisation zu konstatieren ist. IT-Governance kann als Rahmen interpretiert werden, in dem das IT-Management stattfindet und reguliert wird. Davon abgesehen sind die Aufgaben des IT-Governance untrennbar mit denen des IT-Managements verwoben. Daher werden wir im Verlauf dieser Arbeit auf IT-Governance größtenteils nicht mehr explizit eingehen, sondern es implizit im Rahmen unserer Interpretation des IT-Managements (s. insbesondere Abschnitt 2.7) berücksichtigen.

2.4 IT-Service

Betrachtungsgegenstand dieses Abschnitts ist der Begriff *Service* bzw. dessen Spezialisierung *IT-Service*. In der Softwaretechnik, insbesondere der objektorientierten Softwareentwicklung, beschreibt ein Service die Dienstleistung, die eine Klasse für einen Anwender, d.h. die den Service aufrufende Klasse, bereitstellt (vgl. [Balz00], S. 817). Auch Softwarekomponenten bieten ihre Funktionalität nach außen hin über Schnittstellen an (s. [Balz00], S. 856). Hier wird analog zu Klassen die Service-Metapher verwendet.

Innerhalb des zurzeit populären Paradigmas der *Serviceorientierten Architektur* (SOA) bezeichnen Services logische Einheiten, welche die Geschäftslogik von Anwendungen kapseln und in Form von Diensten anderen Anwendungen zur Verfügung stellen (vgl. [Erl05], S. 33). Eine SOA ist demnach eine technische Architektur, welche durch ihre Komponenten die Implementierung einer Serviceorientierung ermöglicht (vgl. [Erl05], S. 54). *Web Services* sind eine besondere Form von Services innerhalb einer SOA, die durch herstellerunabhängige Standards beschrieben (WSDL), genutzt (SOAP) und verwaltet (UDDI) werden (vgl. [Erl05], S. 111).

Unabhängig vom Kontext der Softwareentwicklung kann ein Service allgemein als ein Nutzeneffekt charakterisiert werden, der von einem Dienstleister für einen Kunden erbracht wird (s. [HBS06], S. 10 und [NCT+05], S. 14-15). Der Kunde nutzt das Ergebnis des Services zur Unterstützung bzw. Realisierung seiner Geschäftsprozesse. Köhler definiert den Service ähnlich und ergänzt die Bedingung, dass die Qualität und Quantität eines Services messbar sein muss. Er verwendet für die Beschreibung dieser Aspekte eines Services den Begriff *Service Level* (s. [Koeh05], S. 30). Vor dem Hintergrund der vorliegenden Arbeit werden wir im weiteren Verlauf keine Differenzierung zwischen den Begriffen Service und IT-Service vornehmen. Da sich im Bereich des IT-Managements in der Regel alle Services mittelbar oder unmittelbar auf IT beziehen, verwenden wir ausschließlich den Begriff Service.

Böhmann unterscheidet zur Kategorisierung von Services primär zwischen zwei Dimensionen, der *Leistungserstellung* und dem *Leistungsergebnis*. Die Leistungserstellung bezieht sich auf die Art und Weise, wie Services generiert werden, d.h. insbesondere in welchem Umfang IT beteiligt ist. Das Leistungsergebnis beschreibt den Output der Leistung. In diesem Zusammenhang ist von Interesse, in welchem Umfang der Output auf IT-Systeme zielt. Hieraus resultiert die Einteilung von Services in die folgenden drei Hauptkategorien (s. [Böhm04], S. 32):

- *IT-Dienstleistungen im weiteren Sinne*: IT-Aktivitäten oder IT-Systeme sind wesentlicher Bestandteil des Leistungsergebnisses
- *IT-Dienstleistungen im engeren Sinne*: IT-Aktivitäten oder IT-Systeme sind wesentlicher Bestandteil des Leistungsergebnisses, zusätzlich existiert eine starke Einbindung von IT-Systemen oder anderen IT-bezogenen Faktoren in den Leistungserstellungsprozess
- *IT-basierte Dienstleistungen*: IT-Systeme sind nicht Ziel des Leistungsergebnisses, aber es existiert eine starke Einbindung von IT-Systemen oder anderen IT-bezogenen Faktoren in den Leistungserstellungsprozess

Als zentrale Eigenschaften, welche im Zusammenhang mit Services und im Kontext der Arbeit von Belang sind, können die folgenden betrachtet werden (vgl. [HBS06] S. 11 ff.):

- *Service-Funktionalität*: Funktionen der IT, welche die Bereitstellung eines Services ermöglichen (z.B. Druckfunktion eines Druckers)
- *Service-Qualität*: Eigenschaften wie Verfügbarkeit, Transaktionszeit, maximale Ausfalldauer
- *Service- Einheit*: Verrechnungseinheit eines Services (z.B. Nutzungsminuten)
- *Service-Ergebnis*: Endergebnis des erbrachten Services (z.B. gedrucktes Textdokument)
- *Service-Wirkung*: Auswirkungen des Service-Ergebnisses (z.B. Kundenzufriedenheit)
- *Service-Capability*: Produktionsbereitschaft für einen definierten Service (gesamte Infrastruktur)
- *Service-Nutzer*: Rolle, die einen Service in Anspruch nimmt
- *Service-Kunde*: Rolle, die einen Service beauftragt
- *Service-Provider*: Organisationseinheit, die einen Service bereitstellt

Eine Differenzierung zwischen einem Service und einem IT-Produkt (im Folgenden analog zu IT-Service und Service *Produkt* genannt) wird oftmals nicht durchgeführt oder recht unterschiedlich ge-

handhabt. So kann ein Produkt bspw. dahingehend von einem Service unterschieden werden, dass es u.a. materiell, identisch reproduzierbar, wiederholt nutzbar, lagerbar und beständig ist (vgl. [HBS06], S. 23). Dies trifft auf Hardware und bedingt auf Software zu, welche in diesem Sinne Produkte sind. Services hingegen sind u.a. immateriell, variierend, nur einmal nutzbar, nicht lagerbar und flüchtig. Dies trifft auf Dienstleistungen, wie Druckdienste, Datenbankzugriffsdienste etc. zu. Durch die o.g. Charakterisierung werden Ausprägungsmerkmale zweier Extrema definiert. In der Realität beinhalten Dienstleistungsangebote Merkmale sowohl von Produkten als auch von Services. Als Beispiel hierfür kann der Vertrieb von Hardwaregeräten inkl. Serviceleistungen, wie Einbau oder Wartung gelten. Durch die fehlende Möglichkeit einer scharfen Trennung der Begriffe sehen wir analog zu Zarnekow und Brenner Produkte lediglich als logische Zusammenfassung von Services, welche in dieser Form dem Kunden angeboten werden (vgl. [ZaBr03], S. 10). Die Autoren schlagen vier Kategorien zur Differenzierung von Produkten vor:

- *Stufe 1*: Ressourcenorientierte Produkte (z.B. Rechenleistung)
- *Stufe 2*: Informationssystemorientierte Produkte (z.B. Anwendung zur Rechnungsstellung)
- *Stufe 3*: Prozessorientierte Produkte (z.B. IT-Unterstützung des Rechnungsprozesses)
- *Stufe 4*: Geschäftsproduktorientierte Produkte (z.B. Internetzugang, E-Ticket)

Ressourcenorientierte Produkte ermöglichen dem Kunden lediglich den Zugriff auf IT-Ressourcen in der Form von Druckdiensten, Rechenleistung etc. Informationssystemorientierte Produkte enthalten bereits Funktionen zur Unterstützung von geschäftlichen Abläufen. Prozessorientierte Produkte unterstützen mit zusätzlich enthaltener Geschäftslogik komplette Geschäftsprozesse. Geschäftsproduktorientierte Produkte schließlich können komplette Geschäftsbereiche abdecken und bedienen. Die Geschäftsorientierung eines Produkts steigt ausgehend von Stufe 1 nach Stufe 4 an.

IT-Produkte durchlaufen analog zu klassischen Produkten einen Lebenszyklus. Zarnekow und Brenner lassen hier allerdings offen, ob zu dessen Beschreibung ein typischer Produktlebenszyklus oder der eines Informationssystems genutzt werden soll (s. [ZaBr03], S. 11). Ersterer wird oftmals in die Phasen Entwicklung, Einführung, Wachstum, Reife und Rückgang beschrieben. Der Lebenszyklus eines Informationssystems enthält zumindest die Phasen Planung, Entwicklung, Produktion und Außerbetriebnahme, kann aber durchaus feingranularer dargestellt werden. Die Betrachtungsweise hängt u.a. von der Art des Produkts ab.

Die Qualität von Services und Produkten sollte möglichst klar definiert und messbar sein. Generell lässt sich Qualität als „die Gesamtheit von Eigenschaften und Merkmalen eines Produktes oder einer Tätigkeit, die sich auf deren Eignung zur Erfüllung gegebener Erfordernisse beziehen" bezeichnen (aus DIN 55350, Teil 11). Übertragen auf Services lässt sich feststellen, dass die Servicequalität dann gut ist, wenn ein Service bzw. Produkt die in den Anforderungen beschriebenen Eigenschaften erfüllt (vgl. [WoMü03], S. 68). Diese Eigenschaften werden üblicherweise in Form eines *Service Level Agreements* (SLA) festgehalten, welcher zwischen dem Anbieter des Service und dem Kunden ausgehandelt wird. Einige zentrale Aspekte, die ein SLA typischerweise beschreibt, werden im Folgenden aufgelistet (vgl. [ITIL03a], Annex 4D):

- Anbieter
- Kunde
- Datum
- Service-Beschreibung
    - Verfügbarkeitszeiten (z.B. täglich 9-17 Uhr)
    - Garantierte Verfügbarkeit (z.B. 99,5%)
    - Zuverlässigkeit (z.B. max. Anzahl tolerierbare Ausfälle)
    - Kundenservice (Umfang des Supports)
    - Serviceleistungsgrad (z.B. Antwortzeiten)

- Funktionalität
- Vorgehen im Falle von Änderungen
- IT Service Continuity (geplante Sicherheitsmaßnahmen, wie Backups)
- Sicherheit (Maßnahmen zum Schutz gegen Missbrauch, Schäden durch Viren etc.)
- Drucken (spezieller Hinweis auf Druckdienste)
- Verrechnung (Abrechungs- und Strafzahlungsdetails)
- Service Reviews (Vorgehen im Falle periodischer bei Begutachtung des Services)
- Glossar (Erläuterung von Fachausdrücken und Abkürzungen)
- Anhang (verschiedenes)

In der Praxis verwendet eine IT-Organisation SLAs, die auf spezielle Kundenbedürfnisse und die jeweiligen Produkte zugeschnitten sind.

Die Metapher des Services erscheint als ein wichtiges Konzept, um die Unterstützung der Geschäftsprozesse eines Kunden durch die IT eines Dienstleisters sinnvoll zu beschreiben. Diese Sichtweise ist sowohl hinsichtlich der Leistungserbringung für externe als auch für interne Kunden sinnvoll. So ist der Prozess der Leistungserstellung in der Regel unabhängig davon, ob der Kunde der IT-Organisation, die den Service anbietet, eine andere Abteilung innerhalb des eigenen Unternehmens oder ein anderes Unternehmen ist. Entscheidend für die Produktion, Breitstellung und Abrechnung ist lediglich ein entsprechender Vertrag zwischen den Partnern, der die Modalitäten der Leistungserbringung festlegt. Auf diese Weise werden Funktionalität und Qualität der Services verbindlich definiert.

2.5 IT-Kostenrechnung

In vielen Unternehmen ist einer stetiger Anstieg der IT-Kosten an den Gesamtkosten festzustellen (vgl. z.B. [Horv03], S .741). Diese Kosten verursachungsgerecht zuzuordnen ist eine Herausforderung, die aufgrund der Komplexität der Interdependenzen zwischen den betroffenen IT-Komponenten, den angebotenen Dienstleistungen, den unterstützen Geschäftsprozessen sowie den Kunden oftmals nicht zufrieden stellend gemeistert wird. Eine solche Zuordnung ist allerdings Vorraussetzung für die Schaffung eines Kostenbewusstseins bzw. einer Transparenz, welche grundlegend für die Planung eines effizienten Einsatzes der IT ist. Des Weiteren wird eine Informationsbasis benötigt, die es erlaubt, die entstehenden Kosten mit den Kunden respektive Nutzern der jeweiligen Services abzurechnen. Als Nutzer kommen hier sowohl interne, z.B. andere Abteilungen, als auch externe Kunden – Einzelpersonen oder andere Organisationen – in Frage. Eine Empfehlung für die Wahl eines bestimmten Vorgehens bei der IT-Kostenrechnung soll hinsichtlich der angedeuteten, besonderen Komplexität der Domäne an dieser Stelle nicht gegeben werden, zumal hier noch weitere, unternehmensspezifische Einflussfaktoren zu beachten sind. Ein solcher Faktor ist bspw. die Einordnung der IT-Organisation ins Unternehmen, welche in der Form eines *Profit Centers*, *Cost Centers* oder *Recovery Centers* (auch gelegentlich als *Systemhaus* bezeichnet) erfolgen kann. Das Profit Center bezeichnet den Fall, dass die IT-Dienstleistungen gewinnorientiert angeboten werden. Das Cost Center bezieht sich auf die bloße Verrechnung von IT-Kosten, wogegen in einem Recovery Center auch deren Weitergabe an die Nutzer berücksichtigt wird. Konform zur Positionierung der IT-Organisation muss festgelegt werden, welche Granularität die Kostenrechnung aufweisen soll, d.h. inwieweit in Anspruch genommene Dienste weiter auf die beteiligten IT-Komponenten aufgeschlüsselt werden. Auch sollte die Kompatibilität einer Kostenrechnungsmethode mit den im Unternehmen eingesetzten IT-Managementansätzen sichergestellt sein. Kostenmanagement und damit Kostenrechnung ist eine wichtige Aufgabe des IT-Managements und muss daher gemeinsam mit diesem betrachtet werden. Da der Fokus der vorliegenden Arbeit auf den Entwurf von Beschreibungskonzepten für das IT-Management gerichtet ist, sollen im Folgenden die zentralen Begriffe gängiger Kosten-

rechnungsansätze in Kürze aufgearbeitet werden, soweit sie für den Kontext der Arbeit relevant sind.

Eine grundlegende Unterscheidung der Kostenrechnungsansätze begründet sich im zeitlichen Bezug der betrachteten Kosten. So werden in der *Ist-Kostenrechnung* tatsächlich realisierte Kosten einer abgeschlossenen Rechnungsperiode erfasst und verrechnet (s. [Frie04], S. 67). Die *Normalkostenrechnung* bezieht sich ebenfalls auf bereits angefallene Kosten, bedient sich aber als Grundlage der Berechnung nicht effektiver Ist-Kosten, sondern aus Gründen der Vereinfachung fester, normalisierter Verrechnungssätze (s. [Eise02], S. 769). Die *Plankostenrechnung* schließlich dient der Bereitstellung von Informationen über zu erwartende Kosten in einer zukünftigen Abrechnungsperiode und stellt damit ein System zur Sollkostenrechnung dar (s. [Frie04], S. 68).

Eine weitere Differenzierungsmöglichkeit ist der Umfang der verrechneten Kosten (vgl. [Frie04], S. 68 ff und Eise02], S. 643). Hier sind vor allem die *Vollkostenrechnung* und die *Teilkostenrechnung* zu unterscheiden. In der Vollkostenrechnung werden die gesamten Kosten einer Periode auf die Bezugsobjekte verrechnet. Die Teilkostenrechnung bezieht sich lediglich auf die variablen Teile der Gesamtkosten, während die Fixkosten den Bezugsobjekten nicht zugerechnet werden (s. [Eise02], S. 748). Somit wird berücksichtigt, dass Fixkosten nicht durch die Erstellung einer konkreten Leistung anfallen, sondern durch die Leistungserstellung insgesamt entstehen.

Unabhängig von der Art der Kostenrechnung lassen sich drei Teilrechnungen identifizieren, welche entlang des Abrechnungswegs genutzt werden können: die *Kostenartenrechnung*, die *Kostenstellenrechnung* und die *Kostenträgerechnung*. Die Kostenartenrechnung dient der Gliederung der erfassten Kosten in rechnungszielorientierte Kostenkategorien (s. [Frie04], S. 69). Im Kontext von IT bieten sich bspw. die Kategorien Hardware, Software, Personal, Miete, externe Dienstleistungen oder Transfers an (vgl. z.B. [ITIL03a]), die jeweils weiter in untergeordnete Kostenarten untergliedert werden können. Die Nutzung externer Dienstleistungen verursacht Kosten, da sie von externen Organisationen in Rechnung gestellt werden. Transfer bezieht sich auf die Nutzung von Dienstleistungen oder den Erwerb von Hard- und Software von anderen Organisationseinheiten desselben Unternehmens. Personalkosten sind in der Form von Arbeitsstunden zu interpretiert, die in Serverbetrieb, Support für Applikationen, Reparaturen, Upgrades und Softwareentwicklung investiert werden (vgl. [Bart02], S. 59). Kirsch differenziert Softwarekosten weiter in Aufwendungen, die aus der Planung, dem Entwurf, des Erwerbs, der Entwicklung, der Inbetriebnahme, des Betriebs und der Wartung von Software resultieren (vgl. [Kirs99], S. 136).

Die Kostenstellenrechnung dient der Verrechung der in der Kostenträgerrechnung erfassten Kosten auf die verursachenden Kostenstellen. Diese sind selbständige Verantwortungsbereiche im Unternehmen, welche zum Zwecke der Kostenkontrolle definiert sind (vgl. [Eise02], S. 675). Je gröber die Einteilung der Kostenstellen durchgeführt wird, desto einfacher ist der Prozess der Kostenrechnung, wobei die Aussagekraft der Ergebnisse in der Regel entsprechend geringer ausfällt.

Um schließlich zu ermitteln, wofür eine Kostenstelle die anfallenden Kosten beansprucht hat, bedient man sich der Kostenträgerrechnung. Kostenträger können Produkte sein, bei deren Erstellung Kosten anfallen, aber auch innerbetriebliche Leistungen (vgl. [Frie04], S. 183). Als Kostenträger im Kontext der IT-Kostenrechnung bieten sich z.B. CPU-Zeit, genutzter Festplattenplatz, ausgedruckte Seiten, Online-Zugriffe oder Arbeitsstunden an (vgl. [Horv03], S. 740).

Ein Nachteil der traditionellen Kostenrechnungssysteme ist in der Vernachlässigung der Unternehmensbereiche zu sehen, die nicht direkt an der Leistungs- bzw. Produkterstellung beteiligt sind. Dazu zählen bspw. Organisationen, deren Kernkompetenz nicht in der Erstellung von IT-Services zu sehen ist. Die dort entstehenden Kosten werden oftmals als fixe Gemeinkosten behandelt, was einer verursachungsgemäßen Verrechnung nicht förderlich ist (s. [Eise02], S. 788). Hier setzt die *Prozesskos-*

tenrechnung[1] an, welche die traditionellen Kostenrechnungssysteme um ein weiteres Bezugsobjekt ergänzt, den *Prozess* (s. [Mens98], S. 34). Der Einfluss einer wiederholten Ausführung von Prozessen auf die anfallenden Ressourcenkosten soll hier besser berücksichtigt werden (vgl. [Maen95], S. 15 ff.). Zum diesem Zwecke werden *Kostentreiber* identifiziert (vgl. [Brau96], S. 53). Ein Kostentreiber ist eine Maßgröße, welche die repetitiven Aktivitäten in der Form von Prozessergebnissen quantifiziert (vgl. [KuLo95], S. 90), d.h. er ermöglicht es, das Aktivitätsniveau eines Prozesses zu messen (vgl. [Mens98], S. 35). IT-Kostentreiber sind z.B. die Nutzung einer IT-Dienstleistung, verbrauchte CPU-Zeit oder Festplattenkapazität. Diese können als Basis für eine Abrechnung der IT-Kosten verwendet werden. Insgesamt verspricht der Ansatz der Prozesskostenrechnung für die Kostenrechnung in IT-Organisationen eine genauere Zuordnung der verursachten Kosten und somit die Bereitstellung höherwertiger Informationen für das IT-Controlling sowie die strategische Planung.

Ein Ansatz zur möglichst ganzheitlichen Berücksichtigung der Kosten, die Eignung und Betrieb von IT im Unternehmen verursachen, ist die in den 80er Jahren von der Gartner Group vorgeschlagene *Total Cost of Ownership* (TCO, s. z.B. [Krcm00], S. 181 ff., [GRB04], S. 499 ff.). TCO beschränkt sich bei der Definition von Kostenarten nicht nur auf die initialen Beschaffungskosten, sondern erweitert den Fokus auch auf weniger offensichtliche Lebenszykluskosten (vgl. [Krcm00], S. 181 ff.). Diese werden als *nicht-budgetierte Kosten* bezeichnet, da sie im Gegensatz zu den *budgetierten Kosten* nicht im „Buch" auftauchen. Einige grundlegende budgetierte Kosten beziehen sich auf die Beschaffung von Hardware und Software, Softwareentwicklung, Kommunikation, Abschreibung, Support, Administration, Betrieb, Installation, Optimierung und Wartung von IT. Die schwer zu quantifizierenden, nicht-budgetierten Kosten können u.a. durch negative Produktivitätseffekte (Verzögerungen, ergonomische Mängel), Systemausfälle oder nicht ausreichend qualifizierte Nutzer entstehen. TCO zielt darauf, eine größere Transparenz bzgl. der Entstehung von IT-Kosten zu erreichen. Gleichzeitig kann TCO als Planungswerkzeug im Rahmen der Bewertung von projektierten Investitionen im IT-Bereich eingesetzt werden.

Zur IT-Kostenrechnung bietet sich zusammenfassend neben den traditionellen Kostenrechnungssystemen vor allem die Prozesskostenrechnung an. TCO kann ergänzend genutzt werden, um weitere weniger offensichtliche Kostenarten zu berücksichtigen. Insgesamt hängt die Wahl eines Kostenrechnungssystems, wie oben schon erwähnt, von der Einbindung der IT-Organisation in das Unternehmen sowie der damit verbunden der Zielsetzung der Kostenrechnung ab. Daher soll an dieser Stelle keine Empfehlung für ein konkretes Vorgehen abgegeben werden.

2.6 Nutzen von IT

Verglichen mit der Ermittlung der Kosten ist die Identifikation des potenziellen *Nutzens* von IT eine weitaus größere Herausforderung (s. z.B. [HeKu00], S. 307). In diesem Zusammenhang weist Brynjolfsson daraufhin, dass die Produktivität bzw. der Nutzen von IT u.a. deswegen oftmals nicht wahrgenommen wird, weil keine adäquaten Messmethoden existieren, mit denen evtl. Wertschöpfungen durch IT nachgewiesen werden können ([Bryn93]). Dies liegt wiederum darin begründet, dass sich IT-Nutzen zu großen Teilen nicht vollständig quantifizieren, d.h. in monetären Größen angeben, lässt und somit eine Messung schwer durchzuführen ist.

In einschlägiger Literatur wird der Versuch unternommen, verschiedene Nutzenkategorien zu definieren, um auf diese Art eine Grundlage für die Ermittlung von Nutzeneffekten zu schaffen. So differenziert bspw. Horvath einerseits zwischen *direktem* und *indirektem* Nutzen (abhängig vom Entstehungsort des Nutzens) und andererseits zwischen *direkt quantifizierbarem*, *schwer quantifizierbarem* und *nicht quantifizierbarem* Nutzen (abhängig von der monetären Bewertbarkeit, vgl. [Horv03], S. 736). Die Kategorie des schwer quantifizierbaren Nutzens wird von Nijland als *sekundärer* oder

[1] Im amerikanischen Raum existiert ein vergleichbarer Ansatz, das *Activity Based Costing* (s. z.B. [KaCo99]).

auch *intangibler* Nutzen bezeichnet (s. [Nijl04]). Dieser wird durch *primären* Nutzen (*tangibler* Nutzen) erzeugt und kann nicht zuverlässig quantifiziert bzw. vorausgesagt werden. Als Beispiel für einen tangiblen Nutzen wird die Erhöhung der Qualität eines Produkts genannt, woraus der intangible Nutzen der Erhöhung der Zahl der Kunden resultiert, welche mittelbar durch das hochwertige Produkt geworben werden können. Allerdings ist hier kritisch anzumerken, dass es in vielen Fällen schwierig erscheint, die Qualität eines Produkts objektiv zu messen. Daher wird in dieser Arbeit ein tangibler Nutzen analog zu Remenyi ([Reme00]) als Effekt interpretiert, der in Form von konkreten (*greifbaren*) Zahlen beschrieben werden kann (z.B. Kostenreduktion, Gewinnsteigerungen oder ein positiver ROI). Als Beispiel für eine Ausprägung intangiblen Nutzens wird die Arbeitserleichterung für Mitarbeiter genannt.

Ein spezieller Nutzen von IT kann aus den für deren Einführung notwendigen organisatorischen Umstrukturierungen abgeleitet werden (vgl. [AsDo03]). Da oftmals die Geschäftsprozesse eines Unternehmens an ein neues Informationssystem angepasst werden, entstehen allein durch die neuen Abläufe u.U. schon Nutzenpotenziale, die unabhängig von der eigentlichen Funktionalität des Informationssystems sind.

Niemann zählt hinsichtlich des Nutzens von IT hauptsächlich qualitative Kategorien auf (s. [Niem06], S. 144 ff.). So nennt er als Nutzeneffekte bspw. den von Nutzern und Managern subjektiv empfundenen Nutzen, die Auswirkungen auf Unternehmensziele sowie den Unterstützungsgrad der Geschäftsprozesse.

Weiterführend kann IT als so genannter *Enabler* fungieren (s. z.B. [LHP01]). In dieser Form erzeugt sie selbst keine Nutzeneffekte, eröffnet aber aufgrund ihrer Einsatzmöglichkeiten neue Handlungsoptionen (*Enabler-Effect*). So sind Informationssysteme bspw. als Enabler des E-Commerce zu sehen, da dieser ohne die entsprechenden Technologien nicht stattfinden könnte (vgl. [FrLa06]).

Noch schwieriger als die Kategorisierung des Nutzens gestaltet sich dessen Bewertung. Horvath nennt in diesem Kontext einige grundlegende Probleme, die bei der Ermittlung und Bewertung der Wirtschaftlichkeit von Informationssystemen, d.h. der Gegenüberstellung der zu erwartenden Kosten und des Nutzens im Falle der Inbetriebnahme, auftreten (s. [Horv03], S. 736):

- *Ungewissheitsproblem*: Kosten und Nutzen geplanter IT nur schwer schätzbar
- *Quantifizierungsproblem*: Kosten und Nutzen geplanter IT nur schwer quantifizierbar
- *Komplexitätsproblem*: Kosten-/Nutzeninterdependenzen können aufgrund der Komplexität des Systems nicht komplett erfasst werden
- *Systemabgrenzungsproblem*: je nach Abgrenzung gegen das Umsystem sind unterschiedliche Ergebnisse zu erwarten
- *Zeitrestriktionsproblem*: die Wirtschaftlichkeitsfrage ist Teil des Entscheidungsprozesses der Systemgestaltung und muss daher in gegebenen Zeitrestriktionen beantwortet werden

Einer einfachen Erfassung des potenziellen Nutzens von IT wirken also primär die hohe Komplexität des Untersuchungsgegenstandes, die mitunter unscharfe Abgrenzung gegenüber der Umwelt sowie die fehlende Greifbarkeit der zu erwartenden Nutzeneffekte entgegen (vgl. [Pott98], S. 58). Als mögliche Verfahren zur Messung der Kosten bzw. des Nutzens von IT schlägt Horvath die folgenden vor ([Horv03], S. 737):

- Kosten-Nutzen-Analyse
- Time-Saving Time-Salary-Verfahren
- Hedonistische Verfahren
- Nutzenwirkungsnetz
- FAOR-Kosten-Nutzen-Analyse
- Wirkungskettenanalysen

- Transaktionskostenanalyse

Für eine kurze Beschreibung der Verfahren sei an dieser Stelle auf die o.g. Quelle verwiesen.

Es bleibt festzustellen, dass die mögliche Kategorisierung sowie ein Vorgehen bei der Identifizierung und Ermittlung von IT-Nutzeneffekten in der Literatur teilweise recht unterschiedlich dargestellt werden. Ein Konsens existiert allerdings bzgl. der Feststellung, dass Nutzeneffekte oftmals in nicht quantifizierbaren Ausprägungen auftreten. Der Begriff des *intangiblen Nutzens* erscheint hier als Abgrenzung gegen den *tangiblen*, quantifizierbaren Nutzen sinnvoll. Weiterhin besteht eine weitgehende Einigkeit darin, dass ein Hauptnutzen von IT die Unterstützung bzw. Realisierung von Geschäftsprozessen ist (vgl. MuOe99], S. 455). Dieser Effekt kann bspw. über Services gekapselt und somit in Teilnutzen aufgebrochen und messbar gemacht werden. Hier besteht der Nutzen in den verschiedenen Ausprägungen von Serviceeffekten sowie dem Erzielen von Einnahmen durch die Verrechnung von Services.

2.7 Zusammenfassung

Bei der Betrachtung der obigen Ausführungen zu den Aufgabenbereichen des IT-Managements lässt sich feststellen, dass in der Literatur keine durchgehend einheitliche Terminologie in diesem Themenbereich verwendet wird. Weiterhin unterscheiden sich die Ziele und Aufgaben, welche dem IT-Management zugeordnet werden, teilweise recht deutlich. Die folgende Zusammenfassung versucht daher eine umfassende und generalisierende Perspektive auf das IT-Management zu präsentieren, um so einen Überblick über die zentralen Ziele und Aufgaben zu bieten. Diese werden innerhalb der drei Hauptkategorien *Gestaltung und Ausrichtung der IT-Organisation und des IT-Managements*, *Durchführung des IT-Managements durch die IT-Organisation* und *Kostenermittlung und Verrechnung* erläutert.

Gestaltung und Ausrichtung der IT-Organisation und des IT-Managements

- Einordnung der Organisation ins Unternehmen
- Spezifizieren von Entscheidungsbefugnissen und Verantwortlichkeiten
- Gestaltung der Organisation und der Managementprozesse
- Berücksichtigung gesetzlicher (und anderer) Rahmenbedingung
- Definition einer IT-Strategie
- Entscheidung für einen Kostenverrechnungsansatz

Die Aufgaben, welche die Gestaltung der IT-Organisation selbst zum Inhalt haben, sind an den Unternehmensstrategien auszurichten. Dazu gehört zunächst die Entscheidung, in welcher Form die IT-Organisation im Unternehmen eingeordnet werden soll (z.B. Outsourcing vs. Abteilung). Weiterhin sind die Verantwortlichkeiten für die Aufgabenbereiche zu definieren und in diesem Zusammenhang eine Aufbau- und Ablauforganisation zu etablieren. Grundlegend ist hierbei sicherzustellen, dass das IT-Management letztendlich in der Verantwortung der Unternehmensleitung verbleibt. Nur auf diese Weise kann nachhaltig sichergestellt werden, dass IT ihre zentrale Rolle und strategischen Potenziale im Unternehmen auch wahrnehmen und umsetzen kann und dies nicht an Zuständigkeitsdifferenzen oder zu geringer Priorisierung scheitert.

Auf die Gestaltung der Organisation wirken zusätzlich gesetzliche, kulturelle u.a. Rahmenbedingungen ein, welche im Gestaltungsprozess mit berücksichtigt werden müssen. Die Entscheidung für eine IT-Strategie ist auf dieser Ebene weniger als Entscheidung für den Einsatz konkreter Technologien zu sehen. Hier sollte lediglich festgelegt werden, auf welche Weise IT die langfristigen Ziele eines Unternehmens unterstützen kann (z.B. Einführung von E-Commerce-Lösungen). Die Entschei-

dung für die Art der Kostenverrechnung hängt unmittelbar mit der Einordnung der IT-Organisation in das Unternehmen zusammen. Hier ist in Abhängigkeit der Konzeption als Cost, Recovery oder Profit Center aus verschiedenen Ansätzen zu wählen.

Durchführung des IT-Managements durch die IT-Organisation

- Planung und Entwurf
- Implementierung
- Betrieb
- Evaluation und Kontrolle

Die von der IT-Organisation durchgeführten Aufgaben des IT-Managements sollen im Ergebnis ein effektives und effizientes Funktionieren der IT gewährleisten. Dies bezieht sich zunächst auf das Management der IT-Landschaft. Hier sind verschiedene *idealtypische* Managementphasen entlang eines Entwicklungs- bzw. Lebenszyklus einer IT-Landschaft festzustellen (s. Abbildung 1). Die zyklische Anordnung der Phasen in der Abbildung deutet an, dass die innerhalb einer Phase erstellten Artefakte (z.B. Modelle) und Dokumente an die nachgelagerten Phasen weitergereicht werden. Dabei unterscheiden sich die Aktivitäten einer Phase in Abhängigkeit von der Ausgangssituation eines Projekts hinsichtlich ihrer Inhalte, auszuführenden Tätigkeiten und zugrunde liegenden Modelle. So kann das Ziel einerseits die Planung und Umsetzung einer IT-Landschaft zur Realisierung von Services ohne Berücksichtigung von bereits existierenden Infrastrukturen bzw. Services sein. Andererseits ist an die Durchführung von Änderungen an bestehenden Installationen zu denken. In beiden Fällen werden alle Phasen durchlaufen.

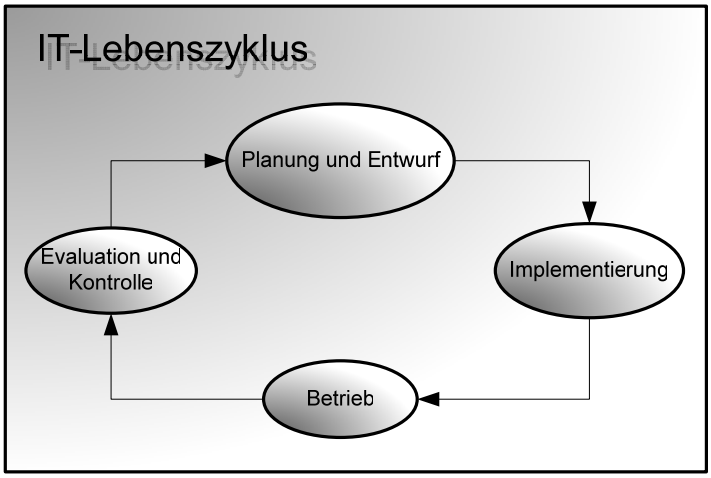

Abbildung 1: Idealtypischer Lebenszyklus von IT-Landschaften

Planung und Entwurf

Zunächst werden IT-Projekte aufgrund von Anforderungen seitens eines Auftraggebers oder Eingaben aus der Evaluationsphase initiiert. Der Leistungserstellungsprozess für Services wird anhand der Auftraggeberanforderungen, welche in SLAs festgehalten werden sollten, geplant sowie die einzelnen Services definiert. Entwurfsentscheidungen für konkrete IT-Architekturen werden ebenfalls auf der Basis der Anforderungen sowie evtl. vorhandener Randbedingungen getroffen.

Implementierung

Die Implementierungsphase hat die Umsetzung der Planungs- und Entwurfsvorgaben zum Inhalt und deckt Tätigkeiten im Bereich des Aufbaus von Infrastrukturen, der Beschaffung von IT-Komponenten,

der Softwareentwicklung und der Inbetriebnahme von IT ab. Die Produktionsbereitschaft für Services wird durch diese Maßnahmen sichergestellt.

Betrieb

Unter Betrieb fallen die Aufgaben im Bereich der Administration, Wartung, Reparatur, Support usw. von IT-Landschaften. Der reibungslose Ablauf des Tagesgeschäfts ist zu gewährleisten, wobei eine angemessene Benutzerbetreuung ebenfalls gegeben sein muss.

Evaluierung und Kontrolle

Eine fortwährende Evaluierung und Kontrolle der Effektivität (Leistungsfähigkeit) und Effizienz (Wirtschaftlichkeit) der IT-Landschaft ist Gegenstand dieser Phase. Dies dient sowohl der Förderungen einer hohen Qualität der Leistungen, als auch der Kostenkontrolle. Gleichzeitig sollte der Nutzen der IT bzw. der Services in die Betrachtung mit einbezogen werden. Dies kann bspw. im Rahmen des IT-Controllings durch die Ermittlung von Kennzahlen oder die Durchführung vergleichender Benchmarks erfolgen. Monitoring-Verfahren können für die Überwachung der Lastverteilung innerhalb der IT-Infrastruktur verwendet werden. Nicht zuletzt bieten sich aber auch Analysen auf der Basis konzeptueller Modelle an, sofern solche vorhanden sind. Weiterhin ist eine permanente Beobachtung des Marktes hinsichtlich alternativer Technologien notwendig. Auf der Basis der Ergebnisse der Evaluationsphase kann im Bedarfsfall beginnend mit einer weiteren Planungsphase die IT-Landschaft im Rahmen eines geregelten Change Managements verändert oder erneuert werden.

Kostenermittlung und Verrechnung

- Verursachungsgerechte Ermittlung von Kosten
- Verrechnung von Services/Generierung von Nutzen

Die Durchführung der Tätigkeiten der o.g. Phasen des IT-Lebenszyklus verursachen der Organisation Kosten. Diese sind zumindest möglichst verursachungsgerecht zu ermitteln sowie ggf. an die Nutzer der Services, welche durch die IT-Organisation zur Verfügung gestellt werden, weiterzugeben. In den Prozess der Kostenermittlung fließen maßgeblich die in der Evaluations- und Kontrollphase gewonnenen Informationen ein. Darauf aufbauend ist in Abhängigkeit von der Einordnung der IT-Organisation im Unternehmen die Verrechnung der Services zu leisten. Dadurch wird ein tangibler Nutzen generiert und in der Konsequenz eine wichtige Grundlage für die Wirtschaftlichkeit der IT-Organisation insgesamt geschaffen. Im Falle eines unternehmensinternen Cost bzw. Recovery Centers für die IT-Organisation entsteht allerdings kein direkt messbarer Nutzen. Hier ist der Nutzen der unterstützten Geschäftsprozesse mit den Kosten der Serviceproduktion in Relation zu setzen, um eine Wirtschaftlichkeitsbetrachtung der IT durchführen zu können. Im Falle eines Profit Centers entsteht ein direkt messbarer Nutzen durch die Abrechnung der gelieferten Services. Somit sind hier zur Bewertung der Wirtschaftlichkeit primär die Kosten der Produktion mit den Einnahmen durch die Services zu verrechnen.

Nach der Vorstellung der zentralen Aufgaben des IT-Managements ordnen wir diese in einen *Handlungsrahmen für das IT-Management* ein. Der Rahmen soll die Zusammenhänge zwischen den einzelnen Aufgabenfeldern noch einmal verdeutlichen. Es wird zunächst zwischen der Sicht auf die IT-Organisation in der Funktion eines Dienstleisters und der Sicht des Kunden, welcher die Dienstleistungen konsumiert, unterschieden. Diese Differenzierung basiert auf der grundlegenden Trennung der Leistungserstellung und der Nutzung dieser Leistung. Ob ein Kunde innerhalb des Unternehmens – z.B. in Form einer anderen Abteilung – oder außerhalb des Unternehmens angesiedelt ist, ist für die Betrachtung zunächst nicht relevant. Abbildung 2 stellt den Handlungsrahmen schematisch dar.

Von außen wirken auf beide Partner *Umwelteinflüsse*, wie Marktgegebenheiten, Konkurrenten, Gesetze etc. ein und beeinflussen mittelbar den gesamten Bereich des IT-Managements. Auf Seiten des Kunden stehen die *Geschäftsprozesse* im Mittelpunkt, welche die *Unternehmensstrategien*[1] realisieren und für das Unternehmen einen *Nutzen* generieren. Weiterhin verfügt der Kunde in der Regel über eine IT-Landschaft, die der Unterstützung der Geschäftsprozesse dient. Die Sicht des Kunden auf den Dienstleister beschränkt sich zunächst auf die zentrale Schnittstelle zwischen den beiden Organisationen, welche durch die vom Dienstleister zur Unterstützung der Kundenprozesse angebotenen *Services* gebildet wird.

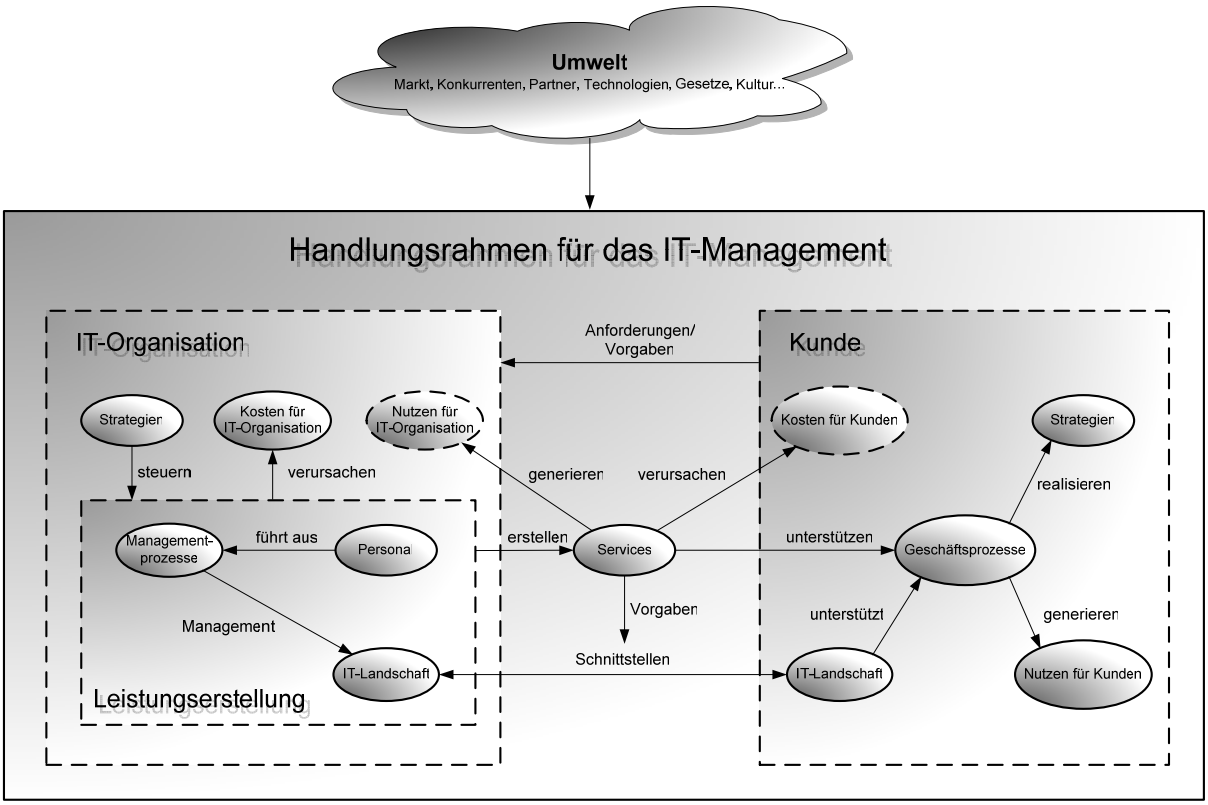

Abbildung 2: Handlungsrahmen für das IT-Management

Auf der Seite des Dienstleisters existieren ggf. ebenfalls Strategien, welche einen übergeordneten, steuernden Einfluss auf die *Managementprozesse*, das die Prozesse ausführende *Personal* sowie die Gestaltung der technischen Basis – der *IT-Landschaft* - ausüben. Die Formulierung dieser Strategien sowie die Gestaltung der IT-Organisation fallen in den Aufgabenbereich des IT-Managements. Für die Durchführung der Managementtätigkeiten ist das Personal[2] verantwortlich. Die Prozessausführung ist auf das Management der IT entlang des Lebenszyklus einer IT-Landschaft ausgerichtet. Die Summe der Aktivitäten innerhalb der Phasen des Lebenszyklus bildet die Grundlage für die *Leistungserstellung*, d.h. die Produktion von Services, welche dem Kunden angeboten werden. Die Leistungserstellung verursacht der IT-Organisation *Kosten*, deren Ermittlung eine weitere wichtige Aufgabe des IT-Managements darstellt. Diese Kosten bilden die Basis zur Bewertung der Wirtschaftlichkeit der IT-Organisation, indem sie mit dem generierten *Nutzen* in Beziehung gesetzt werden. Ob und wie ein Nutzen für die IT-Organisation generiert wird, hängt von der Organisationsform ab (s.o.). Der tangible Nutzen (Einnahmen durch Services) verursacht auf Kundenseite Kosten, welche

[1] Hierbei ist zu beachten, dass Unternehmen sehr oft über keine expliziten Strategieformulierungen verfügen bzw. solche bewusst verfolgen. Dennoch ist anzunehmen, dass zumindest implizit langfristige Planungsvorgaben existieren, auch wenn diese lediglich – wie gerade in kleineren Unternehmen zu beobachten – in den Köpfen der Unternehmensleitung existieren.

[2] Der Begriff Personal bezieht sich hier nicht zwingend auf Mitarbeiter der IT-Organisation, sondern bezeichnet den Personenkreis, der für die Durchführung der Prozesse verantwortlich ist. Hierfür kommen ebenso externe Dienstleister in Frage.

dort wiederum auf den durch die unterstützten Geschäftsprozesse generierten Nutzen bezogen werden können. Da Kosten und Nutzen eines Services in der Regel nur dann betrachtet werden, wenn eine Abrechnung seitens des Dienstleisters durchgeführt wird, sind diese Aspekte in der Abbildung mit einer gestrichelten Linie eingefasst.

Die Spezifikation der Services liefert Vorgaben für die *Schnittstellen*, die zwischen der IT-Landschaft des Anbieters und der des Kunden etabliert werden müssen. Dies bezieht sich auf Services, in deren Leistungserstellung die IT des Anbieters einfließt, wobei gleichzeitig in deren Leistungsergebnis die IT des Kunden involviert ist. Die Schnittstellen zwischen den beiden IT-Landschaften ermöglichen die Aufnahme der Leistungen durch den Kunden. Es muss hierbei keine zwingende physische Trennung zwischen den beiden IT-Landschaften existieren. Die Trennung zwischen Produktionssystem und Kundensystem findet zunächst nur auf logischer Ebene statt. Im konkreten Fall kann eine IT-Komponente durchaus sowohl Teil des einen als auch des anderen Systems sein.

Die Kundenseite formuliert je nach Art der Beziehung zur IT-Organisation unterschiedliche *Anforderungen* bzw. *Vorgaben* an deren Adresse. Dies können Vorgaben für IT-Strategien abgeleitet aus den Unternehmensstrategien sein, ein begrenztes Budget zur Finanzierung des Betriebs oder konkrete Anforderungen an einzelne Services.

Der hier skizzierten Handlungsrahmen für das IT-Management nutzt die Metaphern *Kunde* und *Dienstleister* zur Beschreibung der grundlegenden Perspektiven des IT-Nutzers und des Service-Erstellers. Diese Sichtweise ermöglicht die Spezifikation wohldefinierter Schnittstellen zwischen den Partnern und wird weiterführend von dem Konzept des Services als auszutauschende Leistung unterstützt. Der Handlungsrahmen erhebt nicht den Anspruch, umfassende und allgemeingültige terminologische und inhaltliche Vorgaben für den Themenbereich zu präsentieren. Er beantwortet ebenfalls nicht die Frage, *wie* IT-Management letztlich funktioniert. Es wird lediglich schematisch aufgezeigt, *was* in diesem Rahmen getan werden muss. Im weiteren Verlauf der Arbeit werden die hier gewonnen Erkenntnisse über das Wesen des IT-Managements als Grundlage für die Formulierung von Anforderungen an Ansätze und Methoden für das IT-Management genutzt. Diese können sowohl der Einordnung und Bewertung des State of the Art dienen als auch als Leitfaden für den Entwurf neuer Methoden genutzt werden.

Es bleibt festzuhalten, dass eine möglichst effektive und effiziente Unterstützung der Kundenprozesse durch IT sowohl dem Kunden als auch dem Dienstleister zum Vorteil gereicht. Dies trägt maßgeblich zur Wirtschaftlichkeit der beteiligten Organisationen und damit zum Geschäftserfolg bei. Zur Erreichung dieses Ziels erscheint ein methodisches Vorgehen, wie es im Folgenden hergeleitet wird, hilfreich und notwendig.

3 Anforderungen an eine Modellierungssprache für das IT-Management

Aufbauend auf den Erkenntnissen des vorangegangen Kapitels, in dem die grundlegende Terminologie des IT-Managements eingeführt sowie die damit verbundnen Kernkonzepte identifiziert wurden, sowie vor dem Hintergrund der intendierten Sprachanwendungen werden nun Anforderungen an eine IT-Modellierungssprache formuliert. Zunächst soll eine Sprache im Rahmen einer Methode für das IT-Management auf eine Bereitstellung von anwendungsnahen Konzepten zur angemessenen Beschreibung der zentralen betriebswirtschaftlich und informationstechnisch orientierten Perspektiven auf ein Unternehmen zielen. Dies ermöglicht den Managementverantwortlichen grundlegende Einblicke in die Eigenschaften der Komponenten einer IT-Landschaft sowie deren Zusammenhänge mit anderen Komponenten und organisatorischen Konzepten. Weiterhin soll die Durchführung technischer Analysen (hinsichtlich der Effektivität von IT, der Ermittlung von Leistungsengpässen und Medienbrüchen u.a.) sowie betriebswirtschaftlicher Analysen (Effizienz der IT, Betrachtung von IT-Kosten und IT-Nutzen) unterstützt werden. Schließlich ist darauf zu achten, dass die Wiederverwendbarkeit der Sprachkonzepte vor dem Hintergrund einer zufrieden stellenden Modellierungseffizienz auf hohem Niveau gewährleistet ist. Die hier genannten primären Zielsetzungen spiegeln sich in den Anforderungen, welche in den folgenden Abschnitten entwickelt werden, wider.

3.1 Sprachspezifikation

Unabhängig von der Art der Sprachspezifikation hat diese einigen grundlegenden Anforderungen zu genügen. Hier sind zunächst Korrektheit sowie Einheitlichkeit und Redundanzfreiheit zu nennen (vgl. [FrLa03], S. 26). Korrektheit zielt auf die Möglichkeit, anhand der Sprachspezifikation unzulässige Modelle zu identifizieren sowie die Menge aller - hinsichtlich Syntax und Semantik - gültigen Modelle zu generieren. Weiterhin sollte eine Sprache keine redundanten Konzepte enthalten (Redundanzfreiheit) sowie gleiche Sachverhalte nicht auf unterschiedliche Weise beschreiben (Einheitlichkeit). Als Formen der Sprachspezifikation bieten sich hinsichtlich der Erfüllung dieser Anforderungen formale oder metamodellbasierte Ansätze an. Der Untersuchungsgegenstand der Arbeit in Form von IT-Landschaften mit formal kaum zu beschreibenden Konzepten auf hohem semantischem Niveau lässt allerdings vermuten, dass ein metamodellbasierter Ansatz geeignet scheint. Informale Sprachbeschreibungen sind vor allem hinsichtlich der automatisierten Verifikation von Modellen nicht anzuraten.

3.2 Wiederverwendung

Die Wiederverwendung von Modellen bzw. Teilmodellen ist nicht nur aus ökonomischer Sicht ein erstrebenswertes Ziel der Modellierung. Wiederverwendung fördert neben der Modellierungseffizienz auch die Vermeidung von Fehlern durch die Verwendung bereits bewährter Konstrukte. Wiederverwendbarkeit wird prinzipiell durch Abstraktion gefördert (vgl. [FrLa03], S. 27). Konkret bedeutet dies, dass eine Modellierungssprache Abstraktionskonzepte vorhalten sollte, die es erlauben, von im Kontext nicht relevanten Sachverhalten zu abstrahieren. Somit wird die Komplexität der Modelle reduziert und ihre allgemeine Anwendbarkeit erhöht. Eine geringere Modellkomplexität hat zusätzlich positive Auswirkungen auf die Wartbarkeit des Modells.

Als Konzepte zur Förderung der Wiederverwendung im Kontext dieser Arbeit bieten sich die beiden nachfolgenden an:

- *Typbildung/Kapselung*
  Typbildung ermöglicht die Abstraktion von Instanzen sowie die Kapselung ihrer gemeinsamen Eigenschaften in einem Typ. Der Typ wird im Modell stellvertretend für die Menge sei-

ner Instanzen verwendet werden, statt diese jeweils einzeln aufzuführen. Modellwartung erfolgt nur in Form von Änderungen an Typen, nicht an den Instanzen.

- *Generalisierung/Spezialisierung/Vererbung*
  Durch die Spezialisierung eines Typs in einen oder mehrere Subtypen werden die vorhandenen Typdefinitionen wiederverwendet. Die Wartbarkeit eines Modells wird dadurch verbessert, dass Änderungen zentral an einem Supertyp durchgeführt werden können, welcher diese dann durch einen Vererbungsmechanismus an seine Subtypen weitergibt.

Ungeachtet der oben beschriebenen Wiederverwendungskonzepte muss das semantische Niveau der Sprachkonzepte so gewählt sein, dass durch sie eine anwendungsnahe Beschreibung einer Domäne möglich ist. Andererseits dürfen die Typen nicht zu sehr auf die Eigenarten spezieller Instanzen eingehen, um die Reichweite der Wiederverwendung nicht zu reduzieren. Hier gilt es einen Kompromiss zwischen diesen beiden Anforderungen zu finden (vgl. [Fran94], S. 64 ff.).

Zusammenfassend ist von einer Modellierungssprache zu fordern, dass sie Typbildung/Kapselung und Generalisierung/Spezialisierung/Vererbung unterstützt. Weiterhin müssen die Konzepte der Sprache ein für die Wiederverwendung förderliches Abstraktionsniveau aufweisen.

3.3 Anwendungsnahe Konzepte

Gegenstand der konzeptuellen Modellierung ist die Darstellung von realweltlichen Objekten und Sachverhalten durch möglichst anwendungsnahe, im Zeitverlauf stabile Sprachkonzepte innerhalb eines Modells. Eine Modellierungssprache sollte eine Menge von Sprachkonzepten enthalten, welche dieser Anforderung genügen (vgl. [FrLa03], S. 32). Die Konzepte sollten hinsichtlich ihrer Semantik ausreichen, um die Domäne angemessen abzubilden. Gleichzeitig ist die Sprache weitgehend einfach zu halten, d.h. die Anzahl der Konzepte soll möglichst gering sein, um eine einfache Anwendung zu gewährleisten (vgl. [FrLa03], S. 28). Auch sollten die Sprachkonzepte dem Modellierer leicht verständlich sein, d.h. im Idealfall mit seiner konzeptuellen Sicht auf die Domäne korrespondieren (vgl. [FrLa03], S. 29). Mit zu weit abstrahierten Konzepten kann dieses Ziel nicht erreicht werden. In diesem Zusammenhang ist allerdings darauf zu achten, dass die im letzten Abschnitt diskutierte Wiederverwendungsreichweite der Konzepte nicht zu stark eingeschränkt wird. Schließlich ist noch die Forderung nach der Anschaulichkeit eines mit der Sprache erstellten Modells zu nennen. Diese hängt nicht nur von den vorhandenen Konzepten ab, sondern auch von der verwendeten Notation. Die Symbole für die Modellelemente müssen die Konzepte eindeutig repräsentieren und eine einfache Identifikation der Semantik fördern.

Aufbauend auf der Herleitung der grundlegenden IT-Managementaufgaben in Abschnitt 2 sowie den oben formulierten allgemeinen Anforderungen werden diejenigen Konzepte, die eine Modellierungssprache in unserem Kontext vorhalten sollte, zusammengefasst. Dabei wird an dieser Stelle nicht unterschieden, welche Konzepte durch die Sprache selbst und welche über Schnittstellen zu weiteren Modellierungssprachen, welche diese Konzepte enthalten, realisiert werden. Im Vordergrund steht lediglich die Forderung der Verfügbarkeit. Als notwendig zu erachten sind die folgenden Konzeptkategorien:

- Technische Konzepte
  - Hardware
  - Software
  - Standard
- Betriebswirtschaftliche Konzepte
  - Organisation und Service
  - Kosten und Nutzen
- Übergeordnete Konzepte

- Informationssystem

Technische Konzepte -> Hardware

Hardwarespezifische Konzepte basieren prinzipiell auf einem *Hardwaregerät*, welches unterschiedliche Funktionen innehaben kann. Hier sind vor allem *Computer*, *Drucker*, *Scanner*, *Telefon*, *Fax* sowie *Netzwerkkomponenten* zu nennen. Computer sind Geräte, die auf Basis von Eingaben Berechnungen durchführen und ein Ergebnis ausgeben. Sie verfügen über dedizierte Rechenkapazitäten. Drucker transformieren elektronisch vorliegende Daten auf ein physisches Medium, wie z.B. Papier oder Folie. Scanner bieten die Möglichkeit eine solche Transformation umzukehren. Sie verwandeln gedruckte Informationen in elektronisch speicherbare Daten[1]. Ein Telefon steht stellvertretend für alle Kommunikationsgeräte, die es erlauben Ton und ggf. Bild auf elektronischem Wege mit einem Kommunikationspartner auszutauschen. Dies erfolgt in der Regel mit minimaler Zeitverzögerung. Ein Fax beschränkt sich auf den Austausch von Papiervorlagen, welche nach dem Einscannen und Senden an der empfangenden Gegenstelle wieder auf Papier ausgegeben werden. Sofern die zu faxenden Daten schon in elektronischer Form vorliegen, wird kein Papier zum Senden benötigt. Auch auf der Empfangsseite kann auf Papier verzichtet werden, wenn das Fax gespeichert statt ausgedruckt wird. Eine Netzwerkkomponente ist ein Gerät, dessen Aufgabe die Realisierung eines Teils einer Netzwerkinfrastruktur ist. Es implementiert Anschlüsse, Knotenpunkte, Verteiler, Signalverstärkung, Filterfunktionen etc. Die hier genannten Gerätefunktionen haben alle – bis auf das Fax – die Gemeinsamkeit, dass sie nicht durch eine Kombination der anderen Funktionen realisiert werden können. Eine Faxfunktion ist prinzipiell durch die Kombination von Telefon, Scanner und Drucker zu realisieren (ggf. an einem Computer). In der Realität existieren allerdings Faxgeräte, die nicht außerhalb der Realisierung der Faxfunktion als Scanner, Drucker oder Telefon einsetzbar sind. Daher ist das Konzept des Faxes hier mehr als dedizierte Funktion zu sehen, denn als Kombination aus anderen Funktionen. Dies entspricht nicht zuletzt der vorherrschenden Interpretation und Terminologie in der Domäne, die mit einer Sprache für die Modellierung von IT abgebildet werden soll.

Die hier genannten Hardwarefunktionen können jeweils einzeln einem dedizierten Gerät zugeordnet sein oder, wie oben schon angedeutet, von Mehrzweckgeräten erfüllt werden. Ein Computer kann zusätzlich als Telefon fungieren, während ein Office-Drucker Scan-, Fax und Druckfunktionen in sich vereint. Dieser Sachverhalt muss beim Entwurf der Modellierungskonzepte beachtet werden. Weiterhin sind die *Standorte* der Geräte zu berücksichtigen. Die können recht grob spezifizierte Gebiete (Werksgelände, Funkzelle) sein, aber auch Gebäude und Büros. Durch Hardwaregeräte werden lokale oder externe *Netzwerke* gebildet, zu denen die Geräte auf unterschiedliche Art und Weise Zugriff haben. Die Kommunikation zwischen Geräten erfolgt auf der Basis von *Schnittstellen*. Schließlich sind noch *physische Datenmedien* zu berücksichtigen, welche der Speicherung von anfallenden Daten dienen.

Eine Modellierungssprache sollte zusammenfassend mindestens die folgenden hardwarespezifischen Konzepte bereithalten:

- Hardwaregerät
  - Computer
  - Drucker
  - Scanner
  - Telefon
  - Fax

[1] Bei digitalen Kameras, 3D-Scannern, Barcode-Lesern etc. handelt es sich prinzipiell ebenfalls um Scanner in unsrem Sinne. Diese unterscheiden sich lediglich durch die unterschiedliche Ausprägung der zu digitalisierenden Information, so dass an dieser Stelle keine weitere Differenzierung erfolgt.

    - Netzwerkkomponente
- Schnittstelle
- Netzwerk, Netzwerkzugriff
- Physisches Datenmedium
- Standort

Hardwaregeräte als zentrale Abstraktion sollten Beziehungen zu allen anderen Konzepten aufweisen.

Technische Konzepte -> Software

Zentrales Konzept in dieser Kategorie ist *Software*. Software kann weiter in *Softwareprodukte*, lauffähige Software (*Softwareartefakt*) sowie *Softwarelizenzen* differenziert werden. Lauffähige Software muss auf einem Hardwaregerät installiert sein, gehört zu einem Softwareprodukt und ist einer Lizenz zugeordnet. Gewisse Softwaretypen benötigen andere, um lauffähig zu sein. So werden in der Regel von Anwendungssystemen Betriebssysteme benötigt, Softwarekomponenten verlangen eine spezielle Laufzeitumgebung. Softwareartefakte bilden Teile einer *Architektur*, welche üblicherweise in *Schichten* organisiert ist. Weiterhin ist Software zuständig für die Speicherung von *Daten* oder greift darauf zu. Daten werden in einem bestimmten *Format* vorgehalten. Die Kommunikation zwischen Softwareartefakten erfolgt auf der Basis von *Schnittstellen*. Diese nutzen *Kommunikationsprotokolle*.

Es sind von einer Modellierungssprache die nachfolgenden softwarespezifischen Konzepte zu fordern:

- Software, Softwareprodukt, Lizenz
- Architektur, Architekturschicht
- Daten, Format
- Schnittstelle, Kommunikationsprotokoll

Abhängigkeiten zwischen Softwaretypen müssen darstellbar sein.

Technische Konzepte -> Standard

Standards existieren für die unterschiedlichsten Ausprägungen von IT in diversen Formen. Sie beziehen sich sowohl auf software- als auch hardwarespezifische Konzepte. So können von den bisher genannten Abstraktionen mindestens Hardwaregeräte, Software, Softwareprodukte, Formate, Architekturen sowie Kommunikationsprotokolle von Standards beschrieben werden. Von einer Modellierungssprache ist zu fordern, dass diese Konzepte einem Standardkonzept zuzuordnen sind. Weiterhin muss die Beschreibungsstruktur eines Standards Aspekte wie Offenheit, zuständige Organisation u.a. vorsehen.

Betriebswirtschaftliche Konzepte -> Organisation und Services

Die Darstellung organisatorischer Aspekte eines Unternehmens basiert in der Hauptsache auf der Abstraktion des *Services* zur Unterstützung von *Geschäftsprozessen* sowie verschiedener Ausprägungen von *Organisationsrollen*. Geschäftsprozesse sind verknüpft mit *Ereignissen*, die Prozesse initiieren können bzw. von diesen generiert werden. *Strategien* werden von Geschäftsprozessen umgesetzt. Organisationsrollen bilden ab, welche Funktionen ein Rollenträger für das Unternehmen innehat. Potenzielle Rollenträgertypen sind sowohl *Personentypen* (Administrator, CIO) als auch Organisationen oder Teilorganisationen. Letztere können in *Organisationseinheiten*, *strategische Geschäftseinheiten* oder *externe Partner* unterschieden werden. Services werden von Organisationsrollen zur Verfügung gestellt und bilden die Schnittstelle zwischen einer dienstleistenden und einer dienstkonsumierenden Organisation.

Es ergibt sich die Forderung nach den folgenden Konzepten:

- Service
- Geschäftsprozess, Ereignis
- Strategie
- Organisationsrolle
    - Person
    - Organisationseinheit
    - Strategische Geschäftseinheit
    - Externer Partner

Eine Organisationsrolle hat sowohl mit allen hier genannten Konzepten als auch mit den zentralen Soft- und Hardwarekonzepten in Beziehung zu stehen, um Verantwortlichkeiten und Aufgaben abzubilden. Ein Service kann mit der realisierenden Software verbunden sein, sofern es sich im einen IT-basierten Service handelt. Die weiteren Eigenschaften eines Services, die in Abschnitt 2.4 hergeleitet wurden, sind zum großen Teil durch die technischen Sprachkonzepte abzubilden.

Betriebswirtschaftliche Konzepte -> Kosten und Nutzen

Kosten entstehen in Verbindung mit einer Vielzahl von IT-Komponenten. Ihre Ausprägungen können entsprechend differenziert mithilfe von *Kostenträgern* bzw. *Kostentreibern* spezifiziert werden. Mittels *Kostenarten* werden Kosten den verursachenden Objekten zugeordnet. Hier sind *Personal*, *Hardware*, *Software*, *Unterbringung* sowie weitere (*nicht-budgetierte*) Kosten zu nennen. *Kostenstellen* dienen der Beschreibung der Organisationsrollen, denen Kosten angerechnet werden. Nutzen ist weniger einfach zu messen und ist tangibel oder intangibel ausgeprägt.

Zu fordern sind demnach mindestens die nachstehenden kostenspezifischen Konzepte:

- Nutzen
- Kostenträger
- Kostentreiber
- Kostenart
    - Personalkosten
    - Hardwarekosten
    - Softwarekosten
    - Unterbringungskosten
    - Nicht-budgetierte Kosten
- Kostenstelle

Die Kostenarten müssen den zugehörigen Konzepten (Hardwarekosten -> Hardware usw.) zugeordnet werden können. Kostenstellen können durch Organisationsrollen vertreten werden. An dieser Stelle ist anzumerken, dass eine Modellierungssprache kein primäres Werkzeug für die Durchführung eines betrieblichen Rechnungswesens sein kann. Sie dient vielmehr dazu, die an den entstehenden Kosten beteiligten Typen sowie in eingeschränkter Form den möglichen Nutzen von IT aufzuzeigen. Diese Informationen müssen aufgegriffen und einem instanzbasierten Rechungswesen zugeführt werden.

Übergeordnete Konzepte -> Informationssystem

Ein den technischen und betriebswirtschaftlichen Abstraktionen einer IT-Landschaft übergeordnetes Konzept ist das *Informationssystem*. Es aggregiert Software, Hardware und Organisationsrollen zu einer logischen Einheit und ermöglicht damit eine eher strategische Sicht auf IT. Auf diesem höheren Betrachtungsniveau kann auch der *Nutzen* von IT hinterfragt werden. Dieser ergibt sich aus der

Unterstützung von Geschäftsprozessen mittels Services oder der Schaffung einer Grundlage für die Verfolgung neuer Strategien. Weiterhin sind *Umweltfaktoren*, welche von außen auf Informationssysteme einwirken (bspw. regionale Marktspezifika, gesetzliche Rahmenbedingungen) sowie auf sie einwirkende *Risiken* (d.h. potenzielle *Zwischenfälle*) und *Gegen-* bzw. *Schutzmaßnahmen* zu betrachten. Es sind somit folgende Konzepte von einer Modellierungssprache vorzuhalten:

- Informationssystem
- Umweltfaktor
- Risiko
    - Zwischenfall
    - Maßnahme

Das Informationssystem sollte direkt mit den realisierenden Software- und Hardwaretypen in Beziehung stehen. Weiterhin sind die zentralen ablauf- und aufbauorganisatorischen Konzepte anzubinden.

3.4 Operationalisierbarkeit

Mit Operationalisierbarkeit bezeichnet man die Möglichkeit, ein Modell über die reine Visualisierung von Sachverhalten hinaus weiter zu verwenden. Dies beinhaltet oft dessen Transformation in eine andere Sprache, z.B. eine Implementierungssprache. Hier ist es wichtig, dass die Konzepte in der Modellierungssprache mit kompatiblen Konzepten in der Implementierungssprache korrespondieren und eine Abbildung möglichst verlustfrei erfolgen kann. Die Simulation geplanter oder existierender Sachverhalte ist eine weitere verbreitete Anwendung von Modellen. Von größerer Relevanz im Kontext dieser Arbeit ist allerdings die Verwendung von Modellen über IT-Landschaften für die Durchführung von Analysen sowie die Umsetzung der im Modell festgehaltenen und geplanten Sachverhalte in die Realität.

Analysen sind mit unterschiedlichster Zielsetzung denkbar. So stehen bei technischen Analysen primär Aspekte wie Kompatibilität und Effektivität von IT-Komponenten im Vordergrund. Es werden z.B. die Ursachen von Leistungsengpässe ermittelt oder Medienbrüche identifiziert. Aus betriebswirtschaftlicher Sicht definiert vor allem die Effizienz der IT hinsichtlich der verursachten Kosten in Relation zu ihrem Nutzen das Untersuchungsziel. Eine Modellierungssprache sollte Sprachkonzepte vorweisen, welche die Durchführung solcher Analysen ermöglicht. Dies setzt eine grundlegende syntaktische und semantische Kompatibilität der Eigenschaften der Konzepte, welche innerhalb einer Analyse miteinander in Beziehung gesetzt werden sollen, hinsichtlich des Analyseziels voraus.

Beispiele zentraler Fragestellungen für modellbasierte Analysen über IT-Landschaften sind die folgenden:

- Welche Architekturen werden realisiert?
    - z.B. Client/Server, Komponententechnologien
- Wie ist die Flexibilität der IT-Architektur einzuschätzen?
    - Aufwand bei der Implementierung von Änderungen, Reaktionszeiten
- Welche technischen Abhängigkeiten existieren innerhalb der IT-Landschaft?
    - Welche Hardware- bzw. Softwarekomponenten werden von einer Software vorausgesetzt?
    - Wie sehen die Implikationen der Abhängigkeiten für Änderungen an der IT-Landschaft aus?
- Welche potenziellen Engpässe existieren in einer IT-Landschaft?
    - z.B. Verfügbarkeit, Redundanzen, Medienbrüche, Leistungsengpässe
- Welche Integrationspotenziale verbergen sich in einer IT-Landschaft?

    - z.B. bzgl. gemeinsame Nutzung von Daten, Kommunikation zwischen IT-Komponenten
- Welche Funktionalitäten deckt eine IT-Landschaft ab?
    - z.B. Bereitstellung von Services, Unterstützung von Geschäftsprozessen
- Wie ist die Qualität der IT-Unterstützung?
    - z.B. Verfügbarkeit, Leistung
- Wie ist der organisatorische Kontext der IT?
    - Welche Rollen existieren?
        - z.B. Administrator, Programmierer, Nutzer
    - Welche Rollenträger existieren?
        - z.B. Abteilung, Person, externe Partner
    - Welche Rollenbeziehungen existieren?
        - z.B. wartete, nutzt, repariert, ist verantwortlich
- Wie zukunftssicher ist die eingesetzte IT?
    - Welche Standards werden eingesetzt?
    - Welche Umweltfaktoren und Randbedingungen müssen beachtet werden?
- Wie sicher ist die IT?
    - Welche Zwischenfälle können eintreten?
    - Welche Sicherheitsmaßnahmen existieren?
- Welche IT-Kosten existieren?
    - z.B. Hardware, Software, Personal
- Welchen Nutzen generiert IT?
    - tangibler und intangibler Nutzen
- Welche Daten werden von der IT verwaltet?
    - Wie relevant sind die Daten?
    - Wie und in welchem Umfang werden die Daten genutzt?
    - Wie werden die Daten gespeichert?

Wie an den obigen Fragen zu erkennen ist, kann in der Regel nicht klar zwischen technischen und betriebswirtschaftlichen Analysen unterschieden werden. So ist die Untersuchung einer IT-Landschaft auf Zukunftssicherheit hin sicherlich primär betriebswirtschaftlich motiviert, fußt aber auf der Betrachtung der zugrunde liegenden Technologien und Standards.

Ein großer Teil der hier aufgeführten Fragestellungen kann bereits auf Modellebene weitreichend beantwortet werden. Allerdings wird bei einigen Analysen implizit davon ausgegangen, dass Informationen über Instanzen in der Form von ermittelten Durchschnitts-, Minimal oder Maximalwerten vorliegen. Dies betrifft vor allem Angaben zu Kosten und Mengen, wie z.B. Datenvolumina. Eine weitere Kategorie von Analysen ist ausschließlich dann durchführbar, wenn konkrete Informationen über einzelne Instanzen verfügbar sind. Hierzu zählen bspw. die Ermittlung des tatsächlichen Bestands an Hard- und Software, die Zuordnung von Kosten zu einzelnen Geräten, die Betrachtung von Netztopologien sowie das Generieren der meisten Arten von Kennzahlen. Ein konzeptuelles Modell kann an dieser Stelle nur einen Rahmen festlegen, innerhalb dessen sich solche Analysen bzw. Auswertungen konzipieren lassen. Die tatsächliche Durchführung muss allerdings von einem Werkzeug unterstützt werden, dass auf der Basis eines Modells die Instanzen verwaltet und den Zugriff auf sie ermöglicht.

Im Gegensatz zur Durchführung einer Analyse, die in Modellen enthaltene Informationen lediglich interpretiert, mündet die Implementierung eines Modells in der Konstruktion einer realen IT-Landschaft. Hier gilt analog die Einschränkung, dass auf Modellebene nur ein Rahmen festgelegt werden kann, der die Installation der realen Ausprägungen anleitet, ohne konkrete Vorgaben für jede einzelne Instanz zu liefern. Die oben formulierten analytischen Fragestellungen können prinzi-

piell auch als Leitfaden für die Planung einer IT-Landschaft verstanden werden. Eine hierbei notwendige Anforderung an eine Modellierungssprache ist, dass sie diejenigen Konzepte anbietet, die benötigt werden, um sowohl existierende als auch geplante IT-Landschaften darzustellen. Denkbar sind entweder dedizierte Konzepte, die jeweils nur einen der beiden Aspekte abbilden, oder Konzepte, die für beide Anwendungen geeignet sind. Im ersten Falle können Planungs- und Bestandsinformationen in einem Modell enthalten sein, wodurch aber Redundanzen innerhalb dieses Modells verursacht werden. Bei der Nutzung eines Konzepts für beide Anwendungen muss zwischen Planungs- und Bestandsmodellen unterschieden werden. Hier wird Redundanz innerhalb eines Modells vermieden, jedoch zum Preis von Redundanz über mehrere Modelle.

Letztlich soll hier keine Empfehlung für eine der beiden Herangehensweisen getroffen werden. Es bleibt lediglich die Anforderung an eine Modellierungssprache festzuhalten, dass Konzepte bereitzustellen sind, die sowohl die Planung als auch Analyse von bestehenden IT-Landschaften ermöglichen. Diese sind miteinander kompatibel zu gestalten, so dass ein Abgleich zwischen Bestandsmodellen und Planungsmodellen durchgeführt werden kann. Die Analysekonzepte sollen durch entsprechende Eigenschaften und Beziehungen der Sprachkonzepte mindestens die Beantwortung der oben aufgeführten Fragestellungen auf Basis eines Modells ermöglichen.

3.5 Schnittstellen zu anderen Modellierungssprachen

Wie innerhalb der Diskussion der geforderten Konzepte schon angedeutet, muss eine Sprache für die Modellierung von IT-Landschaften nicht zwingend alle benötigten Konzepte selbst vorhalten. Falls bereits Modellierungssprachen existieren, die einige dieser Aspekte angemessen berücksichtigen, so ist im Sinne der Wiederverwendung effizienter, diese mit der IT-Modellierungssprache zu integrieren. Dies kann über Schnittstellenkonzepte geschehen, die in den Metamodellen aller zu integrierender Sprachen spezifiziert sind. Ausgehend von einem wohldefinierten Kern der Sprache, der von den Konzepten für die Darstellung von IT-Komponenten und deren Beziehungen untereinander gebildet werden sollte, bietet es sich an, Schnittstellen zu Sprachen zur Strategie- und Organisationsmodellierung (Aufbau- und Ablauforganisation) vorzusehen. Als grundlegendes Schnittstellenkonzept zu existierenden Geschäftsprozessmodellierungssprachen (wie z.B. EPK, IDEF3 oder MEMO-OrgML[1]) ist der Geschäftsprozess und das Ereignis zu nennen. Schnittstellen zu Strategiemodellierungssprachen (bspw. MEMO-SML[2]) sind über das Konzept der Strategie zu realisieren.

Die Integration einer Ressourcenmodellierungssprache erscheint ebenfalls sinnvoll. Dies manifestiert innerhalb eines Unternehmensmodells den Ressourcencharakter von IT-Komponenten und ermöglicht eine weitere betriebswirtschaftlich motivierte Sicht auf IT. Weiterhin ist hinsichtlich der Modellierung von Daten und Informationen an eine Verwendung objektorientierter oder relationaler Modellierungskonzepte zu denken. Dazu bieten sich ein erweitertes *Entity Relationship Model* (ERM[3]) bzw. die UML oder MEMO-OML[4] an.

Insgesamt ist von der Modellierungssprache zu fordern, dass sie Zugriff auf alle in Abschnitt 3.3 aufgelisteten Konzepte bietet, indem sie die eigenen Konzepte durch die Integration anderer Sprachen entsprechend der in dieser Arbeit erläuterten Aufgabenstellung erweitert.

---

[1] S: Abschnitt 4.2.1.

[2] S. Abschnitt 4.2.1.

[3] S. [Chen76].

[4] S. Abschnitt 4.2.2.3.

3.6 Integritätsbedingungen

Um die Semantik eines Modell weiterführend zu spezifizieren bzw. einzuschränken, bietet sich die Verwendung von *Integritätsbedingungen* an. Im Kontext der Modellierung von IT-Landschaften, die sich primär mit statischen Aspekten befasst, sollte eine Sprache zu diesem Zweck die Verwendung von *Kardinalitäten* und *Zusicherungen* erlauben.

Kardinalitäten finden bei vielen Modellierungssprachen Verwendung (z.B. ERM, UML-Klassendiagramm), welche für die Erstellung von statischen Sichten konzipiert wurden. Sie erlauben es, auf Typebene Aussagen über die zulässige Anzahl der beteiligten Instanzen zu machen, welche durch die jeweiligen Typen repräsentiert werden. So kann bspw. mit Hilfe von Kardinalitäten in einem Modell ausgedrückt werden, welche maximale Anzahl an Nutzern zu einem Zeitpunkt auf ein Datenbanksystem zugreifen darf (vorausgesetzt ist eine Beziehung zwischen einem Typ *Nutzer* und einem Typ *Datenbank*). Allerdings sind Kardinalitäten nicht für alle Beziehungen sinnvoll und daher nicht generell zu fordern.

Weiterhin sind Kardinalitäten nicht nur in der Form von Annotationen an Beziehungen denkbar, sondern auch für eine generelle Beschränkung der Menge an konkreten Ausprägungen eines Typs zu nutzen (z.B. um auszudrücken, dass eine Applikationskomponente maximal hundertmal instanziert werden darf).

Einschränkungen[1] stellen eine weitere Präzisierung der Semantik bestimmter Konzepte dar. Innerhalb einer Einschränkung können Beschränkungen bzgl. der Zulässigkeit der Beziehung zwischen Konzepten, der Belegung von Attributen sowie andere Bedingungen ausgedrückt werden, welche die Natur der zulässigen Instanzen der betroffenen Typen spezifischer beschreibt. Bspw. ist es hiermit möglich, die Beziehung zwischen einer Applikation und einem Computer nur dann zuzulassen, wenn der Computer auch mit einem Betriebssystem verknüpft ist, welches kompatibel mit der Applikation ist.

Gerade was Einschränkungen betrifft, sind unterschiedliche Formalisierungsgrade denkbar. Im einfachsten Fall sind im Modell natürlichsprachliche Beschreibungen der Einschränkung zu annotieren. Hinsichtlich einer automatisierten Analyse auf Basis eines Modells wäre es allerdings notwendig, einen höheren Formalisierungsgrad zu wählen. Anbieten würde sich hier eine Sprache auf der Basis einfacher Prädikatenlogik, deren Ausdrücke von einem Werkzeug ausgewertet werden können[2]. Eine sinnvolle Forderung ist hier sicherlich die nach einer Kombination aus natürlichsprachlichen und formalen Zusicherungen, wie sie in ähnlicher Form auch in der UML Verwendung finden.

[1] Ähnlich dem *Constraint*-Konzept der UML.

[2] Ein Beispiel für eine solche Sprache ist die OCL (s. [OMG03).

4 Die Information Technology Modelling Language

Die abstrakte Syntax der in diesem Kapitel vorgestellten Modellierungssprache MEMO-ITML (*MEMO Information Technology Modelling Language*), welche gemäß den Anforderungen des letzten Kapitels entworfen wurde, wird in den nachfolgenden Abschnitten in Form eines Metamodells spezifiziert. Dieses Vorgehen begründet sich in der Hauptsache auf den folgenden Aspekten (vgl. auch Kapitel 1):

- Anwendungsnahe Konzepte mit hohem Semantikgrad
- Rekonstruktion der domänenspezifischen Terminologie auf angemessenen Niveau
- Leichte Erlernbarkeit der Sprache
- Vorgaben für Modellierungswerkzeuge

Das semantische Niveau der abzubildenden Konzepte ist für eine formale Spezifikation ungeeignet. Es handelt sich bei den Domänenobjekten nicht um mathematisch exakt zu beschreibende sondern vielmehr um Sachverhalte, die einen gewissen Spielraum bei der Interpretation durch einen Betrachter offen lassen. Der Versuch einer vollkommenen Objektivierung der domänenspezifischen Fachterminologie würde eine Vielzahl dieser durchaus gewünschten Interpretationen ausschließen und wäre darüber hinaus unnötig kompliziert und wenig angemessen. Ein semiformaler, metamodellbasierter Ansatz hingegen wird dem Wesen der abzubildenden Domäne eher gerecht, da er die Terminologie gleichsam hinreichend formal hinsichtlich des Anwendungszwecks der Modelle spezifiziert, als auch dem Anwender eine einfachere Erlernbarkeit der Sprache ermöglicht. Metamodelle können weiterhin als Basis für die Umsetzung der Sprache in einem Modellierungswerkzeug genutzt werden, und so Modelle einer teilautomatischen Validierung sowie Analysen zuzuführen.

Die Sprache ist dahingehend konzipiert, dass sie dedizierte Schnittstellenkonzepte zu den in MEMO bereits vorhandenen Modellierungssprachen aufweist. Auf diese Weise können die dort vorhandenen und reichhaltigen Sprachkonzepte zur Darstellung von Strategien, Geschäftsprozessen und Organisationsstrukturen wiederverwendet und um IT-spezifische Konzepte erweitert werden. Die Notation für die Darstellung des Metamodells ist an die UML angelehnt. Dies begründet sich darin, dass die UML eine weite Verbreitung aufweist und somit eine große Auswahl an Werkzeugen zur Auswahl steht. Weiterhin ist aufgrund der Bekanntheit der UML die Lesbarkeit der Metamodelle für eine größere Zielgruppe gegeben. Die Semantik des Metamodells wird jedoch durch das MEMO-Metametamodell definiert (s. [Fran98a]). Abbildung 3 zeigt den für diese Arbeit relevanten Ausschnitt aus dem Metametamodell. Von Interesse sind vor allem die Spezialisierungen des generischen Metametatyps *MetaConcept*.

Alle Metatypen des ITML-Metamodells sind Instanzen von *MetaEntity*, verfügen über einen Namen und sind optional abstrakt, d.h. sie besitzen ggf. selbst keine Instanzen. Spezialisierung im Sinne der Objektorientierung (Vererbung aller Eigenschaften vom generellen auf den speziellen Typen) ist möglich und darf nicht zyklisch sein. Die Beziehung zwischen zwei Metatypen erfolgt über Instanzen von *MetaAssociationLink*, der jeweils genau einer *MetaEntity* und einem zweiten *MetaAssociationLink* zugeordnet ist. Auf diese Art können Beziehungsenden mit einem Bezeichner versehen werden. Weiterhin verfügen sie über Kardinalitäten (*MetaMultiplicity*) mit minimalen und maximalen Werten, wobei der minimale Wert größer oder gleich 0 sein muss und der maximale Wert größer oder gleich dem minimalen. Die Kardinalitäten beschränken die Menge der zulässigen Instanzen eines Metatyps. In dieser Arbeit wird die der UML entlehnte Schreibweise der Kardinalitäten gemäß Tabelle 1 verwendet.

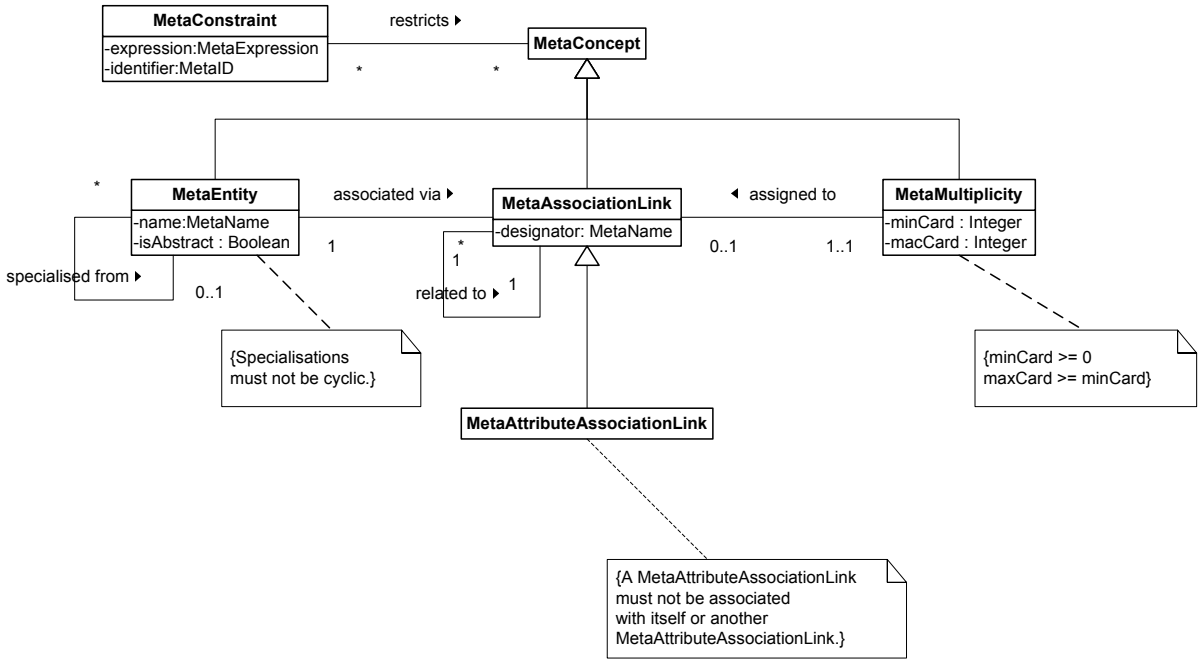

Abbildung 3: Ausschnitt aus dem MEMO-Metametamodell (aus [Fran98a], Fig. 2, S. 8)

Metaattribute werden als Beziehungsenden spezifiziert (*MetaAttributeAssociationLink*). Es gilt die Einschränkung, dass Metaattribute keine Beziehungen mit sich selbst oder anderen Metaattributen eingehen dürfen. Gegenüber der Originaldarstellung in [Fran98a] ist die Einschränkung, dass eine an ein Metaattribut annotierte Mutliplizität eine Kardinalität von mindestens und höchstens 1 haben muss, nicht aufgeführt. Somit dürfen Metaattribute in der ITML generell optional und mehrwertig sein. Es wird im ITML-Metamodell die an die UML angelehnte Schreibweise *attributeName [minCard..maxCard]* verwendet. Wenn keine Attributkardinalität angegeben ist, dann entspricht das dem Wert [0..1], d.h. das Attribut ist optional und hat maximal einen zulässigen Attributwert.

minCard	maxCard	Schreibweise
0	1	0..1
1	1	1
0	*	*
1	*	1..*
n	m	n..m

Tabelle 1: Schreibweise der Kardinalitäten

Jedes Metakonzept kann durch *Constraints* in seiner Verwendung eingeschränkt werden. Das Konzept *MetaConstraint* verfügt hierfür über eine ID und einen Ausdruck, der die Einschränkung erläutern soll. Im Verlauf der Arbeit werden Einschränkungen auf Metamodellebene in der Regel innerhalb des fließenden Texts formuliert.

In den nachfolgenden Abschnitten wird die Spezifikation der abstrakten Syntax im Detail vorgestellt. Dabei wird auch die Semantik der Metaelemente sowie Einschränkungen der Verwendung diskutiert. Im Vorfeld werden die Kernkonzepte der ITML erläutert sowie eine kurze Einführung in MEMO und die dort enthaltenen Modellierungssprachen gegeben, wobei die Integration der ITML mit diesen Sprachen aufgezeigt wird.

4.1 Kernkonzepte der ITML

Alle Metatypen, die im Metamodell der ITML definiert sind, stellen direkte oder indirekte Subtypen des Metatyps *Element* dar (s. Abbildung 4, vgl. auch [Kirc06], S. 231). Sie erben somit sowohl die Attribute von *Element* (s.u.) als auch die Beziehungen zu den Metatypen *Constraint*, *PropertyType* und *ValueType*. Die letzteren beiden erlauben es auf Modellebene benutzerdefinierte Eigenschaften, ähnlich dem Attribut- bzw. Tagged-Value-Konzepten der UML, zu definieren (vgl. [OMG05]).

Element:

- *name*: Name des Typs
- *description*: Beschreibung der intendierten Semantik eines Typs
- *documentation*: ein Verweis auf eine Menge von Dokumenten, welche einen Typen dokumentieren (Spezifikationen, Funktionsbeschreibungen etc.)
- *numInstances*: geplante oder tatsächliche Anzahl der Instanzen eines Typs

ValueType dient der benutzerdefinierten Erstellung von Name/Werte-Paaren auf Typebene.

Bsp.: *Author: Lutz Kirchner*

Date: 2006-10-11

ValueType:

- *name*: Name des Wertetyps
- *value*: Wert des Wertetyps

PropertyType dient der benutzerdefinierten Erstellung von Attributen auf Typebene (Instanzeigenschaften).

Bsp.: *active: Boolean*

color: String

PropertyType:

- *name*: Name des Eigenschaftstyps
- *type*: Typ des Eigenschaftstyps

Das *Constraint*-Konzept dient dem Sprachanwender zur Definition von Einschränkungen für ein Modellelement. Hier ist z.B. an Restriktionen bzgl. zulässiger Beziehungen, Anzahl von Instanzen, Ausprägungen von Attributen etc. zu denken.

Constraint:

- *id*: eindeutiger Identifikator der Einschränkung
- *expression*: einschränkender Ausdruck (natürlichsprachlicher Text oder formaler Ausdruck)

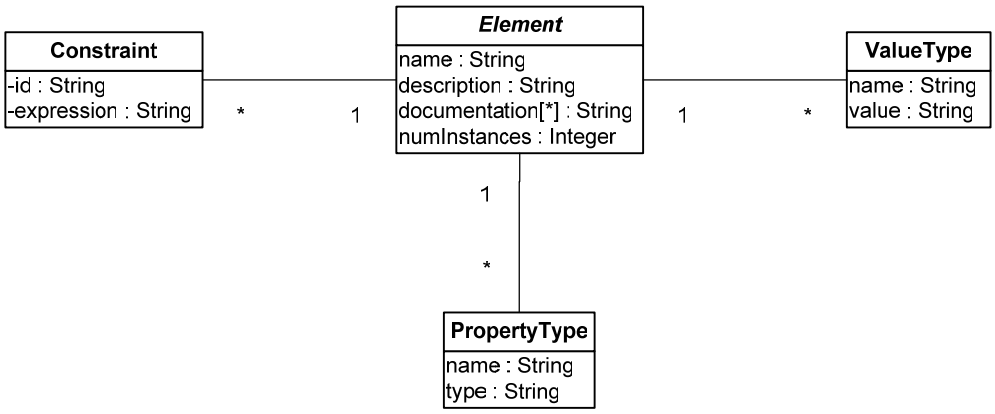

Abbildung 4: Basiselemente der Modellierungssprache

Die zentralen anwendungsnahen Konzepte der ITML gehen aus dem Metamodell in Abbildung 5 hervor[1]. Zuunterst sind Standorte (*Location*) von Hardwarekomponenten (*Hardware*) abgebildet, welche zum Betrieb von Software (*Software*) benötigt werden. Die Software bietet Dienstleistungen (*Service*) an. Geschäftsprozesse (*BusinessProcess*) eines Unternehmens werden durch diese Services unterstützt und realisieren schließlich die Strategien (*Strategy*). Organisationsrollen (*OrganisationalRole*) haben vielfältige Beziehungen zu allen anderen Konzepten. Als eine Zusammenfassung all dieser Kernkonzepte dient das Informationssystem (*InformationSystem*). Falls eine Detailsicht auf die Bausteine eines Informationssystems nicht gewünscht bzw. notwendig ist, steht mit diesem Konzept eine höhere Abstraktion zur Verfügung. Die Integration von Kosten- und Nutzenaspekten erfolgt im weiteren Verlauf des Kapitels.

In Abschnitt 4.2 wird beschrieben, wie die Integration der Sprache in MEMO auf Basis dieser und weiterer ergänzender Konzepte erfolgen kann. Anschließend wird auf die Eigenschaften der abgebildeten Metatypen sowie weiterführende Konzepte in Abschnitt 4.3 näher eingegangen.

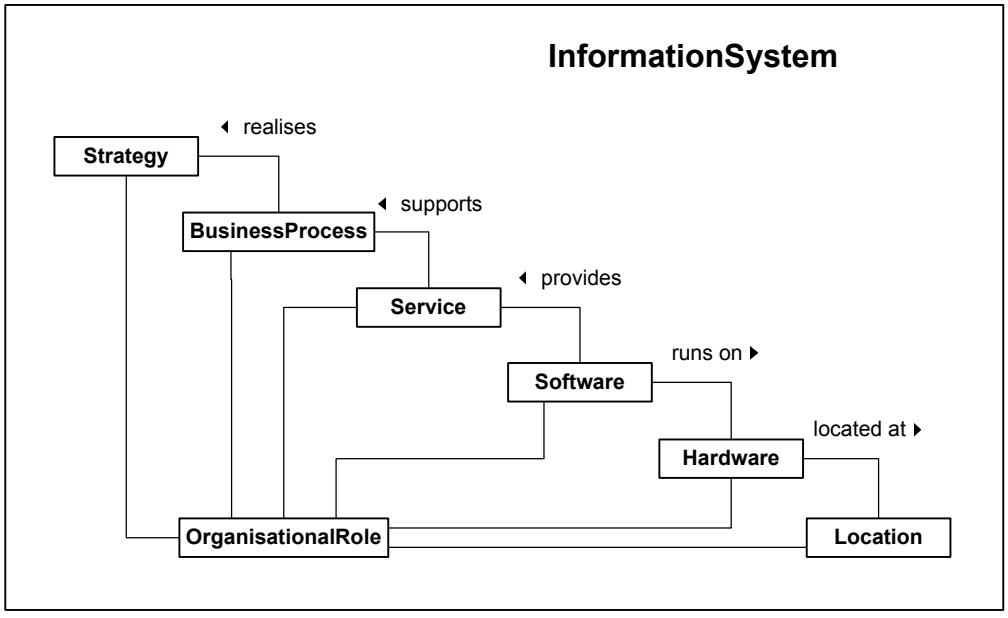

Abbildung 5: Die Kernkonzepte der ITML

[1] Die Bennennung der Konzepte ist aus Gründen der höheren Anschaulichkeit nicht kompatibel zu den Bezeichnern der Metatypen im Metamodell der ITML gehalten.

4.2 Einordnung der ITML in das MEMO-Framework

Wie schon in den vorangegangen Abschnitten angesprochen, ist zur zielführenden Anwendung der ITML die Einbettung in MEMO von großem Vorteil. In den nachfolgenden Abschnitten wird beschrieben, auf welche Art die Integration erfolgt. Zunächst wird MEMO einleitend beschrieben. Anschließend werden die in MEMO enthaltenen Modellierungssprachen vorgestellt und die Schnittstellen zur ITML identifiziert.

4.2.1 Einführung in MEMO

In [Fran94] wird die Methode *MEMO* (Multi-Perspective Enterprise Modelling) zum Entwurf objektorientierter Unternehmensmodelle beschrieben. Die Modelle sollen die Analyse und den Entwurf von betrieblichen Informationssystemen unterstützen. Durch die Verwendung der gleichen anwendungsnahen Konstrukte für die verschiedenen Phasen der Systementwicklung fördert die Methode eine erhebliche Reduktion der vielfältig auftretenden Friktionen zwischen Spezifikation und Implementierung. Es werden in MEMO nicht nur softwaretechnische, sondern insbesondere auch betriebswirtschaftliche Anforderungen an Informationssysteme berücksichtigt. Dazu werden verschiedene Abstraktionsebenen vorgeschlagen, die eine Konzeptualisierung wichtiger Perspektiven auf das Unternehmen sowie eine Integration derselben gestatten. Auf diese Weise kann der Fokus u.a. auf Aspekte der strategischen Planung, auf die Gestaltung von Geschäftsprozessen oder auf die Spezifikation der Klassen eines Objektmodells gerichtet werden. Ergänzend werden Vorgehensmodelle sowie Heuristiken angeboten, welche die Analyse einer Domäne sowie die Erstellung von Unternehmensmodellen anleiten. Ein kompakter Gesamtüberblick über die Methode wird in [Fran02] geboten.

Abbildung 6 zeigt die in MEMO angebotenen *Perspektiven* auf Strategien, Organisation und Informationssysteme sowie die dort jeweils zu betrachtenden *Aspekte* Ressourcen, Struktur, Prozesse, Ziele/Erfolgsfaktoren und Umwelt. Hieraus lassen sich die *Subjekte* eines Unternehmens ableiten, welche durch Modelle beschrieben werden können. Ein Subjekt entspricht im Wesentlichen einem Ausschnitt aus dem gesamten Unternehmensmodell. Da in den weiter oben notierten Perspektiven in der Abbildung das Aggregationsniveau – und somit auch das Abstraktionsniveau - der Beschreibung und Modelle steigt, ist in dieser Richtung ein tendenzielles Absinken des Formalisierungsgrads festzustellen.

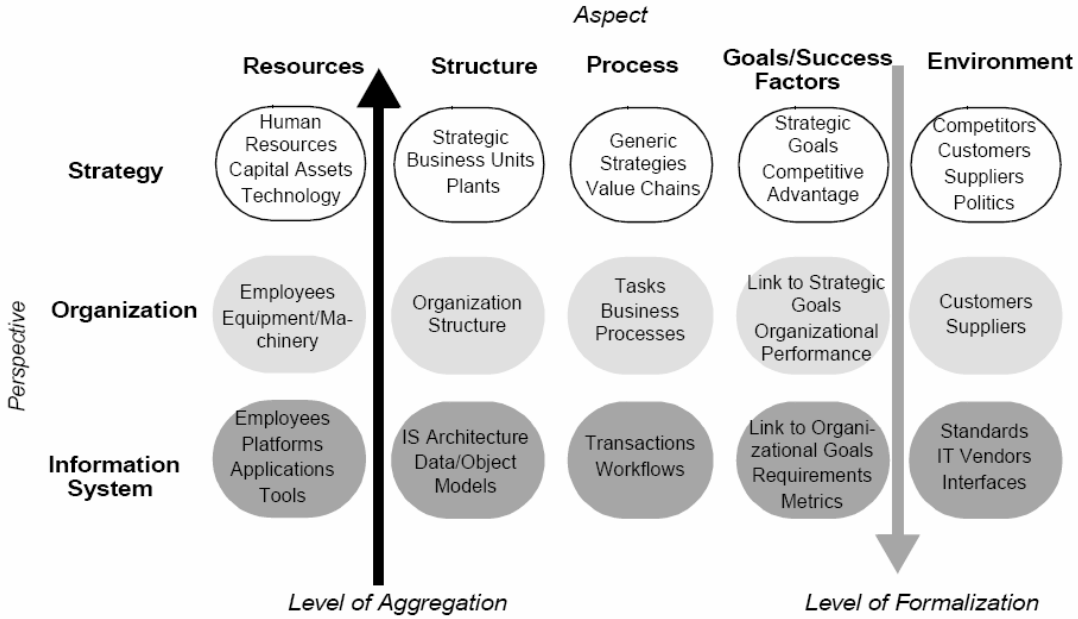

Abbildung 6: MEMO Perspektiven, Aspekte und Subjekte (aus [Fran02], Fig. 1)

4.2.2 Die MEMO-Sprachen

Zur Realisierung eines Unternehmensmodells stehen für jede Sicht Konzepte verschiedener Modellierungssprachen zur Verfügung. Hier sind zunächst die *Strategy Modelling Language* (SML), die *Object Modelling Language* (OML) und die *Organisation Modelling Language* (OrgML) zu nennen (vgl. [Fran99]). Im Rahmen der Weiterentwicklung der Methode wurde die *Resource Modelling Language* (ResML) hinzugefügt. Ergänzend können etablierte Abbildungsformen wie Wertschöpfungsketten (s. [Port85]) und Organigramme genutzt werden.

Die Integration mit den bisher in MEMO vorhandenen Sprachen erfolgt durch die Nutzung gemeinsamer Konzepte auf Metaebene. In Abbildung 7 ist schematisch dargestellt, wie die Integration der Sprachen realisiert wird. Die Metamodelle aller MEMO-Modellierungssprachen sind Instanzen des MEMO-Metametamodells. Sie enthalten Sprachkonzepte, die in dieser oder ähnlicher Form in den anderen Metamodellen ebenfalls enthalten sind oder stehen in einer definierten Beziehung zu Sprachkonzepten anderer Metamodelle. Auf Typebene sind die Modelle angesiedelt, welche eine Anwendung der jeweiligen Sprachen repräsentieren. Hier manifestiert sich die Integration durch die Möglichkeit, Elemente verschiedener Sprachen innerhalb eines Modells zu nutzen, ohne dass die Konsistenz des Gesamtmodells gefährdet wird. Auf dieser Ebene befinden sich Geschäftsprozessmodelle, Darstellungen von Strategien und Ressourcen etc. Das neue Mitglied der Sprachfamilie zur Modellierung von IT-Landschaften, die MEMO-ITML, ist rot hervorgehoben auf der rechten Seite der Abbildung platziert.

Da in den Metamodellen aller MEMO-Sprachen ein Spezialisierungskonzept verwendet wird, das es erlaubt, Eigenschaften eines Metatyps an einen spezialisierten Metatyp zu vererben, verbirgt sich an dieser Stelle eine Herausforderung für die Integration der Sprachen. Es kann durchaus vorkommen, dass Konzepte, welche in mehreren MEMO-Sprachen genutzt werden, in jeweils unterschiedlichen Generalisierungshierarchien angeordnet sind. Z.B. ist *PhysicalDataMedium* (s. Abbildung 14) sowohl eine Spezialisierung des ResML-Konzepts *PhysicalResource* als auch des ITML-Konzepts *Element*. In einem solchen Fall ist darauf zu achten, dass etwaige Eigenschaften bei der daraus resultierenden Mehrfachvererbung kompatibel zueinander sind bzw. Lösungsstrategien für entstehende Konflikte vorgehalten werden. Für die vorliegende Arbeit wird – den aktuellen Stand der Sprachspezifikationen zugrunde legend - davon ausgegangen, dass keine konfligierenden Eigenschaftsdefinitionen vorliegen und somit keine Widersprüche auftreten können. Da sich jedoch alle MEMO-Sprachen in einem Prozess der ständigen Weiterentwicklung befinden, ist jeweils entwicklungsbegleitend sicherzustellen, dass auch in Zukunft keine Widersprüche auftreten.

Wie die Schnittstellen der ITML zu den anderen Sprachen im Einzelnen konzipiert sind, wird nun in den nachfolgenden Abschnitten erläutert. Dabei werden allerdings lediglich die grundlegenden Schnittstellen ausgeführt. Auf weitere wird im Kontext der Beschreibung der ITML hingewiesen (dies bezieht sich insbesondere auf das Konzept *InformationSystem* in Abschnitt 4.3.6).

Die verwendeten Begriffe *Beziehung* und *Assoziation* für die Beschreibung der Zusammenhänge der Metatypen im Metamodell werden im Folgenden synonym benutzt. Der letztere Begriff ist geprägt durch die Objektorientierung, hat aber im Kontext der statischen Modellierung eine gleichwertige Semantik. Der Begriff *Eigenschaften* wird auch für *Attribute* eines Metatypen genutzt. Dabei wird ein Attribut als Eigenschaft eines Metatypen interpretiert, wobei Beziehungen zu anderen Metatypen oder semantische Einschränkungen ebenfalls Eigenschaften sind. Diese Begriffe sind also nicht gleich zu setzen, werden aber im Kontext der Attribute alternativ verwendet.

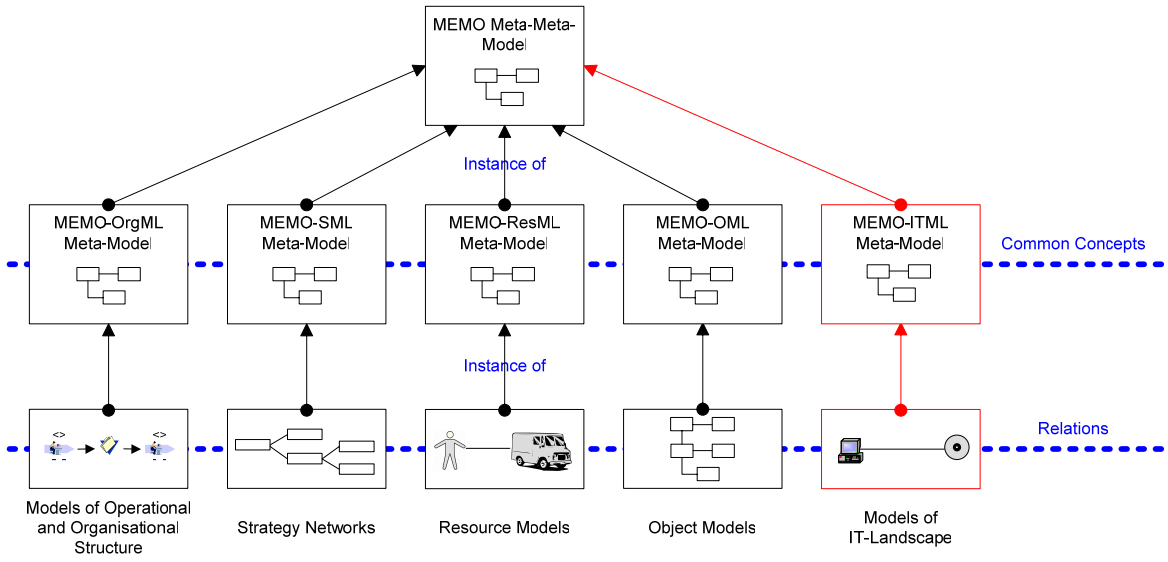

Abbildung 7: Integration der MEMO-Sprachen (in Anlehnung an [Fran99], Fig. 3)

4.2.2.1 MEMO-SML

Die MEMO-SML ist eine Sprache zur Beschreibung von Unternehmensstrategien. Als zentrales Darstellungsmittel dient das Strategienetz (s. [FrLa06]). Eine baumartige Struktur erlaubt, ausgehend von generischen Strategien (Kernstrategien) die Beschreibung von Unternehmensstrategien und deren schrittweise Verfeinerung. Auch werden Maßnahmen als Teil der Strategieimplementierung formuliert. Abbildung 8 zeigt ein exemplarisches Strategienetz mit der Kernstrategie *Reduktion der Vertriebskosten*, weiteren abgeleiteten Strategien (z.B. *Modifikation der Wertschöpfungskette*) sowie zugehörigen Maßnahmen (bspw. *Eliminierung von Zwischenhändlern*).

Abbildung 9 zeigt einen Ausschnitt aus dem Metamodell der SML, ohne auf die Eigenschaften der Konzepte im Detail einzugehen. (s. dazu [FrLa06], Fig. 3). Die rot und mit gestrichelter Umrandung dargestellten Metatypen sind Schnittstellenkonzepte, die in den Metamodellen der beiden zu integrierenden Sprachen - hier SML und ITML - existieren. Metatypen mit schwarzer, durchgezogener Umrandung sind ausschließlich in der mit der ITML zu integrierenden Sprache (in diesem Fall die SML) zu finden. Blau und mit fett hervorgehobenem Rahmen markierte Metatypen sind ITML-spezifisch. Diese Art der Darstellung wird auch in den Abbildungen der folgenden Abschnitte verwendet.

Strategy und *CoreStrategy* stellen Spezialisierungen von *AbstractStrategy* dar. *Measure* steht mit *Strategy* in Beziehung, da Maßnahmen der Umsetzung von Strategien dienen (*relates to*). *Strategy*, welches eines der Kernkonzepte der ITML ist, dient als primäre Schnittstelle zur SML. Die Integration von ITML- und SML-Modellen erfolgt somit hauptsächlich über Beziehungen dort vorhandener ITML-Elemente mit Strategien. Hier ist vor allem auf das Konzept *ProcessType* hinzuweisen, welches nicht zum Metamodell der SML gehört. Es findet sich allerdings in der OrgML und der ITML wieder. Die Beziehung *realises* zwischen einem Geschäftsprozesstyp und einer Strategie drückt aus, dass letztere durch Geschäftsprozesse umgesetzt wird.

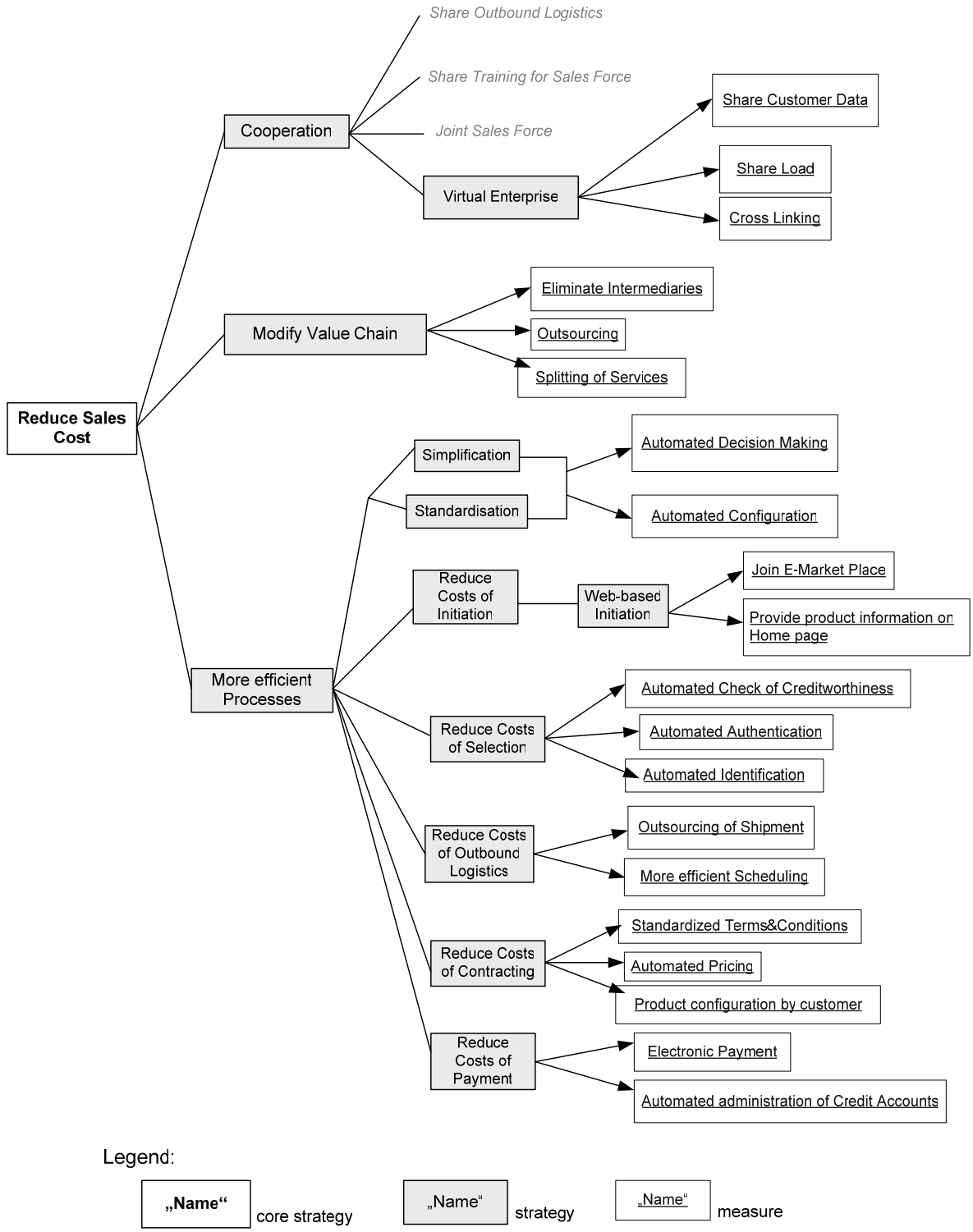

Abbildung 8: Exemplarisches Strategienetz (aus [FrLa06], Fig. 4)

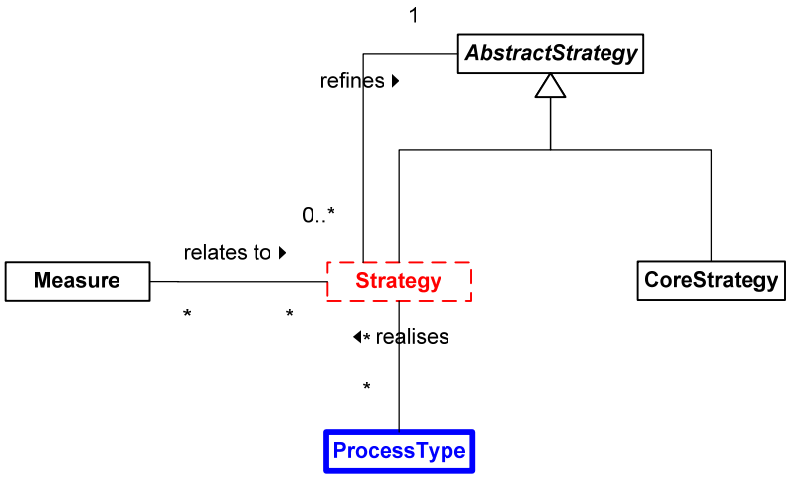

Abbildung 9: Konzepte der SML und ITML

4.2.2.2 MEMO-OrgML

Die MEMO-OrgML ist eine Sprache zur Darstellung von Aufbau- und Ablauforganisation eines Unternehmens (s. [Fran99]). Die Aufbauorganisation kann mithilfe von Organigrammen beschrieben werden. Die Ablauforganisation basiert in der Hauptsache auf Geschäftsprozessen, Ereignissen und Kontrollflüssen. Abbildung 10 zeigt einen beispielhaften Prozess in der OrgML-Notation. Dargestellt ist die Auftragsannahme in einem Unternehmen. Es existieren verschiedene Typen von Ereignissen und Geschäftsprozessen. *Auftrag eingegangen* ist das Startereignis des Prozesses. Es ist mit einer eindeutigen ID *E1* und einem Namen versehen. Das Startereignis initiiert den teilautomatischen Prozess, d.h. einen manuellen Prozess mit IT-Unterstützung, *Auftrag überprüfen*. Der Prozess verfügt analog über die eindeutige ID *1* und einen Namen sowie zusätzlich über einen Verweis auf die verantwortliche Organisationseinheit *Vertrieb*. Aus der Beendigung des Prozesses resultiert ein alternativer Kontrollfluss, entweder zum Endereignis *Auftrag abgelehnt* oder zum nicht näher spezifizierten Ereignis *Auftrag angenommen*. Während ersteres den gesamten Prozess beendet, führt letzteres zu einer parallelen Ausführung der Folgeprozesse *Auftragsbestätigung senden* (vollautomatisch) und *Auftrag weiterleiten* (teilautomatisch). Wenn beide Prozesse beendet sind, wird das Endereignis *Auftragsannahme beendet* erzeugt.

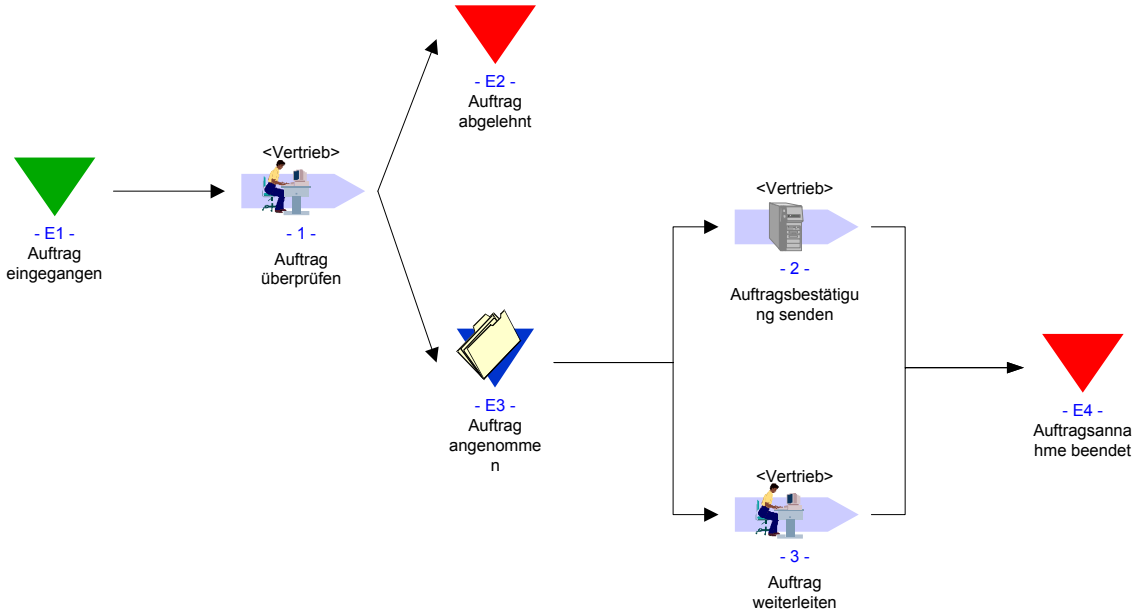

Abbildung 10: Beispiel für einen Geschäftsprozess mit der MEMO-OrgML

Verallgemeinernd betrachtet werden Ereignisse von Geschäftsprozessen erzeugt und können diese wiederum initiieren (*creates* und *triggers* zwischen *ProcessType* und *EventType*). Weiterhin werden Ereignisse durch strukturierte Daten bzw. Informationen beschrieben (*EventType corresponds with Data*). Der Zusammenhang zwischen Daten und Geschäftsprozessen ist im Rahmen der Geschäftsprozessmodellierung ebenfalls von Interesse. Dies wird mit der Beziehung *accesses* zwischen *ProcessType* und *Data* ausgedrückt. Services verschiedener Art unterstützen Prozesse (*Service supports ProcessType*). Organisationsrollen, welche z.B. von einer in der OrgML vorgesehen Organisationseinheit (*OrganisationalUnit*) eingenommen werden können, partizipieren an den Prozessen. Weitere Prozesstypen (z.B. manuell, extern), Ereignistypen (z.B. eingehende Nachricht, Zeitpunkt erreicht) und Kontrollflüsse (z.B. Iteration) sind möglich.

Die Integration der OrgML mit der ITML erfolgt über die gemeinsamen Metatypen *EventType*, *ProcessType* und *OrganisationalUnit*. Die Beziehungen der in Abbildung 11 dargestellten Typen sind vereinfacht dargestellt. Für eine detaillierte Beschreibung wird an dieser Stelle auf Abschnitt 4.3.4 und [Fran99] verwiesen.

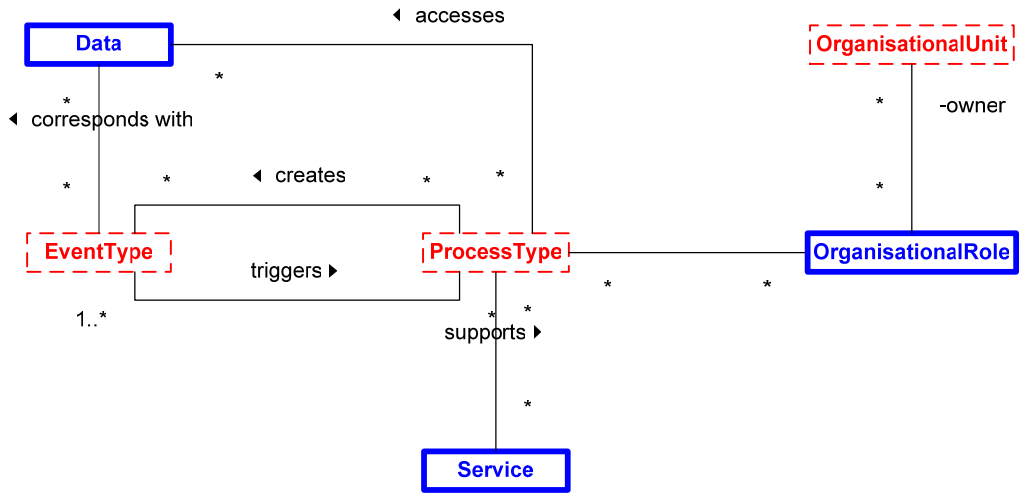

Abbildung 11: Konzepte der OrgML und ITML

Falls andere Sprachen zur Geschäftsprozessmodellierung, wie z.B. die EPK, über ähnliche Konzepte wie die OrgML verfügen, ist deren Integration mit der ITML auf gleiche Art möglich.

4.2.2.3 MEMO-OML

Eine Sprache für die Erstellung von Objektmodellen ist die MEMO-OML (s. [Fran98b] und [Frank98c]). Sie verfügt über die in der Objektorientierung üblichen Konzepte wie Klasse, Attribut, Assoziation etc., aber auch über fortgeschrittene und weniger verbreitete, wie die Delegation. Die Sprache kann sowohl für die Modellierung der Struktur von Softwaresystemen als auch für die Datenmodellierung genutzt werden. Abbildung 12 zeigt ein Beispiel eines OML-Objektmodells. Dieses zeigt einen Ausschnitt aus der Strukturbeschreibung eines Systems zur Verwaltung von Lehrveranstaltungen an einer Universität. Ein *Professor* ist für eine nicht näher spezifizierte Anzahl von Vorlesungen (*EssentialLecture*) verantwortlich (dargestellt mit der Assoziation *in charge of*). Die Vorlesungen verfügen über Eigenschaften in Form von Attributen, wie Titel (*title*) vom Typ *String* u.a. Die Angaben hinter den Typen beschreiben die Kardinalitäten der Attribute. Die Operationen bzw. die von der Klasse angebotenen Dienste (*Services*) sind ausgeblendet. Ein Student nimmt an beliebig vielen konkreten Vorlesungen (*ConcreteLecture*) teil. Die Klasse *ConcreteLecture* repräsentiert eine Rolle, die der Rolleninhaber *EssentialLecture* annehmen kann. Dies wird über die Delegationsbeziehung *represents* zwischen den beiden Klassen ausgedrückt.

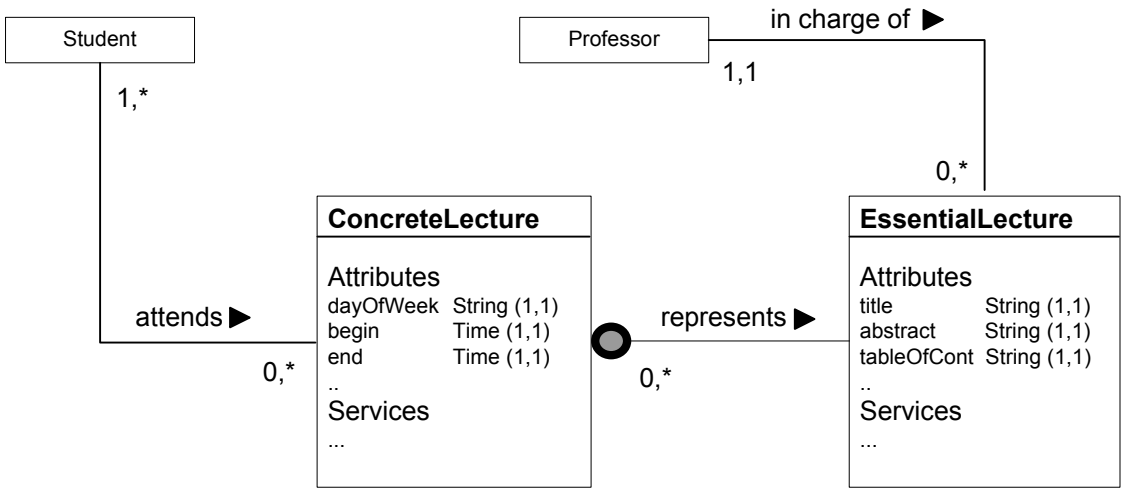

Abbildung 12: Beispiel für ein OML-Objektmodell (aus [Fran98c], Fig. 8)

Die Integration der OML mit der ITML erfolgt primär über das Klassendiagramm bzw. das Objektmodell, das bspw. dazu genutzt werden kann, komplexe oder aggregierte Daten, welche in der ITML durch den Typ *Data* repräsentiert werden, durch Klassen und deren Beziehungen untereinander abzubilden. Andere objektorientierte Sprachen wie die UML oder das *Entity Relationship Model* (ERM) können allerdings auch in dieser Funktion verwendet werden. Weiterhin ist es möglich, mithilfe der OML Softwarestrukturen abzubilden. In dieser Hinsicht weisen objektorientierte Modellierungssprachen generell eine weite Verbreitung auf.

Da es sich bei der OML um eine GPL handelt, können prinzipiell alle Sachverhalte einer IT-Landschaft mit ihr modelliert werden. Innerhalb der hier beschriebenen Methode verweisen wir allerdings lediglich auf die Möglichkeiten zur Modellierung von Daten und Softwarestrukturen, da die weiteren Aspekte mit dedizierten Sprachmitteln abgedeckt werden. Eine Einbeziehung der OML in den Modellierungsprozess ist jedoch dann vorteilhaft, wenn Datenstrukturen hinsichtlich einer Analyse oder der Softwareentwicklung abgebildet werden sollen.

4.2.2.4 MEMO-ResML

Zur Modellierung von Ressourcen im Kontext der Geschäftsprozessmodellierung dient die MEMO-ResML[1]. Hierbei wird nicht speziell auf IT-Ressourcen fokussiert, sondern das Konzept *Ressource* hinsichtlich einer breiteren Anwendbarkeit betrachtet. Daraus resultiert ein generell höheres Abstraktionsniveau der Betrachtung, da auf spezifische Eigenschaften spezieller Ressourcen kaum eingegangen wird. Abbildung 13 zeigt ein Beispiel eines ResML-Modells, in dem über einen Kommunikationskanal zwei Humanressourcen verbunden sind. Konkret ist eine Telefonleitung dargestellt, die unter Verwendung zweier Kommunikationsressourcen (in diesem Fall Telefon und Telefonanlage) von Mitarbeitern und Kunden zur Kommunikation genutzt wird.

[1] Die ResML wird in [Jung07] eingehend beschrieben. Eine Vorstellung der grundlegenden Entwurfsmerkmale der Sprache finden sich in [Jung02] und [Jung03]. In [Jung06] wird das Konzept des Transportkanals detailliert erläutert.

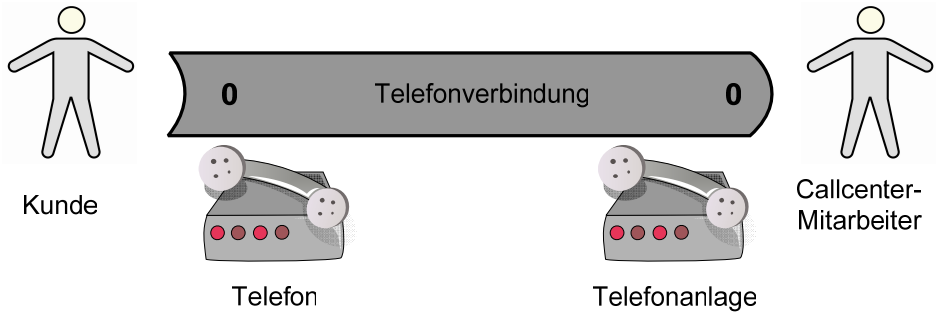

Abbildung 13: Beispiel für ein ResML-Modell (aus [Jung07])

In Abbildung 14 sind die Schnittstellenkonzepte aus den Metamodellen der beiden Sprachen dargestellt. Der Metatyp *Software* der ITML korrespondiert semantisch mit dem gleichnamigen Metatyp der ResML. Im Metamodell der ResML ist Software eine Spezialisierung des Konzepts der intangiblen Ressource (*IntangibleResource*). Dies trifft auch auf das Nutzungsrecht (*BeneficialInterest*) zu. Das ITML-Konzept Lizenz (*License*) steht mit diesem über eine Assoziation *corresponds with* in Beziehung. Somit wird ausgedrückt, dass eine semantische Übereinstimmung der Konzepte existiert, ohne dass Sie als identisch anzusehen sind oder sich in einer Spezialisierungsbeziehung befinden. Daten (*Data*) stehen mit Informationen (*Information*) in Beziehung (*represents*). Außerdem werden sie unter Verwendung eines Formattyps (*FormatType*) strukturiert (*structured by*) und auf physischen Medien (*PhysicalDataMedium*) gespeichert (*stored on*). Letzteres ist eine Spezialisierung des ResML-Konzepts der physischen Ressource (*PhysicalResource*). Dies gilt auch für die ResML-Konzepte *ComputerDevice* und *CommunicationDevice*. Sie finden sich in dieser Form nicht in der ITML wieder, sondern werden als Hardwarekomponenten (*HardwareDevice*) mit entsprechenden Rollen dargestellt. Dieser Sachverhalt wird in Abschnitt 4.3.1 eingehend erläutert. Als Konsequenz daraus ergibt sich die Notwendigkeit, Ausprägungen dieser Metatypen in einem ResML-Modell explizit in ITML-Konzepte zu überführen, falls dies im Einzelfall notwendig erscheint. In Zukunft wird sich dies durch die Realisierung einer angemesseneren Integration der beiden Sprachen aber erübrigen.

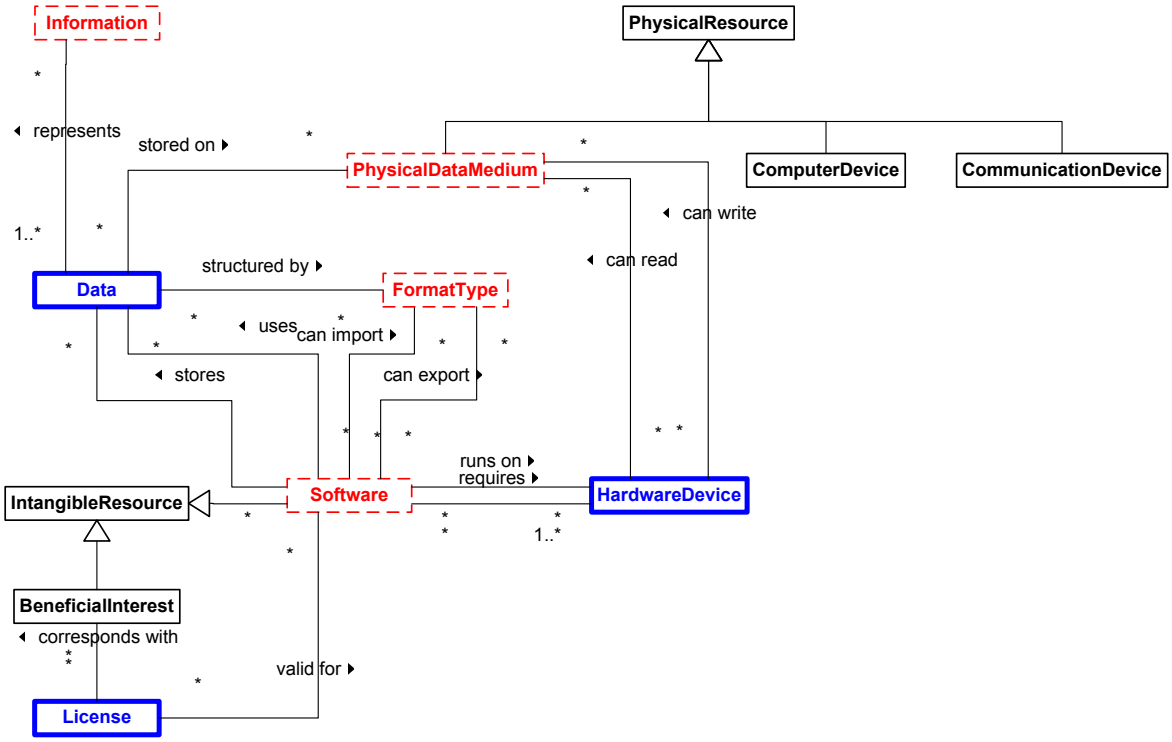

Abbildung 14: Konzepte der ResML und ITML

4.3 IT-spezifische Sprachkonzepte der ITML

In den folgenden Abschnitten werden die spezifischen Konzepte der ITML, welche im Kontext der Integration der Sprache in MEMO teilweise schon angesprochen wurden, im Detail beschrieben. Zuerst werden hardwarespezifische Konzepte erläutert und diese dann um die softwaretechnischen ergänzt. Im Anschluss folgt die Dokumentation der verwendeten Konzepte für die Darstellung der Zusammenhänge von IT mit Aufbau- und Ablauforganisation sowie weitere, kostenspezifische Konzepte. Abschließend wird die Abstraktion des Informationssystems eingeführt und mit zusätzlichen Konzepten für die strategische Planung und Analyse von IT-Landschaften angereichert.

4.3.1 Hardwarespezifische Konzepte

Der Begriff *hardwarespezifische Konzepte* subsumiert alle Abstraktionen der Modellierungssprache, die in den Bereich der tangiblen Ressourcen fallen bzw. deren komplexe Eigenschaften, Funktionen oder Beziehungen beschreiben. Diese Konzepte sind in Abbildung 15 zusammengefasst.

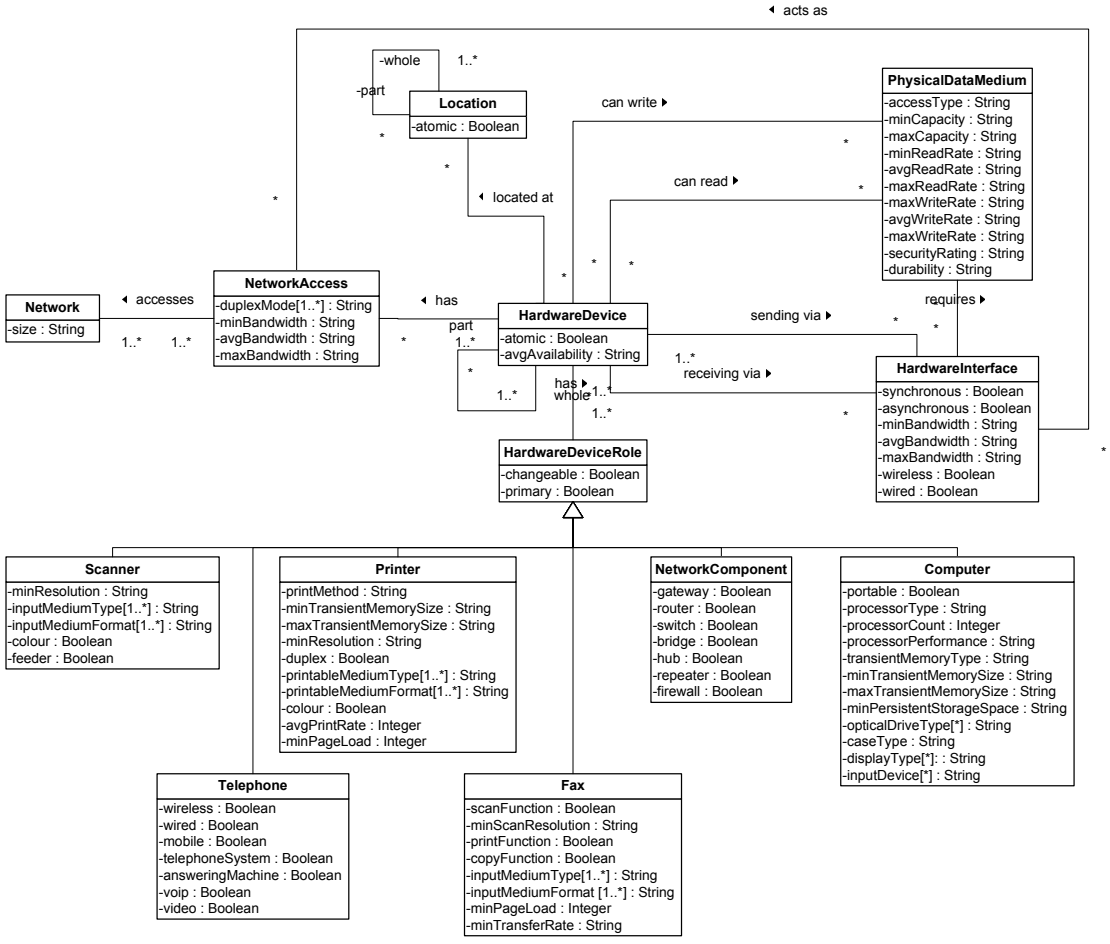

Abbildung 15: Hardwarespezifische Konzepte der ITML

HardwareDevice

Das zentrale Konzept dieses Metamodellausschnitts ist die Hardwarekomponente (*HardwareDevice*). Hierunter wird ein Gerätetyp verstanden, der im Sinne einer generischen IT-Ressource bestimmte Rollen oder Funktionen innerhalb einer betrieblichen IT-Landschaft einnehmen kann. Der Typ hat selbst zwei spezifische Eigenschaften.

HardwareDevice:

- *atomic*: zeigt an, ob ein Gerät dieses Typs noch in weitere Geräte zerlegt werden kann, die eine eigene Identität haben bzw. eine sinnvolle Funktion ausüben können (*false*) oder dies nicht möglich ist (*true*). Ein Computer bspw. enthält Komponenten, die eine eindeutige Funktion besitzen und aus einem System in ein anderes portiert werden können (*atomic = false*). Ein Prozessor allerdings kann nicht mehr weiter zerlegt werden, ohne ihn zu zerstören (*atomic = true*).
- *avgAvailability*: in diesem Attribut kann optional eine geplante oder ermittelte durchschnittliche Verfügbarkeit eines Hardwaregerätetyps gespeichert werden (z.B. 99,8 %)

atomic kann genutzt werden, um die Modularität einer Hardwarekomponente abzubilden. Die zyklische Assoziation mit den Rollen *whole* und *part* an *HardwareDevice* bildet den Sachverhalt ab, dass ein Komponententyp aus weiteren Komponententypen bestehen kann. Hier gilt die Einschränkung, dass am *whole*-Ende der Assoziation kein Gerätetyp verwendet werden darf, der den Wert *true* für die Eigenschaft *atomic* besitzt. Weiterhin gilt die für Ganzes/Teile-Beziehungen übliche Einschränkung (im Folgenden *zyklische Einschränkung* genannt), dass ein Typ nicht gleichzeitig als Teil und als Ganzes mit einem anderen Typen verbunden sein darf. Dabei ist weiter darauf zu achten, dass diese Einschränkung nicht nur für unmittelbare, sondern auch für mittelbare Beziehungen gilt (in der Form *B* Teil von *A*, *C* Teil von *B* => *A* darf nicht Teil von *C* sein).

Als alternative Art der Spezifikation von zusammengesetzten Typen würde sich vordergründig das *Composite Pattern* nach Gamma et al. (vgl. [GHJ+95]) anbieten. So könnte ein genereller und abstrakter Metatyp *AbstractHardwareDevice* in *ElementaryHardwareDevice* und *CompositeHardwareDevice* spezialisiert werden, wobei letzterer Metatyp eine Beziehung zu *AbstractHardwareDevice* hätte, die ausdrückt, dass er aus weiteren zusammengesetzten oder elementaren Hardwaregerätetypen zusammengesetzt sein kann. Diese Alternative wurde nicht gewählt, da es während der Erstellung eines Modells oftmals nicht klar ist, ob ein Hardwaregerätetyp elementar oder zusammengesetzt ist. So kann je nach Modellierungszweck ein Typ mit der Rolle eines Computers elementar sein, oder auch mit dem Fokus auf seine Komponenten als zusammengesetzter Typ betrachtet werden. Durch die hier gewählte Lösung ist es möglich, flexibel auf diesen Sachverhalt mit der Änderung des Wertes von *atomic* zu reagieren, statt einen neuen Typen erstellen zu müssen.

HardwareDeviceRole

Welche Funktion ein Gerätetyp ausübt, wird mithilfe eines Rollenkonzepts beschrieben. Der Metatyp *HardwareDeviceRole* steht über *has* mit *HardwareDevice* in Beziehung und dient der Kapselung der Rollenbeschreibung. *HardwareDeviceRole* ist generisch und kann entweder durch die Instanzierung einer der sechs spezialisierten Rollen auf Metaebene oder durch die Spezifikation benutzerdefinierter Rollen auf Typebene weiterführend beschrieben werden. Die Entwurfsentscheidung für dieses Konzept begründet sich damit, dass IT-Komponenten zum einen oftmals mehrere Funktionen ausführen können und zum anderen diese Funktionalität nicht permanent zur Verfügung stehen muss, d.h. die Eigenschaften eines Gerätes im Zeitverlauf tendenziell variant sind. So kann ein PC je nach eingesetzter Software und Peripherie sowohl als typischer Bürorechner als auch als Telefon oder Fax dienen. Ein Faxgerät mag zusätzlich als Drucker oder Scanner dienen. Vor diesem Hintergrund muss es möglich sein, Funktionalitäten im Sinne von Rollen flexibel einem Gerätetyp zuordnen und ggf. wieder entziehen zu können.

Der generische Metatyp *HardwareDeviceRole* hat folgende Eigenschaften:

HardwareDeviceRole:

- *changeable*: beschreibt, ob die Rolle permanent zugeordnet wird (*false*) oder im Zeitverlauf entfernt werden kann (*true*). Die Rolle des Telefons, die ein PC innehat, kann bspw. durch

Deinstallation der benötigten Software obsolet werden (*changeable = true*). Wenn dieselbe Rolle allerdings an ein traditionelles Telefon gebunden ist, so ist diese nur im Falle eines Defekts zu entfernen, wobei das Gerät insgesamt dann keine sinnvolle Funktion mehr innehat (*changeable = false*).

- *primary*: bei mehren Rollen, die ein Gerät innehaben kann, wird eine primäre Rolle identifiziert. Es gilt die Einschränkung, dass nur eine Rolle, welche einem Gerätetyp zugeordnet ist, den Wert *true* haben darf. Alle anderen müssen den Wert *false* haben. Falls nur eine Rolle existiert, muss *primary* den Wert *true* haben. Hiermit wird zweierlei erreicht: Erstens wird ein Hinweis auf die primäre Natur eines Geräts gegeben (z.B. *Computer* bei einem PC) und zweitens wird das zu verwendende Notationssymbol für den Typ festgelegt.

Die sechs spezialisierten Rollen, welche auf Metamodellebene definiert werden, sind Computer (*Computer*), Drucker (*Printer*), Scanner (*Scanner*), Fax (*Fax*), Telefon (*Telephone*) und Netzwerkkomponente (*NetworkComponent*). Diese Rollen werden als zentral angesehen, da sie für IT-Ressourcen elementare Funktionalitäten kapseln bzw. gängige Metaphern für Funktionalitäten repräsentieren (s. auch Abschnitt 3.3).

Die Rolle *Computer*

Der Begriff Computer beschreibt einen Gerätetyp, der in der Lage ist Informationen (bzw. Daten) entgegenzunehmen, zu verarbeiten und auszugeben[1]. Dabei verfügt er über typische Funktionalitäten zur Ein- und Ausgabe sowie über Datenverarbeitungskomponenten. Die Eigenschaften des Konzepts in der ITML sind die Folgenden:

Computer:

- *portable*: *true*, wenn der Computer als tragbar klassifiziert wird (z.B. ein Notebook), *false* sonst (Desktop PC)
- *processorType*: gibt an, welcher Prozessortyp enthalten ist (X86, AMD64, Intel CoreDuo etc.)
- *processorCount*: Anzahl der Prozessoren im System
- *processorPerformance*: Angabe zur relativen Leistungsfähigkeit des Prozessors (z.B. sehr hoch, hoch, durchschnittlich, unterdurchschnittlich, gering)
- *transientMemoryType*: Typ des verwendeten Hauptspeichers
- *minTransientMemorySize*: minimale Menge an Hauptspeicher, über die ein Computer dieses Typs verfügt (bzw. verfügen muss). Darf nicht größer als *maxTransientMemorySize* sein[2]
- *maxTransiertMemorySize*: maximale Menge an Hauptspeicher, über die ein Computer dieses Typs verfügen kann. Darf nicht kleiner als *minTransientMemorySize* sein
- *minPersistentStorageSpace*: minimale Menge an persistentem Speicher, über die ein Computer dieses Typs verfügt (bzw. verfügen muss)
- *opticalDriveType*: Typ des optischen Laufwerks, über das der Computer verfügt. Ein Computer kann mehrere optische Laufwerkstypen nutzen (z.B. CD-ROM, DVD-RW).
- *caseType*: Typ des Gehäuses (Desktop, MiniTower etc.)
- *displayType*: Typ des Anzeigegeräts. Ein Computer kann mehrere Anzeigegerätetypen nutzen (bspw. TFT-Monitor, CRT-Monitor, Beamer)

---

[1] Dies entspricht prinzipiell dem Begriff der Datenverarbeitungsanlage, wie er bspw. in der Informatik verwendet wird.

[2] Obwohl es sich hier um eine Eigenschaft vom Typ *String* handelt, wird davon ausgegangen, dass die Werte mittels kleiner/größer verglichen werden können. Dies kann prinzipiell mit der Spezifikation eines Aufzählungstyps erreicht werden, dessen Menge an Werten in einer Ordnungsrelation angeordnet sind (z.B. 1024 MB < 2048 MB). In der Sprachspezifikation wurden solche Aufzählungstypen allerdings nicht verwendet, sondern sind entsprechend der konkreten Domänenanforderungen durch Erweiterung der Sprache in einem Werkzeug umzusetzen. Gleiches gilt für alle anderen Eigenschaften, die minimale und maximale Werte auf Basis von *String* spezifizieren.

- *inputDevice*: Typ des Eingabegeräts. Ein Computer kann mehrere Eingabegerätetypen nutzen (Tastatur, Maus, Grafiktablett etc.)

Es ist im Zuge der Modellerstellung darauf zu achten, dass keine Widersprüche zwischen der Belegung der Eigenschaften, die wie *displayType* Komponententypen bzw. wie *processorCount* deren Anzahl beschreiben, und etwaigen Instanzen des Metatyps *HardwareDevice* auftreten. Dies gilt nicht nur für *Computer*, sondern auch für die anderen Rollentypen. Bei Erzeugung eines Typs, der bspw. ein Anzeigegerät in der Form einer Hardwarekomponente definiert, ist *displayType* entweder mit einem semantisch kompatiblen Wert oder mit einem Leerstring zu belegen. Diese Entwurfsentscheidung, welche potenzielle Redundanzen in Kauf nimmt, ist vor dem Hintergrund zu sehen, dass ein Anwender das Konzept des Computers auch nutzen können soll, ohne alle seine Komponenten als Typen spezifizieren zu müssen. Dies fördert eine einfachere Zugänglichkeit der Sprache. Dennoch hat er die Möglichkeit zu einer detaillierteren Spezifikation, wenn dies für den avisierten Verwendungszweck des Modells notwendig erscheint.

Die Rolle *Printer*

Ein Drucker (*Printer*) ist ein Gerät, das elektronisch vorliegende Daten in der Form von Schrift oder Grafik auf Papier oder andere Materialien drucken kann. Der Metatyp spezifiziert folgende Eigenschaften:

Printer:

- *printMethod*: das eingesetzte Druckverfahren (Laser, Tinte, …)
- *minTransientMemorySize*: minimale Menge an Hauptspeicher, über die ein Drucker dieses Typs verfügt (bzw. verfügen muss). Darf nicht größer als *maxTransientMemorySize* sein
- *maxTransientMemorySize*: maximale Menge an Hauptspeicher, über die ein Drucker dieses Typs verfügen kann. Darf nicht kleiner als *minTransientMemorySize* sein
- *minResolution*: minimale Druckauflösung, die ein Drucker dieses Typs anbietet (bzw. anbieten muss)
- *duplex*: *true*, wenn der Drucker das Druckmedium beidseitig bedrucken kann, *false* sonst
- *printableMediumType*: Typ des bedruckbaren Mediums. Ein Drucker kann mehrere Medien bedrucken (z.B. Papier, Folie)
- *printableMediumFormat*: Format des bedruckbaren Mediums. Ein Drucker kann mehrere Formate bedrucken (bspw. DIN A4, DIN A5)
- *color*: *true*, wenn es sich um Farbdrucker handelt, *false* sonst
- *avgPrintRate*: Anzahl der Seiten, die auf Standardmedien durchschnittlich pro Minute bedruckt werden können
- *minPageLoad*: Anzahl der Druckmedien (Seiten), die ein Drucker des Typs intern laden kann

Die Rolle *Scanner*

Ein Scanner (*Scanner*) bildet das Gegenstück zu einem Drucker, indem er Informationen von Papier oder anderen bedruckten Medien wieder in elektronische Daten transformiert. Er verfügt über folgende Eigenschaften:

Scanner:

- *minResolution*: minimale Scanauflösung, die ein Scanner dieses Typs anbietet (bzw. anbieten muss)
- *inputMediumType*: Typ des lesbaren Mediums. Ein Scanner kann mehrere Medien lesen (z.B. Papier, Dia)
- *inputMediumFormat*: Format des lesbaren Mediums. Ein Scanner kann mehrere Formate lesen (bspw. DIN A4, DIN A5)

- *color*: *true*, wenn es sich um Farbscanner handelt, *false* sonst
- *feeder*: *true*, wenn es sich um Einzugsscanner handelt, *false* sonst

Die Rolle *Fax*

Ein Fax (*Fax*) dient der elektronischen Übertragung von Dokumenten. Zu diesem Zweck verfügt es über Funktionalitäten, die es ihm erlauben, bedruckte Medien einzuscannen, Daten zu übertragen und auszudrucken. Falls die Übertragungsfunktion in ein anderes Gerät (bspw. ein PC) integriert ist, verfügt dieses Gerät selbst ggf. nicht über eigene Scan- oder Druckfunktionalitäten, sondern nutzt zu diesem Zweck andere Geräte.

Ein Fax verfügt über die folgenden Eigenschaften:

Fax:

- *scanFunction*: *true*, wenn das Fax eine Scannereinheit besitzt, *false* sonst
- *minScanResolution*: minimale Scanauflösung, die ein Fax dieses Typs anbietet (bzw. anbieten muss). *minScanResolution* muss leer sein, falls *scanFunction = false*
- *printFunction*: *true*, wenn das Fax eine Druckeinheit besitzt, *false* sonst
- *printMethod*: das eingesetzte Druckverfahren (Laser, Tinte, …). *printMethod* muss leer sein, falls *printFunction = false*
- *copyFunction*: *true*, wenn das Fax einen Kopiermodus besitzt, *false* sonst. *copyFunction* muss *false* sein, wenn *scanFunction* oder *printFunction false* sind
- *inputMediumType*: Typ des lesbaren Mediums. Ein Fax kann mehrere Medien lesen (z.B. Papier, Dia). *inputMediumType* muss leer sein, falls *scanFunction = false* ist
- *inputMediumFormat*: Format des lesbaren Mediums. Ein Fax kann mehrere Formate lesen (bspw. DIN A4, DIN A5). *inputMediumFormat* muss leer sein, falls *scanFunction = false*
- *minPageLoad*: Anzahl der Druckmedien (Seiten), die ein Fax intern laden kann. *minPageLoad* muss leer sein, falls *printFunction = false*
- *minTransferRate*: minimale Datentransferrate, die ein Fax realisieren kann (bzw. muss)

Falls die Rolle *Fax* nicht einem dedizierten Gerät, sondern einem Mehrzweckgerät zugeordnet wird, ist darauf zu achten, dass die Werte der Eigenschaften, die sich mit denen anderer Rollen des Geräts inhaltlich überlappen, nicht widersprechen. So darf bspw. bei einem Gerät, das Druck-, Scan- und Faxfunktionalitäten in sich vereint, die Eigenschaft *inputMediumType* der Faxrolle nicht mit dem Wert *Folie* belegt werden, wenn dies in der Scannerrolle nicht vorgesehen ist. Hier besteht eine intendierte Redundanz. Diese wird in Kauf genommen, da die Metapher des Faxgeräts im Bereich der Büroautomation immer noch grundlegend ist und daher als solche auch Teil einer Modellierungssprache sein sollte, die auf die Darstellung von Informations- und Kommunikationssystemen zielt.

Die Rolle *Telephone*

Ein Telefon (*Telephone*) ist ein Gerät zur Stimm- und ggf. Bildübertragung, welches der Kommunikation zwischen zwei oder mehreren Personen dient. Ein Telefontyp hat folgende Eigenschaften:

Telephone:

- *wireless*: *true*, wenn ein Telefon kabellos eingesetzt werden kann, *false* sonst
- *wired*: *true*, wenn ein Telefon kabelgebunden eingesetzt werden kann, *false* sonst
- *mobile*: *true*, wenn ein Telefon mobil, d.h. tendenziell ortsunabhängig als Mobiltelefon genutzt werden kann, *false* sonst
- *telephoneSystem*: *true*, wenn es sich um einen Telefonanlagentyp handelt, *false* bei einen Endgerätetyp

- *answeringMachine*: *true*, wenn im Telefon des Typs ein Anrufbeantworter integriert ist, *false* sonst
- *volP*: *true*, wenn ein Telefon *Voice over IP* fähig ist, *false* sonst
- *video*: *true*, wenn ein Telefon bildübertragungsfähig ist, *false* sonst

wireless muss *true* sein, wenn *mobile true* ist. Wenn bspw. ein schnurloser DECT-kompatibler Telefontyp beschrieben wird, dann ist *wireless* mit *true*, *mobile* allerdings mit *false* zu belegen. Ein schnurloses Telefon ist im Sinne der Semantik des Attributs nicht mobil, da der Abstand zur Basisstation relativ gering sein muss. Es ist weiterhin zulässig, dass ein Telefontyp gleichzeitig kabellos und kabelgebunden einsetzbar ist (*wireless* und *wired true*).

Die Rolle *NetworkComponent*

Eine Netzwerkkomponente (*NetworkComponent*) ist ein Gerät, welches zur Implementierung eines Netzwerks genutzt wird und diesem Zweck dedizierte Funktionen anbietet. Diese IT-Komponente verfügt über folgende Eigenschaften:

NetworkComponent:

- *gateway*: *true*, wenn eine Netzwerkkomponente dieses Typs über die Funktionalität eines Gateways verfügt, *false* sonst
- *router*: *true*, wenn eine Netzwerkkomponente dieses Typs über die Funktionalität eines Routers verfügt, *false* sonst
- *switch*: *true*, wenn eine Netzwerkkomponente dieses Typs über die Funktionalität eines Switch verfügt, *false* sonst
- *bridge*: *true*, wenn eine Netzwerkkomponente dieses Typs über die Funktionalität einer Bridge verfügt, *false* sonst
- *hub*: *true*, wenn eine Netzwerkkomponente dieses Typs über die Funktionalität eines Hubs verfügt, *false* sonst
- *repeater*: *true*, wenn eine Netzwerkkomponente dieses Typs über die Funktionalität eines Repeaters verfügt, *false* sonst
- *firewall*: *true*, wenn eine Netzwerkkomponente dieses Typs über die Funktionalität einer Firewall verfügt, *false* sonst

Bei der Zuweisung konkreter Werte ist darauf zu achten, dass keine Inkonsistenzen im Modell entstehen. So gilt bspw., dass ein Hub in der Regel eine Repeaterfunktionalität beinhaltet (*hub = true* und *repeater = true*) und eine Bridge wiederum einen Hub integriert (*switch = true* und *hub = true* und *repeater = true*). Hier ist also eine grundlegende Vorkenntnis des Modellierers gefordert. Eine detaillierte Beschreibung der Zusammenhänge der verschiedenen Netzwerkkomponenten findet sich in [Tane05]. In einem Modellierungswerkzeug können gültige Kombinationen vordefiniert und zur Überprüfung der Eingaben des Modellierers verwendet werden.

HardwareInterface

Grundlegend für die Kommunikation von Hardwarekomponenten untereinander ist das Vorhandensein einer gemeinsam nutzbaren Kommunikationsschnittstelle. Diese wird in der ITML durch das Konzept *HardwareInterface* repräsentiert. Dabei handelt es sich nicht um ein physisches Gerät sondern vielmehr um die logische Beschreibung der Schnittstelle. Exemplarische Ausprägungen sind USB, RS232, Ethernet u.a. Das Schnittstellenkonzept wurde nicht als Spezialisierung von *HardwareDeviceRole* realisiert, obwohl semantisch durchaus Parallelen existieren. Es handelt sich auf dem hier eingehaltenen Abstraktionsniveau jedoch weniger um eine Rolle im Sinne einer dedizierten Aufgabe der Komponente, als vielmehr um eine strukturierte Eigenschaft, die sie zusätzlich zu ihrer eigentlichen Rolle aufweist. Diese Eigenschaft erlaubt es der Komponente, mit anderen zu kommu-

nizieren, um ihre eigentliche Aufgabe wahrnehmen zu können. In der Realität existieren allerdings auch dedizierte Schnittstellenkomponenten, wie z.B. ein USB-Hub, welche außer der Schnittstellenfunktion keine weiteren relevanten Funktionalitäten aufweisen. Falls ein solches Schnittstellengerät in einem Modell dargestellt werden soll (z.B. hinsichtlich der Zuordnung von Kosten), geschieht dies gemäß der Terminologie der ITML in der Form einer Hardwarekomponente mit einer benutzerdefinierten Rolle (*USB Hub*), welche über eine angebundene (USB-)Schnittstelle verfügt.

Die Eigenschaften eines Schnittstellentyps sind:

HardwareInterface:

- *synchronous*: *true*, wenn eine Schnittstelle dieses Typs synchrone Kommunikation erlaubt, *false* sonst
- *asynchronous*: *true*, wenn eine Schnittstelle dieses Typs asynchrone Kommunikation erlaubt, *false* sonst
- *minBandwidth*: minimale Bandbreite, die eine Schnittstelle dieses Typs anbietet (bzw. anbieten muss)
- *avgBandwidth*: durchschnittliche Bandbreite, die eine Schnittstelle dieses Typs anbietet (bzw. anbieten muss)
- *maxBandwidth*: maximale Bandbreite, die eine Schnittstelle dieses Typs anbietet (bzw. anbieten kann)
- *wireless*: *true*, wenn die Schnittstelle kabellose Kommunikation unterstützt, *false* sonst
- *wired*: *true*, wenn die Schnittstelle kabelgebundene Kommunikation unterstützt, *false* sonst

Es wird differenziert, ob eine Hardwarekomponente sowohl zum Senden als auch zum Empfangen von Daten genutzt werden kann, oder nur eines von beiden realisiert ist (Assoziationen *sending via* und *receiving via*). So ist z.B. eine USB-Schnittstelle prinzipiell bidirektional, eine ältere Centronics-Schnittstelle hingegen nur unidirektional. Ein Drucker, der durch sie mit einem Computer verbunden ist, kann diese Schnittstelle nur zum Empfang von Daten nutzen, während der Computer lediglich senden kann.

PhysicalDataMedium

Die Kommunikation von Hardwarekomponenten bzw. der Austausch von Daten kann nicht nur auf der Basis obiger Schnittstellen erfolgen, sondern auch über physikalische Datenmedien (*PhysicalDataMedium*). Hierunter versteht man Datenträger wie Festplatten, CDs, Disketten etc. Diese Medien können von einer Hardwarekomponente gelesen und/oder geschrieben werden. Letzteres hängt davon ab, ob das Medium prinzipiell beschreibbar ist und die Hardwarekomponente über eine Subkomponente verfügt, welche eine entsprechende Funktionalität aufweist. So benötigt ein PC ein CD-RW-Laufwerk, um einen CD-Rohling zu beschreiben.

Einige Datenmedien benötigen eine Schnittstelle, damit auf sie zugegriffen werden kann. So ist für den Zugriff auf eine externe Festplatte bspw. eine Firewire- oder USB-Schnittstelle nötig. Dieser Sachverhalt wird durch die Assoziation *requires* zwischen *PhysicalDataMedium* und *HardwareInterface* ausgedrückt.

Ein physisches Datenmedium verfügt über die unten aufgeführten Eigenschaften:

PhysicalDataMedium:

- *accessType*: mögliche Zugriffsart (z.B: read-only, read-write, write-once-read-many etc.)
- *minCapacity*: minimale Datenkapazität, über die ein Medium diesen Typs verfügt (bzw. verfügen muss). Darf nicht größer sein als *maxCapacity*
- *maxCapacity*: maximale Datenkapazität, über die ein Medium diesen Typs verfügt (bzw. verfügen kann). Darf nicht kleiner sein als *minCapacity*

- *minReadRate*: minimale Leserate, die ein Medium diesen Typs bietet (bzw. bieten muss). Darf nicht größer sein als *maxReadRate*
- *avgReadRate*: durchschnittliche Leserate, die ein Medium diesen Typs bietet (bzw. bieten kann). Darf nicht kleiner sein als *minReadRate* oder größer als *maxReadRate* sein
- *maxReadRate*: maximale Leserate, die ein Medium diesen Typs bietet (bzw. bieten kann). Darf nicht kleiner sein als *minReadRate*
- *minWriteRate*: minimale Schreibrate, die ein Medium diesen Typs bietet (bzw. bieten muss). Darf nicht größer sein als *maxWriteRate*
- *avgWriteRate*: durchschnittliche Schreibrate, die ein Medium diesen Typs bietet (bzw. bieten kann). Darf nicht kleiner als *minWriteRate* oder größer als *maxWriteRate* sein
- *maxWriteRate*: maximale Schreibrate, die ein Medium diesen Typs bietet (bzw. bieten kann). Darf nicht kleiner sein als *minWriteRate*
- *securityRating*: Angabe zur relativen Sicherheit des Mediums, bspw. in Bezug auf unautorisierte Zugriffe oder das Einbringen von Schadsoftware ins Unternehmen (z.B. hoch, mittel, niedrig)
- *durability*: Angabe zur erwarteten Haltbarkeit des Mediums, d.h. den Zeitraum des möglichen Zugriffs auf die gespeicherten Daten (z.B. 10 Jahre)

Es ist bei der Verwendung des Medienkonzepts darauf zu achten, dass ein Medientyp mit der Eigenschaft *accessType = read-only* nicht mit einem Gerätetyp über die Beziehung *can write* verbunden ist.

securityRating kann im Zuge der Erstellung eines Sicherheitskonzepts helfen, Hinweise auf potenziell unsichere Datenmedien zu geben. So sind bspw. USB-Sticks als besonders unsicher einzustufen, da sie innerhalb eines Intranets hinter der Firewall zum Einsatz kommen und Würmer oder ähnliche Schadsoftware in Umlauf bringen können.

Network

Der Begriff Netzwerk (*Network*) wird hier im Sinne eines Rechnernetzes verwendet. Es bildet eine Abstraktion über eine Menge von zusammengeschlossenen Hardwarekomponenten, welche untereinander kommunizieren können. Somit sind seine Eigenschaften hauptsächlich aus denen der beteiligten Komponenten abzuleiten.

Network:

- *size*: gibt die Größenordnung eines Netzwerks an, z.B. in Form der potenziellen Anzahl an Komponenten oder Usern

NetworkAccess

Eine Hardwarekomponente benötigt einen dedizierten Anschluss an ein Netzwerk, um mit anderen Komponenten im Netz kommunizieren zu können. Dies wird durch den Metatyp *NetworkAccess* abgebildet. Ein Netzwerkzugang ist kein physisches Gerät, sondern analog zu *HardwareInterface* ein logisches Konstrukt. Ein Netzwerkzugangstyp sollte mindestens einem Hardwarekomponententyp zugeordnet sein, da er ohne ein physisches Gerät nicht existieren kann. Er kapselt Eigenschaften einer Hardwarekomponente, wozu auch die Beziehung zu *HardwareInterface* gehört (*acts as*). Diese drückt aus, dass eine Schnittstelle als Netzwerkzugang fungieren kann. Folgende Eigenschaften sind für den Netzwerkzugang spezifiziert:

NetworkAccess:

- *duplexMode*: gibt an, welche Übertragungsmodi genutzt werden (z.B. simplex, halb-duplex, voll-duplex)

- *minBandwidth*: minimale Bandbreite, die ein Netzwerkzugang dieses Typs anbietet (bzw. anbieten muss)
- *avgBandwidth*: durchschnittliche Bandbreite, die ein Netzwerkzugang dieses Typs anbietet (bzw. anbieten kann)
- *maxBandwidth*: maximale Bandbreite, die ein Netzwerkzugang dieses Typs anbietet (bzw. anbieten kann)

Da das Konzept des Netzwerkzugangs in dieser Form eher für die Nutzung im Rahmen einer frühen Entwurfsphase konzipiert ist, sind die vorgegebenen Eigenschaften recht abstrakt gehalten. Im Laufe der Verfeinerung des Modells kann mithilfe des Schnittstellenkonzepts die Art des Zugangs spezifischer beschrieben werden. Hierbei ist darauf zu achten, dass die Belegung der Eigenschaften von *NetworkAccess* mit denen des in Beziehung stehenden Schnittstellentyps kompatibel ist. Ergänzend ist es möglich mittels *ValueType* (s. Abschnitt 4.1) zusätzliche Eigenschaften zu definieren.

Location

Jeder Hardwarekomponententyp kann bestimmten Ortstypen zugeordnet werden, an denen sich dessen Instanzen befinden können bzw. dürfen. Dies wird durch die Assoziation *located at* zwischen *HardwareDevice* und *Location* ausgedrückt. Die erforderliche Genauigkeit der Ortsbestimmung hängt vom Kontext ab, in dem sie erfolgt. So ist es möglich, Orte unter Verwendung eines Koordinatensystems exakt zu beschreiben. Es kann unter Umständen aber auch ausreichen, Räume, Gebäude oder Gebiete – wie z.B. Mobilfunkzellen - als Orte zu identifizieren. Das Konzept *Location* in der ITML ist ein Sammelbegriff für alle möglichen Ausprägungen des Begriffs *Ort*, die im Rahmen der IT-Modellierung relevant sein können. Ein Ortstyp ist optional zusammengesetzt aus weiteren Ortstypen (*whole-part*-Beziehung). So stehen auf einem Werksgelände Gebäude, welche wiederum Büros enthalten. In diesen befinden sich ggf. mehrere Arbeitsplätze. Letztere lassen sich allerdings in unserem Kontext nicht weiter auf sinnvolle Weise disaggregieren.

Location:

- *atomic*: *true*, wenn es sich um einen Ortstyp handelt, der nicht mehr weiter disaggregiert wird, *false* sonst

Wenn *atomic* mit *true* belegt ist, darf der betreffende Typ nicht mit weiteren Ortstypen in der Rolle *whole* an einer Beziehung teilnehmen. Weiterhin gilt für die *whole-part*-Beziehung die zyklische Einschränkung (s.o. unter *HardwareDevice*).

Die oben beschrieben hardwarespezifischen Konzepte werden im folgenden Abschnitt durch softwarespezifische ergänzt.

4.3.2 Softwarespezifische Konzepte

Unter softwarespezifischen Konzepten verstehen wir diejenigen Abstraktionen der Sprache, die intangible Objekte repräsentieren und unmittelbar mit Software in Zusammenhang stehen. Die betreffenden Konzepte sind im Metamodellausschnitt der ITML in Abbildung 16 dargestellt.

Software

Das Konzept der Software wird in der ITML genutzt, um Softwareartefakte jeglicher Art abzubilden, welche auf Hardwarekomponenten installiert sind oder – die Modellierung eines Sollzustands vorausgesetzt – in Zukunft installiert werden sollen. Dazu zählen sowohl Abstraktionen wie Betriebssysteme oder Office-Applikationen, aber auch Softwarekomponenten wie clientseitige ActiveX-Komponenten oder serverseitige Enterprise JavaBeans. Eine Differenzierung dieser unterschiedlichen Ausprägungen auf Metaebene erscheint aus verschiedenen Gründen nicht sinnvoll. Zunächst ist

eine eindeutige Abgrenzung der Funktion, welche ein Softwareartefakt ausüben kann, nicht eindeutig festzulegen. Die oftmals vorgenommene, grundlegende Unterscheidung in Systemsoftware und Anwendungssoftware ist mit einem Blick auf existierende Produkte, wie z.B. Microsoft Windows, nicht durchzuhalten. Es werden mitunter Funktionalitäten, welche in den Bereich der Anwendungssoftware fallen (bspw. Internet Browser, Texteditor) mit der Systemsoftware (bspw. Betriebssystem, Middleware) untrennbar integriert. Weiterhin lässt sich Software mitunter relativ schnell und flexibel anpassen und so ihre Funktionalität verändern oder erweitern. Ein Rollenkonzept, wie es bei den hardwarespezifischen Konzepten umgesetzt wurde, ist vor diesem Hintergrund ebenfalls nicht befriedigend. Die Menge der potenziellen Rollen, die ein Softwareartefakt einnehmen kann, wird als so groß angenommen, dass der resultierende Umfang der Sprache deren Komplexität stark erhöhen und gleichzeitig deren Flexibilität reduzieren würde.

Ebenfalls nicht implementiert wurde die Differenzierung von Anwendungen und Anwendungskomponenten auf Metaebene. Je nachdem welche Definition des Begriffs Softwarekomponente zugrunde gelegt wird, ist im Einzelfall nicht immer eindeutig zu klären, ob es sich um eine Komponente oder eine Applikation handelt. So ist bspw. Microsoft Word zunächst als Anwendung zu betrachten, kann allerdings darüber hinaus als Komponente der Microsoft Office Suite interpretiert werden. Zusätzlich können Word-Funktionalitäten in der Form von DCOM-Komponenten in anderen Anwendungen genutzt werden, sodass hier eine weitere Überschneidung existiert. In dieser Arbeit soll nicht versucht werden, eine allgemeine Komponentendefinition zu schaffen, die eine trennscharfe Kategorisierung von Softwareartefakten zulässt. Stattdessen wird eine Differenzierung über die angebotenen Dienste und Abhängigkeiten unterschiedlicher Software angestrebt. Dies geht auch dahingehend mit der gängigen Sicht auf Komponenten konform, dass diese primär über ihre Schnittstellen, also das Angebot an Services, charakterisiert werden. Die an dem Metatyp *Software* angebrachte rekursive Beziehung *requires* zeigt Abhängigkeiten auf, bspw. von einer Enterprise JavaBean zu einem Application Server, welche die bedingte Lauffähigkeit von Softwarekomponenten ausdrücken können. Somit ist auch diese Besonderheit von Komponenten in der Sprache berücksichtigt. Die Semantik der Assoziation ist allerdings breiter gefasst, da auch die Abhängigkeit einer Anwendung von dem darunter liegenden Betriebssystem auf diese Art abgebildet werden kann. Falls keine derartigen Abhängigkeiten bestehen, sondern ein weiter gefasster Komponentenbegriff zugrunde gelegt wird, kann die *whole-part*-Beziehung genutzt werden. Als Beispiel hierfür möge ein Datenbanksystem dienen, das typischerweise aus Komponenten zusammengesetzt ist, die keine spezielle Laufzeitumgebung benötigen: dem Datenbankmanagementsystem und einer Menge von Datenbank-Clients. Somit ist *requires* als eine Abhängigkeit im Sinne der Lauffähigkeit eines Softwareartefakts zu interpretieren und im Gegensatz dazu die *whole-part*-Beziehung als logische Zusammenfassung zusammengehöriger Artefakte. Für Letztere gilt die zyklische Einschränkung (s. Abschnitt 4.3.1, *HardwareDevice*). Die gleichzeitige Verwendung von *whole-part*- und *requires*-Beziehungen ist prinzipiell zulässig.

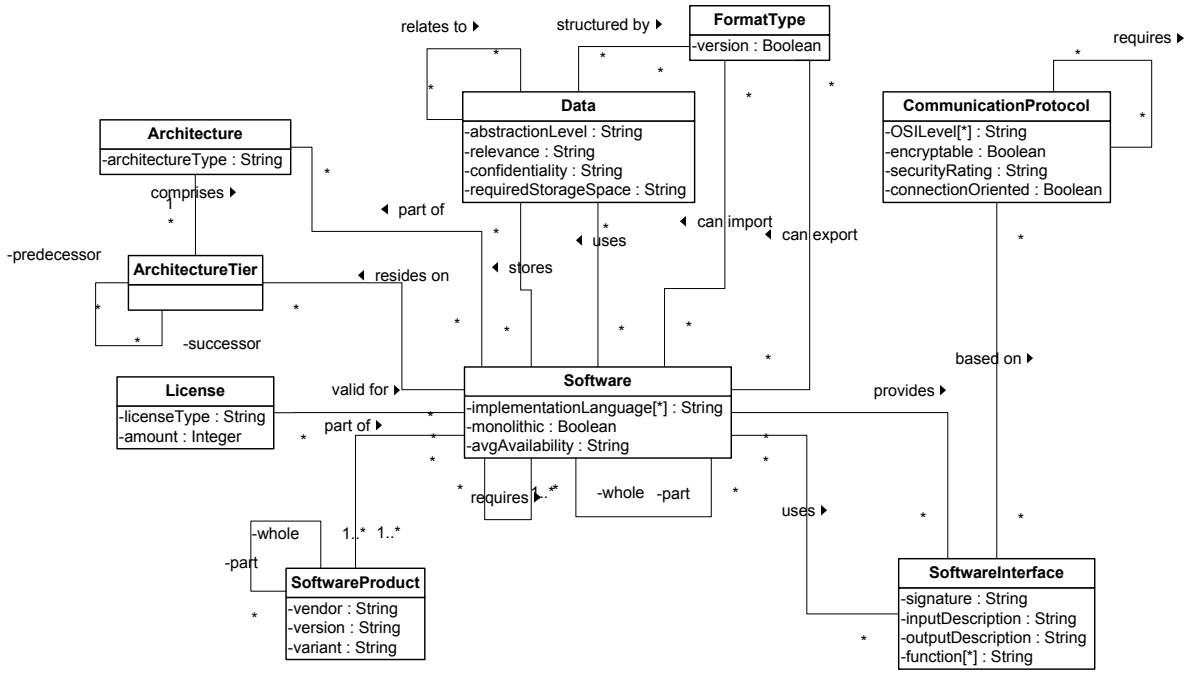

Abbildung 16: Softwarespezifische Konzepte der ITML

Das Konzept *Software* verfügt über die folgenden Eigenschaften:

Software:

- *implementationLanguage*: die Programmiersprache(n), in der ein Softwaretyp implementiert ist. Die Angabe ist optional, da nicht immer von Interesse oder eindeutig
- *monolithic*: *true*, wenn dem Softwaretyp eine monolithische Architektur zugrunde liegt, *false* sonst
- *avgAvailability*: in diesem Attribut kann optional eine geplante oder ermittelte durchschnittliche Verfügbarkeit eines Softwaretyps gespeichert werden (z.B. 99,8 %)

Das Attribut *monolithic* bezieht sich auf die grundlegende Konzeption der Software. Bei einer klassischen Anwendung, die alle Funktionen in einem Softwareartefakt (Modul) kapselt, wäre sein Wert *true*. Bei einer komponentenbasierten Anwendung (*false*) lässt sich zusätzlich mit der Beziehung *part of* zu *Architecture* (s.u.) darstellen, in welchem Komponentenmodell sie eingesetzt werden kann. Die Assoziation *resides on* zu *ArchitectureTier* kann genutzt werden, um sie einer konkreten Architekturschicht zuzuordnen.

Architecture

Unter Architektur verstehen wir im Rahmen dieses Beitrags eine Beschreibung aller Komponenten eines Systems, deren Funktionen, Beziehungen und Schnittstellen[1]. Die in unserem Kontext besonders interessante Ausprägung des Begriffs ist die Informationssystemarchitektur, speziell der Anwendungs- oder Softwarearchitektur. Deren Ausprägungen können bspw. Komponentenmodelle wie J2EE oder CORBA Components etc. sein, aber auch allgemeinere Beschreibungen, wie eine dreischichtige Webarchitektur oder eine Client/Server-Architektur. Das Konzept *Architecture* in der ITML dient der Zuordnung von Software zu Architekturen. So kann bspw. eine Enterprise JavaBean mit einer Webarchitektur in Beziehung gesetzt werden. Dies hat ergänzend dokumentativen Charakter, da die gesamte IS-Architektur durch ein umfassendes ITML-Modell, speziell durch Softwaretypen und

[1] Vgl. z.B. [LHM95], S. 58 und [MEH01], S. 108.

deren Beziehungen, dargestellt wird. Mithilfe des Sprachkonzepts kann der zugrunde liegende Architekturtyp jedoch explizit benannt und beschrieben werden. Dazu wird die Eigenschaft *description* genutzt, welche von dem generellen Konzept *Element* geerbt wird. *Architecture* verfügt über ein weiteres Attribut.

Architecture:

- *architectureType*: Art der Architektur (z.B. Anwendungsarchitektur, Komponentenmodell). Dient der weiteren Spezifizierung, falls es sich um eine spezielle zu benennende Softwarearchitektur handelt

ArchitectureTier

Da Architekturen oftmals explizite Schichten bzw. Sichten aufweisen, steht das Konzept *ArchitectureTier* zur Verfügung, um dies zu dokumentieren. Wie oben schon erläutert, können Softwaretypen einer Schicht zugeordnet werden, welche wiederum einer Architektur zugehörig ist (über die Assoziation *comprises*). Hier sind über die Beschreibung der Schicht hinaus keine speziellen Eigenschaften vorgesehen. Die Ordnung der Architekturschichten kann über die zyklische Assoziation an *ArchitectureTier* ausgedrückt werden, welche mit den Rollen *predecessor* und *successor* annotiert ist.

License

Für jedes Softwareartefakt sollte eine gültige Lizenz eingeplant bzw. vorhanden sein. Die Art der Lizenz kann variieren. Bei selbst erstellter Software ist keine notwendig, ebenso wenig bei Freeware. Dieser Sachverhalt sollte dennoch entsprechend im Modell vermerkt werden (z.B. durch einen Typ *FreeLicense*). Es existieren Lizenzmodelle in Anlehnung an das Shareware-Prinzip, wie die GNU GPL[1], weiterhin Einzelplatz- oder Einzelsystemlizenzen (*node locked*), Mehrplatz (*floating*) und Unternehmenslizenzen etc. Diese Ausprägungen des Lizenztyps können auf Typebene spezifiziert und einem oder mehreren Softwaretypen über die Beziehung *valid for* zugeordnet werden. Ein Softwaretyp kann auf mehrere Lizenztypen verweisen.

License:

- *licenseType*: Ausprägung des Lizenztyps (node locked, floating etc.)
- *amount*: Anzahl der verfügbaren oder geplanten Lizenzen eines Typs

SoftwareProduct

Ein Softwareprodukt kann als die Summe der Softwareartefakte verstanden werden, welche vom Hersteller oder Händler zur Auslieferung an den Nutzer vorgesehen sind[2]. Mehrere Softwaretypen können einem Softwareprodukt zugeordnet werden (Assoziation *part of*). Weiterhin kann ein Produkt logisch aus weiteren Produkten zusammengesetzt sein (darzustellen mittels der *whole-part*-Beziehung an *SoftwareProduct*). Bspw. gehören zu dem Produkt Microsoft Office in der Regel eine Textverarbeitung (Word) und eine Tabellenkalkulation (Excel) u.a. Diese sind ebenfalls Produkte und alternativ einzeln zu erwerben. Wenn diese Teilprodukte installiert sind und dem Benutzer zur Verfügung stehen, repräsentieren sie Ausprägungen der Softwaretypen Textverarbeitung bzw. Tabellenkalkulation, welche mit den jeweiligen Softwareprodukten in Beziehung gesetzt werden können. Auch hier gilt die zyklische Einschränkung aus Abschnitt 4.3.1, *HardwareDevice*.

Der Typ *SoftwareProduct* hat folgende Eigenschaften:

---

[1] *GNU General Public License*, eine weit verbreitete Lizenzform, welche für nicht kommerzielle Zwecke eine weitgehende freie Verwendung der Software vorsieht. S. http://www.gnu.org/licenses/gpl.html

[2] Diese Definition lehnt sich an die Ausführungen des ISO/IEC 14598 Standards - Part 1 für die Evaluierung von Softwareprodukten an (s. [ISO99]).

SoftwareProduct:

- *vendor*: Hersteller des Softwareprodukts
- *version*: Version des Softwareprodukts
- *variant*: Variante des Softwareprodukts (z.B. Personal Edition, Professional Edition)

SoftwareInterface

Softwareartefakte bieten Schnittstellen (*SoftwareInterface*) an, über die andere Softwareartefakte deren Funktionalität nutzen können. Die Spezifikation einer Schnittstelle kann syntaktisch über deren Signatur, d.h. über die Menge der aufzurufenden Funktionen sowie deren Parameter und Rückgabetypen, erfolgen. Weiterhin kann die Semantik der zu sendenden und empfangenden Daten beschrieben werden. *SoftwareInterface* ist mit den Assoziationen *provides* und *uses* mit *Software* verbunden.

SoftwareInterface:

- *signature*: Signatur der Schnittstelle
- *inputDescription*: Beschreibung der Daten, die über eine Schnittstelle empfangen werden
- *outputDescription*: Beschreibung der Daten, die über eine Schnittstelle gesendet werden
- *function*: Menge der Funktionalitäten, die über eine Schnittstelle genutzt werden können bzw. sollen

Schnittstellen können auf unterschiedlichen Abstraktionsebenen definiert werden. So kann bspw. ein Datenbankmanagementsystem über eine Schnittstelle verfügen, welche die gesamte Bandbreite möglicher Datenbankzugriffe abdecken kann. Allerdings ist es auch möglich, eine Programmierschnittstelle zu beschreiben, die der einer objektorientierten Klasse entspricht und über eine entsprechend geringere Mächtigkeit verfügt. Die Wahl des Abstraktionsniveaus bleibt dem Anwender überlassen. Es muss allerdings auf eine in dem jeweiligen Kontext einheitliche Semantik der Schnittstellenbeschreibungen geachtet werden, da sonst die Überprüfung der möglichen Kommunikation zwischen Softwareartefakten auf Basis eines Modells kaum möglich ist.

CommunicationProtocol

Eine Kommunikation zwischen Softwarekomponenten bzw. Softwareartefakten kann nur dann erfolgen, wenn eine gemeinsame Sprache zugrunde liegt. Diese wird durch das zu verwendende Kommunikationsprotokoll definiert. Ein Beispiel für ein weit verbreitetes Standardprotokoll ist das TCP/IP-Protokoll für die Kommunikation in Internet und Intranets. Auf einer höheren Protokollebene finden sich z.B. ODBC oder SOAP als Ausprägungen wieder. Es existiert neben den Standardprotokollen aber auch eine große Zahl von proprietären Protokollen. Der Metatyp *CommunicationProtocol* dient der Darstellung von beliebigen Protokollen, die in einer IT-Landschaft zur Kommunikation genutzt werden. *SoftwareInterface* ist mit diesem Metatyp über die Assoziation *based on* verbunden, um auszudrücken, welche Protokolle ein Softwaretyp zur Kommunikation über eine Schnittstelle nutzt bzw. nutzen kann. Weiterhin kann ein Protokoll ein anderes voraussetzen (z.B. benötigen TCP und UDP das IP-Protokoll). Dies wird mit der zyklischen Beziehung *requires* an *CommunicationProtocol* dargestellt. Der Metatyp verfügt über folgende Attribute:

CommunicationProtocol:

- *OSILevel*: OSI-Schicht[1], auf der ein Protokoll angeordnet ist. Da nicht alle Protokolle kompatibel zu dem OSI-Modell sind, kann hier eine beliebige Anzahl an Schichten angegeben werden
- *encryptable*: *true*, wenn das Protokoll eigene Verschlüsselungsmechanismen mitbringt (z.B. SSL), *false* sonst
- *securityRating*: Angabe zur relativen Sicherheit des Protokolls, bspw. in Bezug auf Abhörbarkeit (z.B. hoch, mittel, niedrig)
- *connectionOriented*: *true*, wenn es sich um einen verbindungsorientiertes Protokoll handelt (z.B. TCP), *false* sonst

SoftwareInterface und *CommunicationProtocol* sind sich ergänzende Abstraktionen zur Beschreibung der Kommunikation zwischen Softwareartefakten. Die Softwareschnittstelle sollte genutzt werden, um die Syntax der einzelnen Aufrufe, die Semantik der auszutauschenden Daten sowie die angebotenen Funktionalitäten zu dokumentieren. Ein Protokoll ergänzt diese Informationen um die Spezifikation weiterer Regeln wie die Abfolge von bestimmten Nachrichtentypen oder Vorgaben, welche Antwortnachrichten auf Standardanfragen generiert werden müssen. Bei der Verwendung beider Konzepte ist darauf zu achten, dass die Schnittstellenbeschreibung kompatibel mit den Vorgaben des zu verwendenden Protokolls ist.

Data

Daten in vielfältiger Form werden von Software sowohl gespeichert als auch genutzt, um interpretiert, modifiziert und anschließend ggf. ausgegeben zu werden. Dies kann mittels der beiden Beziehungen *stores* und *uses* zwischen den Metatypen *Software* und *Data* dargestellt werden. Daten können untereinander in Beziehung stehen (relates to). Der Metatyp *Data* verfügt über folgende Eigenschaften:

Data:

- *abstractionLevel*: weist auf das Abstraktionsniveau der Daten hin (z.B. hoch für Konzepte wie Kunde, mittel für ein Formular, niedrig für Zeichenketten)
- *relevance*: Einschätzung der Relevanz der Daten für das Unternehmen (z.B. hoch, mittel, niedrig)
- *confidentiality*: Aussage über den Grad der Vertraulichkeit der Daten (z.B. hoch, mittel, niedrig)
- *requiredStorageSpace*: Angabe einer zu erwartenden oder tatsächlichen Speicheranforderung der Daten

Mithilfe von *abstractionLevel* kann während der Planung oder Analyse eine erste Einschätzung der Inhalte von Datentypen[2] geschehen. Eine angemesse Strukturierung und Darstellung von Dateneigenschaften und –strukturen soll über die Integration von Datenmodellierungskonzepten erfolgen. Eine explizite Modellierung höherer Konzepte, wie Information und Wissen, liegt nicht im Fokus der Arbeit und erfolgt an anderer Stelle innerhalb des MEMO-Projekts.

Daten können für ein Unternehmen von unterschiedlicher Wichtigkeit sein sowie der Vertraulichkeit unterliegen. So sind Kundendaten in der Regel sowohl äußerst wichtig als auch vertraulich, da sie bei Verlust schwer zu ersetzen sind und ihre Weitergabe strengen gesetzlichen Auflagen unterliegt. Allgemein verfügbare Daten wie Firmenadressen sind weniger relevant, da leicht zu beschaffen

[1] Das *ISO OSI-Schichtenmodell* (OSI = *Open Systems Interconnection*) ist ein Referenzmodell, welches von der ISO publiziert wurde. Es zeigt idealtypische Schichten auf, welche bei der Kommunikation von IT-Systemen identifiziert werden können (s. [ISO94]).

[2] Hier ist nicht der Datentyp im Sinne der Informatik gemeint, sondern die Ausprägungen des Metatyps *Data*.

(Telefonbuch) und allgemein zugänglich, also in der Regel auch nicht vertraulich zu behandeln[1]. Die Abbildung von Zugriffsberechtigungen auf Daten seitens Organisationsrollen wird in Abschnitt 4.3.4 skizziert.

Die Angabe des benötigten Speicherplatzes ist in Verbindung mit Ausprägungen von *Data*, wie z.B. Kundendaten, sinnvoll. Hier kann vermerkt werden, wie viel persistenter Speicher für die Daten eingeplant werden muss. Potenzielle Engpässe können auf dieser Basis aufgedeckt werden.

FormatType

Daten werden in der Regel in einem identifizierbaren Format vorgehalten, um ihre Lesbarkeit und Interpretierbarkeit zu gewährleisten. Dies wird mit der Beziehung *structured by* zwischen *Data* und *FormatType* ausgedrückt.

FormatType:

- *version*: Version des Formattyps

Die Version ist vor allem dann von Interesse, wenn es sich um einen Standardformattypen, wie XML, handelt, der in unterschiedlichen Versionen genutzt wird. Die Kenntnis der Version ist für eine korrekte Interpretation der Daten grundlegend.

FormatType verfügt über zwei Assoziationen zu *Software*, *can import* und *can export*. Welche Formate ein Softwaretyp lesen und schreiben können muss um bestimmte Daten zu verarbeiten, geht indirekt über die Beziehung zu *Data* und dessen Beziehung zu *FormatType* hervor. Welche Formate ein Softwaretyp generell lesen und schreiben bzw. importieren und exportieren kann, wird durch die beiden erstgenannten Assoziationen abgebildet.

Der nächste Abschnitt zeigt nun die Integration der hardwarespezifischen Konzepte aus Abschnitt 4.3.1 mit den soeben vorgestellten softwarespezifischen auf.

4.3.3 Integrations- und ergänzende Konzepte für Hard- und Software

Da softwarespezifische und hardwarespezifische Aspekte zwar oftmals unabhängig voneinander betrachtet werden können, die betriebliche IT aber nur im Zusammenspiel dieser beiden Kategorien von IT-Komponenten funktioniert, sind deren vielfältige Querbezüge in der Sprache zu berücksichtigen. Die hierfür im Metamodell spezifizierten Beziehungen gehen aus Abbildung 17 hervor.

In ihr finden sich die Metatypen wieder, die bei der Integration der beiden Sichten eine Schnittstellenrolle innehaben. Die zentralen Beziehungen werden durch *requires* und *runs on* zwischen *Software* und *HardwareDevice* realisiert. Hiermit wird ausgedrückt, dass ein Softwaretyp einen bestimmten Hardwaretypen benötigt, um ausgeführt zu werden (*requires*), bzw. ein Softwaretyp aktuell auf einem Hardwaretypen läuft oder dies geplant ist (*runs on*). Eine weitere Beziehung *enables* existiert zwischen *HardwareDeviceRole* und *Software*. Durch sie kann dargestellt werden, welche Software ggf. nötig ist, um eine Hardwarerolle zu realisieren. So ermöglicht bspw. eine VoIP-Applikation wie Skype o.ä. einem Personal Computer die Rolle eines Telefons einzunehmen.

PhysicalDataMedium wird mit *Data* durch die Assoziation *stored on* verbunden. So wird abgebildet, welche Daten auf welchen Datenträgertypen zur Speicherung vorgesehen sind. Ein wichtiges Schnittstellenkonzept ist *CommunicationProtocol*. Dieses ist nicht nur für die Kommunikation auf Softwareebene relevant, sondern auch auf Hardwareebene. *NetworkAccess* und *HardwareInter-*

[1] Dies gilt zunächst nur für die Daten selbst. In Kombination mit anderen Daten (Aufträge etc.) kann der Grad der Vertraulichkeit allerdings höher sein als bei einer isolierten Betrachtung. Dies wird mit der ITML dadurch abgebildet, dass wenn bspw. ein Datentyp *Auftrag* einen hohen Wert für *confidentiality* erhält, in der Konsequenz alle Datentypen, die mit *Auftrag* in Beziehung stehen, in diesem Kontext ebenfalls vertraulich zu behandeln sind.

face haben daher jeweils eine Beziehung (*uses* bzw. *is compatible with*) zu *CommunicationProtocol*. Ein Netzzugang muss ein Protokoll nutzen, das von dem Schnittstellentyp verwendet wird bzw. zu welchem der Schnittstellentyp kompatibel sein muss, der den Netzwerkzugang physisch realisiert. Ob dies zutrifft, kann mithilfe der hier vorgestellten Beziehungen überprüft werden.

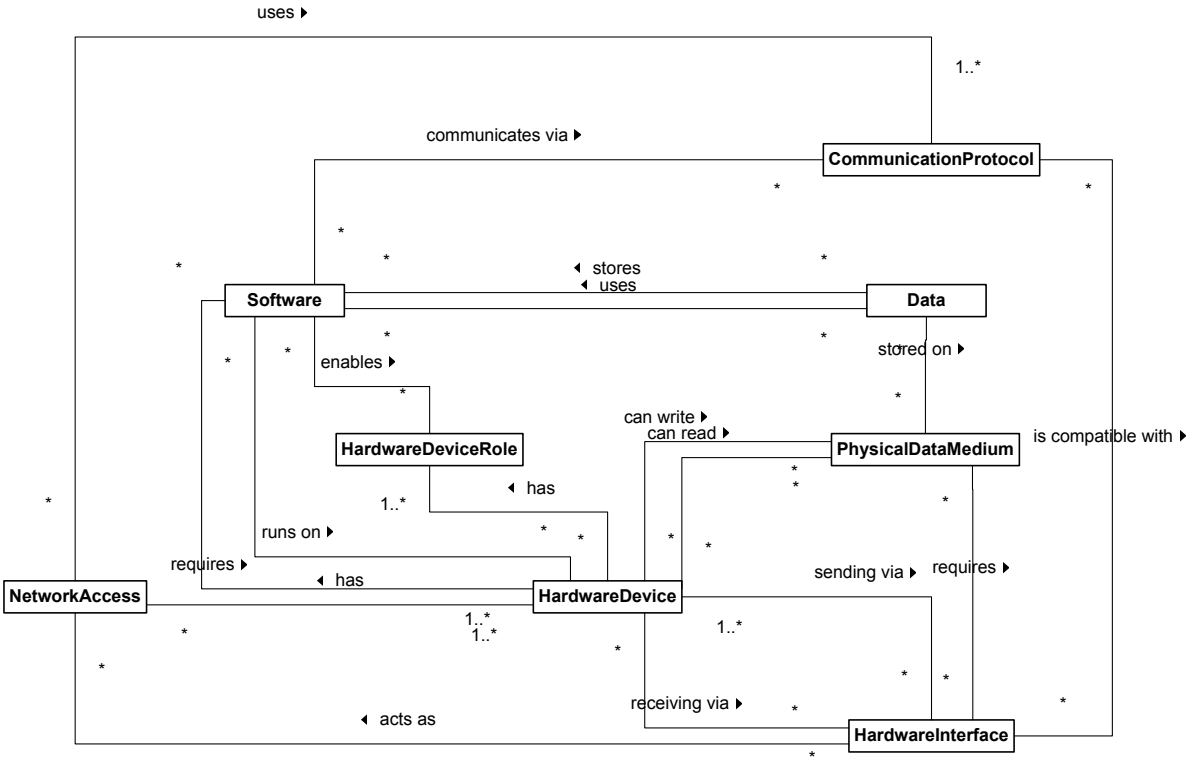

Abbildung 17: Hard- und softwarespezifische Konzepte

Standard

Eine ergänzende Abstraktion, welche sich sowohl auf Hard- als auch auf Software bezieht, ist der *Standard*. Standards existieren für nahezu alle Bereiche der IT in den verschiedensten Ausprägungen und sind somit auch für eine Vielzahl der bisher vorgestellten Konzepte von Belang. Standards existieren selbstverständlich auch außerhalb der IT (z.B. Prozesse, Vorgehen etc.), wovon im Rahmen dieser Arbeit allerdings abstrahiert wird. Abbildung 18 zeigt die Beziehungen von *Standard* zu den weiteren ITML-Konzepten auf.

Standard:

- *standardType*: Art des Standards (z.B. Spezifikation, Produkt)
- *open*: *true*, wenn es sich um einen offenen Standard handelt, *false* sonst
- *organisation*: Verweis auf die Organisationen, welche für den Standard verantwortlich sind
- *securityRating*: Angabe zur Einschätzung der relativen Sicherheit des Standards (z.B. hoch, mittel, niedrig)

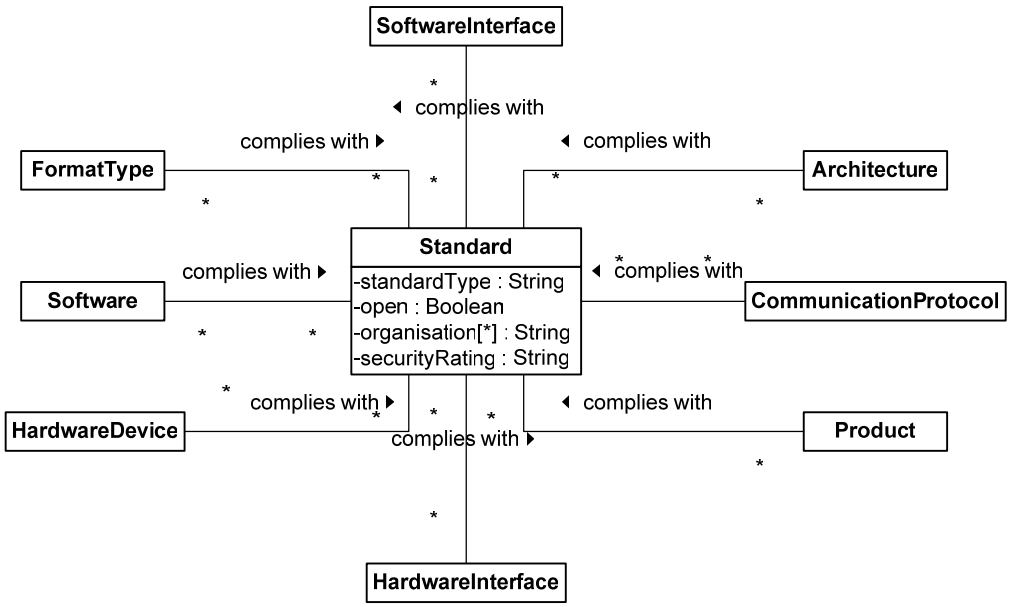

Abbildung 18: Standard und assoziierte Konzepte

Über jeweils eine Beziehung *complies with* stehen die folgenden Metatypen der ITML mit *Standard* in Beziehung:

- *Product*
- *FormatType*
- *Software*
- *HardwareDevice*
- *Architecture*
- *CommunicationProtocol*
- *SoftwareInterface*
- *HardwareInterface*

Durch die Beziehungen ist es in einem Modell bspw. möglich darzustellen, welche Komponenten einer IT-Landschaft standardkonform realisiert wurden. Diese Information kann in der Folge in entsprechende Analysen eingehen.

4.3.4 Organisation und Services

Zu den Aspekten der Organisation werden neben Aufbau- und Ablauforganisation auch Strategien und Services gezählt. Die Modellierung der Aufbau- und Ablauforganisation eines Unternehmens erfolgt in MEMO mit der dafür vorgesehenen Modellierungssprache OrgML (s. Abschnitt 4.2.2.2). An dieser Stelle wird daher lediglich auf die Beziehungen der dort spezifizierten Konzepte mit den oben beschriebenen ITML-Konzepten sowie auf einige Eigenschaften, welche speziell für die Zielsetzung der Arbeit von Interesse sind, eingegangen (s. Abbildung 19).

Strategy

Strategien sind üblicherweise nicht der Ablauf- und Aufbauorganisation zuzurechnen, sondern definieren eine eigenständige, spezifische Sicht auf ein Unternehmen. Da hier allerdings die Beziehung der Strategie zu Geschäftsprozessen und Organisationsrollen im Vordergrund stehen, werden sie in deren Kontext vorgestellt.

Strategy:

- *goal*: eine Menge an Zielen, die mit einer Strategie verknüpft sind bzw. durch diese erreicht werden sollen
- *term*: ein Zeitraum, innerhalb dessen eine Strategie gültig ist bzw. deren Ziele erfüllt werden müssen

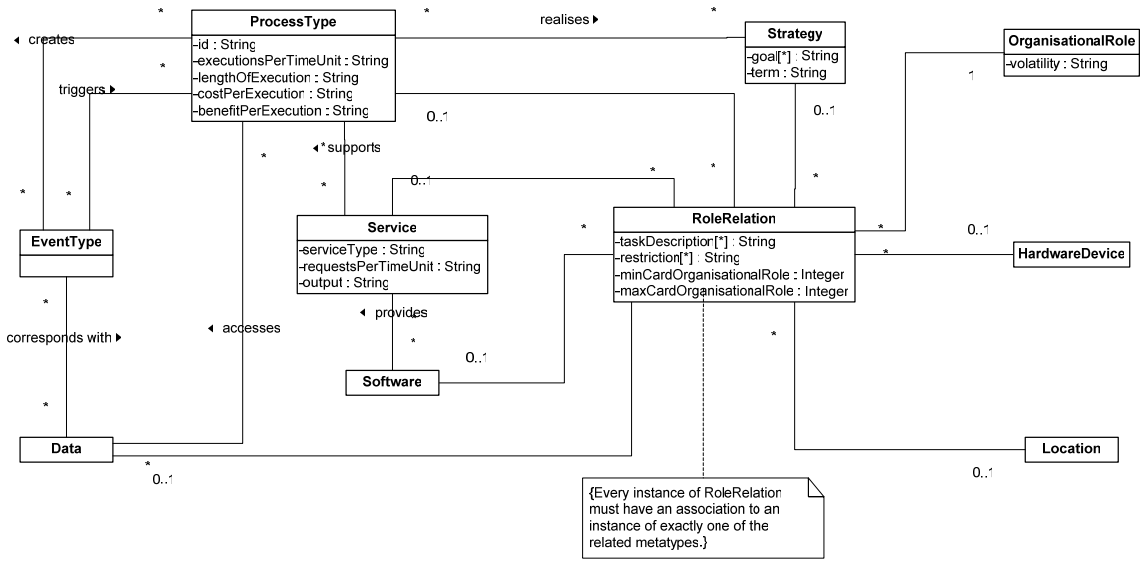

Abbildung 19: Integration von Aufbau- und Ablauforganisationskonzepten

ProcessType

Der Metatyp *ProcessType* beschreibt Typen von Geschäftsprozessen, durch die u.a. Strategien realisiert werden (Assoziation *realises* zwischen *ProcessType* und *Strategy*). Weiterhin kann ein Prozesstyp während der Ausführung einer Instanz auf Daten zugreifen, was sich in der Assoziation *accesses* zwischen *ProcessType* und *Data* ausdrückt.

ProcessType über eine Menge von Eigenschaften, von denen im Folgenden eine Teilmenge vorgestellt wird:

ProcessType:

- *id*: jeder Geschäftsprozesstyp erhält eine eindeutige ID, um ihn trotz Namensgleichheiten identifizieren zu können
- *executionsPerTimeUnit*: die geplante oder gemessene Zahl der Ausführungen eines Geschäftsprozesstypen pro Zeiteinheit
- *lengthOfExecution*: die geplante oder gemessene Dauer einer Ausführung einer Instanz des Geschäftsprozesstypen
- *costPerExecution*: die geplanten oder ermittelten Kosten einer Ausführung einer Instanz des Geschäftsprozesstypen
- *benefitPerExecution*: der geplante oder ermittelte Nutzen einer Ausführung einer Instanz des Geschäftsprozesstypen

Die kostenspezifischen bzw. nutzenspezifischen Attribute dienen der Dokumentation der Kosten und des Nutzens, welche von einzelnen Instanzen eines Geschäftsprozesstyps verursacht bzw. generiert werden. Dies unterscheidet sie von den Kosten- und Nutzenkonzepten, welche in Abschnitt 4.3.5 vorgestellt werden.

EventType

Ereignisse, wie z.B. das Eintreffen einer Nachricht oder der Zeitpunkt der Beendigung einer Ausführung der Geschäftsprozessinstanz, können von Geschäftsprozessen erzeugt werden und sie initiieren. Dies wird über die Assoziationen *creates* und *triggers* zwischen *ProcessType* und *EventType* abgebildet. Ein Ereignistyp steht über die Assoziation *corresponds with* mit *Data* in Beziehung. Somit wird dargestellt, dass mit Ereignissen verknüpfte Daten in einem System gespeichert und verwaltet werden können. Bspw. ist das Eintreffen eines Kundenauftrags ein Ereignis, welches logisch mit den betreffenden Kunden- und Auftragsdaten in Beziehung steht.

Service

Ein Service wird nicht nur durch IT realisiert, sondern auch durch Organisationsrollen (s. weiter unten im aktuellen Abschnitt). Im Sinne der Ausführungen in Abschnitt 2.4 verstehen wir unter Service eine Dienstleistung, welche von einem Anbieter an den Kunden geliefert wird. Durch das Konzept *Service* werden alle Arten von IT-Services erfasst, d.h. sowohl IT-basierte als auch anderweitige (z.B. Wartung).

Es stehen für die Definition von Services folgende Attribute zur Verfügung:

Service:

- *serviceType*: Art des Services (z.B. ressourcenorientiert, produktorientiert, Schnittstellenbeschreibung)
- *requestsPerTimeUnit*: Anzahl der geplanten oder ermittelten Nutzungen eines Services pro Zeiteinheit (z.B. 120 pro Sekunde)
- *output*: Beschreibung des Ergebnisses eines Services

Die Assoziation *provides* wird verwendet, um das Anbieten von Services durch Software darzustellen. Mit dem Attribut *output* wird lediglich das Serviceergebnis, nicht aber dessen Wirkung beschrieben. Diese wird durch eine Beziehung mit dem Nutzenkonzept (s. Abschnitt 4.3.5) abgebildet. Ein Service dient der Unterstützung eines oder mehrerer Geschäftsprozesse, was durch die Beziehung *supports* zwischen *ProcessType* und *Service* dargestellt wird.

Ein Service bildet oftmals eine Schnittstelle zwischen zwei Organisationen und fokussiert dabei weniger auf technische Aspekte, als auf die Vereinbarung und Beschreibung von hochwertigen Leistungen. Die technische Ebene der Kommunikation zwischen zwei IT-Systemen, welche auf Anbieterseite die Leistungen produzieren sowie auf Kundenseite nutzen, wird mit den entsprechenden Schnittstellenkonzepten auf Hard- und Softwareseite abgebildet. Somit ist es für die Planung der technischen Realisierung eines Services notwendig, die Konzepte der Hard- und Softwareebene auf Anbieter- und Kundenseite miteinander in Beziehung zu setzen. Dies erfolgt über die bereits in den Abschnitten 4.3.1 und 4.3.2 beschrieben Sprachkonzepte. Die Beschreibung dieser technischen Schnittstellen sollte in die Servicedokumentation (SLA) mit aufgenommen werden.

OrganisationalRole

OrganisationalRole ist Teil eines Rollenkonzepts, welches externe Organisationen, Unternehmensteilorganisationen oder Personentypen in einer Organisation abbildet.

OrganisationalRole:

- *volatility*: beschreibt qualitativ oder quantitativ die Volatilität der Rolle, d.h. wie oft sie normalerweise einem neuem Rollenträger zugeordnet wird (z.B. hoch, niedrig oder monatlich).

Bevor wir auf die Beziehungen von *OrganisationalRole* und in diesem Zusammenhang auf die Definition der Aufgaben einer Rolle eingehen, folgt eine Beschreibung der vorgesehenen Rollenträgertypen. Diese sind in Abbildung 20 dargestellt.

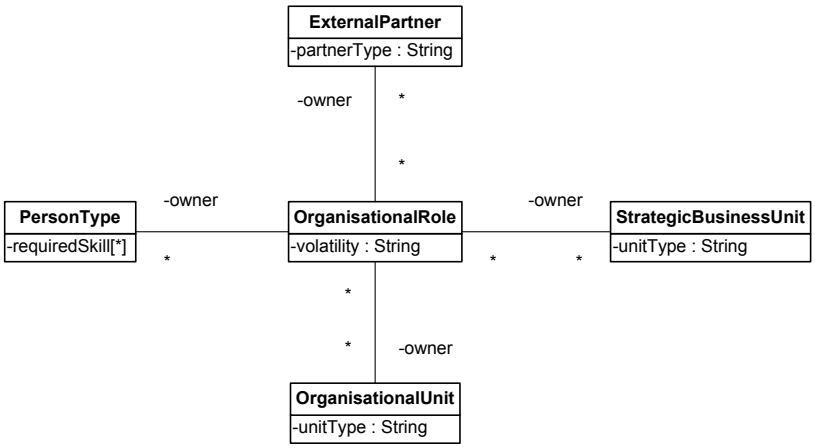

Abbildung 20: Rollenträger einer Organisationsrolle

PersonType

PersonType spezifiziert Typen von Personen, die eine unmittelbare Beziehung zum Unternehmen haben. In der Regel sind diese Personengruppen dem Unternehmen zugehörig, d.h. sie stehen zu ihm in einem Beschäftigungs- oder Teilhaberverhältnis.

PersonType:

- *requiredSkill*: Eine Liste der Fähigkeiten und Qualifikationen, die ein Mitglied der Personengruppe vorweisen muss

Oftmals erfolgt eine Unterscheidung in formale, messbare (z.B. Studienabschluss, mehrjährige Berufserfahrung, Fremdsprachen) und informale (*Soft Skills*, wie Kommunikationsfähigkeit, Teamfähigkeit) Qualifikationen. Dies soll hier aber nicht weiter ausgeführt werden. Stattdessen wird auf [Jung07] verwiesen, wo sich ein detaillierterer Vorschlag zur Modellierung von Qualifikationen findet.

OrganisationalUnit

Die Organisationseinheit (*OrganisationalUnit*) steht für eine identifizierbare Teilorganisation im Unternehmen, wie z.B. eine Abteilung oder ein Unternehmenszweig. Teile einer informalen Organisation sollen hiermit nicht abgebildet werden.

OrganisationalUnit:

- *unitType*: Art der Organisationseinheit

Das Attribut *unitType* wurde eingeführt, da der Name des Typs in der Terminologie eines Unternehmens (z.B. R&D für Forschungs- und Entwicklungsabteilungen) oftmals nicht auf die Art der Einheit schließen lässt. Dies gilt analog für die drei folgenden Konzepte.

StrategicBusinessUnit

Strategische Geschäftseinheiten (*StrategicBusinessUnit*) werden in der Regel um Produkte oder Märkte herum definiert und verfügen über eigene Strategien. Sie stellen Abstraktionen über andere (Teil-) Einheiten eines Unternehmens dar und sind in dieser Form als logische Ergänzung der Organisationsstruktur anzusehen.

StrategicBusinessUnit:

- *unitType*: Art der strategischen Geschäftseinheit

ExternalPartner

Unter externen Partnern (*ExternalPartner*) werden alle betriebsfremden Parteien zusammengefasst, die bei der Betrachtung der zu modellierenden Domäne von Interesse sind. Dies bezieht sich zunächst auf Kunden und Lieferanten, kann aber auch zur Darstellung von Konkurrenten oder Gutachtern (z.B. TÜV-Mitarbeiter in Verbindung mit der periodisch durchgeführten ISO 9000-Prüfung) verwendet werden. Die Differenzierung zu *PersonType* ist mitunter problematisch, da Personengruppen wie freie Mitarbeiter durchaus Eigenschaften beider Konzepte aufweisen. Hier bleibt es letztlich dem Modellierer überlassen, welche Abstraktion im Kontext als passend empfunden wird.

ExternalPartner:

- *partnerType*: Art des externen Partners

RoleRelation

Die möglichen Organisationsrollen stehen mit weiteren ITML-Konzepten in Beziehung. Diese Beziehungen sind durch den Metatyp *RoleRelation* gekapselt.

RoleRelation:

- *taskDescription*: Beschreibung der Aufgaben, die mit der Beziehung einhergehen
- *restriction*: Menge der Einschränkungen, die mit der Beziehung einhergehen
- *minCardOrganisationalRole*: minimale Anzahl der Beteiligungen an der Rollenbeziehung durch Instanzen des angebundenen Organisationsrollentyps
- *maxCardOrganisationalRole*: maximale Anzahl der Beteiligungen an der Rollenbeziehung durch Instanzen des angebundenen Organisationsrollentyps

Es gilt die Einschränkung, dass *maxCardOrganisationalRole* >= *minCardOrganisationalRole* >= 0 ist.

Dieser Metatyp erlaubt es dem Modellierer eine Menge von Beziehungen zu definieren, die festlegen, welche Aufgaben und Randbedingungen einer Rolle in einem bestimmten Kontext zugeordnet werden. So muss im Falle von Änderungen die Rollendefinition nicht angepasst werden. Dies entspricht dem in der Realität üblichen Vorgehen, eine Rolle, wie z.B. Administrator, festzulegen und im Zeitverlauf dessen Verantwortlichkeiten und Berechtigungen in Bezug auf die IT-Landschaftskomponenten anzupassen. Diese Anpassung geschieht im Kontext mit den verbundenen Aufgabenbereichen, welche durch die angebundenen ITML-Konzepte repräsentiert werden.

Eine Rollenbeziehung ist binär. Es wird pro Beziehung jeweils ein Organisationstyp mit genau einem Typ verbunden, der durch einen der Metatypen Strategie, Geschäftsprozess, Service, Hardwaregerät, Software, Daten oder Ort beschrieben wird. Somit ergeben sich die folgenden gültigen Rollenbeziehungen:

OrganisationalRole -> Strategy

OrganisationalRole -> ProcessType

OrganisationalRole -> Service

OrganisationalRole -> Software

OrganisationalRole -> Hardware

OrganisationalRole -> Data

OrganisationalRole -> Location

Die Attribute *maxCardOrganisationalRole* und *minCardOrganisationalRole* ermöglichen eine Annotation von Kardinalitäten an dem Beziehungsende der Organisationsrolle. Somit können Integritätsbedingungen formuliert werden, die es bspw. ermöglichen, die gleichzeitige Anzahl von Nutzern einer Software zu beschränken. Auf Kardinalitäten auf Seiten der Konzepte, an welche die Organisationsrollen angebunden sind, wurde in der ITML verzichtet. Im Falle von Strategien, Geschäftsprozessen, Services und Daten erscheint eine Beschränkung auf eine Teilmenge von Instanzen ohnehin nicht sinnvoll. Hier wird eine Rolle normalerweise an alle möglichen Instanzen gleichzeitig gebunden. Bei Hardware, Software und Orten wäre eine Beschränkung denkbar, aber auf dem zu erwartenden Abstraktionsniveau eines ITML-Modells wenig hilfreich. So könnte zwar eine Beschränkung der Wartungsbeziehung eines Administrators zu Hard- oder Software (d.h. z.B. dass ein Administrator nur max. 10 Server eines Typs oder 10 Datenbankinstallationen eines Typs warten kann) eingeführt werden, allerdings würde dies einer eher realitätsfremden Restriktion entsprechen. Ähnliches gilt für Ortstypen.

Im Folgenden werden exemplarisch Ausprägungen von *RoleRelation* aufgelistet, um die Anwendung des Konzepts zu verdeutlichen. Je Beispiel werden zunächst Attributbelegungen von *taskDescription* und *restriction* aufgeführt. Anschließend werden die Kardinalitäten und die angebundenen Typen in folgender Form beschrieben:

RoleRelation->minCardOrganisationalRole..RoleRelation->maxCardOrganisationalRole OrganisationalRole->name <-> attachedType *->name*

attachedType entspricht dabei einer Ausprägung eines der Metatypen *Strategy*, *ProcessType*, *Service*, *Software*, *Hardware*, *Location* und *Data*.

RoleRelation -> Strategy: Strategy Definition

- *taskDescription*: attached role defines strategy
- *restriction*: attached role may not change strategy without consulting (some *OrganisationalRole*)
- *1..1 CEO <-> Concrete Strategy*

RoleRelation -> ProcessType: Process Execution

- *taskDescription*: attached role executes business process
- *restriction*: attached role may not change business process
- *1..1 Process Owner <-> Sales*

RoleRelation -> Service: Service Definition

- *taskDescription*: attached role defines service
- *restriction*: attached role may not remove service
- *1..2 SLA Manager <-> Ticket*

RoleRelation -> Software: Software Updating

- *taskDescription*: attached role is responsible for applying regular software updates
- *restriction*: attached role may not upgrade software without consulting (some *OrganisationalRole*)
- *1..5 Database Administrator <-> Database*

RoleRelation -> Hardware: Hardware Maintenance

- *taskDescription*: attached role is responsible for maintaining hardware component

- *restriction*: attached role may not replace components without consulting (some *OrganisationalRole*)
- *1..2 Network Administrator <-> Router*

RoleRelation -> Location: Location Guarding

- *taskDescription*: attached role is responsible for the security of the location
- *restriction*: attached role is not allowed to access the location
- *2..3 Security Personnel <-> Server Farm*

RoleRelation -> Data: Data Access

- *taskDescription*: attached role is authorised to access data
- *restriction*: attached role is not allowed to modify data
- *1..* Customer Service <-> Customer Data*

Unter Verwendung der Rollenbeziehung zu betrieblichen Daten können Rahmenbedingungen für deren Zugriff und Modifikation als Teil eines unternehmensweiten Sicherheitskonzepts definiert werden.

4.3.5 Kosten- und Nutzenspezifische Konzepte

In diesem Abschnitt wird zunächst das Nutzenkonzept gefolgt von Kostenkonzepten vorgestellt. Der betreffende Metamodellausschnitt ist in Abbildung 21 abgebildet.

Benefit

Eine ausführliche Diskussion der möglichen Wendungen des Begriffs Nutzen findet sich in Abschnitt 2.6. Eine detaillierte Abbildung der dort aufgezeigten unterschiedlichen Sichtweisen auf Metamodellebene erscheint wenig zielführend. Vor diesem Hintergrund wurde *Benefit* in der ITML als ein relativ generisches Konzept entworfen. *Benefit* ist mit *Service* über eine Assoziation *generates* verbunden. Mit dieser Beziehung kann der Nutzeneffekt, den ein Service durch sein Ergebnis erzeugt, beschrieben werden. Gleichermaßen existiert eine Beziehung *generates* zu *ProcessType*, womit unabhängig von Services der Nutzen eines Geschäftsprozesstyps dargestellt werden kann. Dies ist vor allem aus Kundensicht sinnvoll, wenn die durch den Bezug von Services entstandenen Kosten einem Nutzen gegenübergestellt werden sollen.

Benefit:

- *tangible*: *true*, wenn es sich um einen tangiblen Nutzen handelt (z.B. ein Geldbetrag), *false* sonst
- *expression*: qualitative (z.B. hoch, gering) oder quantitative (z.B. 12000 Euro) Beschreibung des Nutzeneffekts

Aufbauend auf den in Abschnitt 2.5 identifizierten zentralen Begriffen der IT-Kostenrechnung werden nun die Sprachkonzepte zur Modellierung von Kosten vorgestellt[1]. Als primäres Entwurfsziel der Sprachkonzepte steht an dieser Stelle nicht die Schaffung eines Werkzeugs für eine detaillierte betriebliche Kostenrechnung. Vielmehr sollen die Zusammenhänge zwischen den Kosten einerseits und den Kernkonzepten der ITML andererseits verdeutlicht sowie die Grundlage für Planungs- und Analysetätigkeiten auf einem höheren Abstraktionsniveau geschaffen werden. Hinter den Ausprägungen

[1] In [Kirc05] wird eine frühere Version des Metamodells diskutiert und anhand von Beispielen veranschaulicht.

der hier vorgestellten Kostenkonzepte verbergen sich daher aggregierte Ergebnisse aus der Kostenrechnung bzw. zusammengefasste Planungsvorgaben.

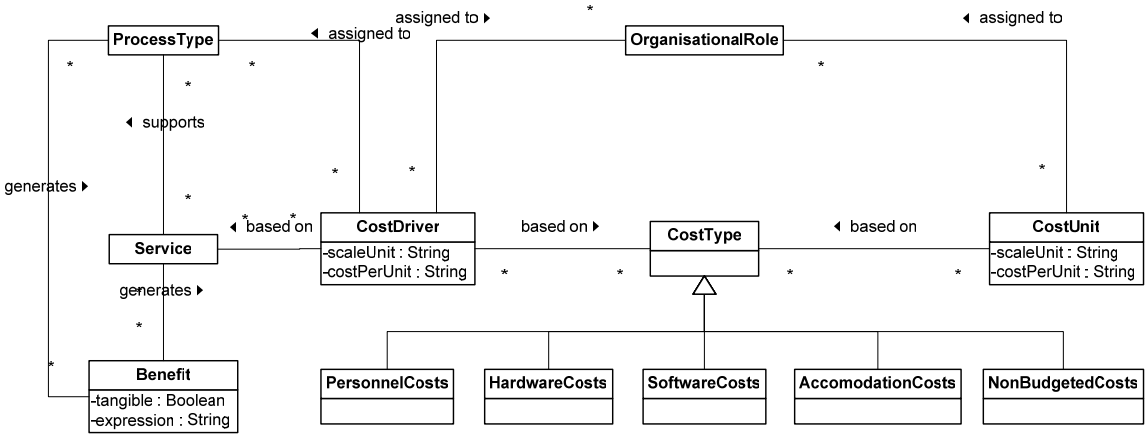

Abbildung 21: IT-spezifische Kostenkonzepte

CostUnit

Ein Kostenträger dient innerhalb einer Kostenträgerechnung als Verrechnungsbasis für anfallende Kosten, welche einer Kostenstelle in Rechnung gestellt werden können. Das entsprechende Konzept *CostUnit* verfügt über die Beziehung *assigned to* zu *OrganisationalRole*, deren Ausprägungen als Kostenstelle fungieren. Die Ermittlung der Kosten, die einem Kostenträger zugerechnet werden, erfolgt auf Basis der tatsächlich anfallenden oder geplanter Kosten. Es besteht eine Beziehung zwischen Kostenträgern und Kostenarten (Assoziation *based on* zwischen *CostUnit* und *CostType*).

CostUnit:

- *scaleUnit*: Maßeinheit eines Kostenträgers (z.B. CPU-Zeit in Millisekunden, Papier in Anzahl der Blätter)
- *costPerUnit*: geplante oder ermittelte Kosten pro Kostenträgereinheit

CostDriver

Analog zu Kostenträgern, allerdings im Kontext einer betrieblichen Prozesskostenrechnung, dienen Kostentreiber als Verrechnungsbasis für IT-Kosten. Somit existiert auch hier ein Bezug zu den Kostenarten (*based on*). Zusätzlich zur Zuordnung zu einer Kostenstelle können hier noch Services und Geschäftsprozesse angebunden werden. Ein Kostentreiber wird in Relation zu einem Geschäftsprozess definiert, welcher durch wiederholte Ausführung Kosten verursacht (im Metamodell *CostDriver assigned to ProcessType*). Die Kosten können zusätzlich über die Anzahl der Inanspruchnahme bestimmter IT-Services berechnet werden. Dies manifestiert sich im Metamodell durch die Assoziation *based on* zwischen *CostDriver* und *Service*.

CostDriver:

- *scaleUnit*: Maßeinheit eines Kostentreibers (z.B. CPU-Zeit in Millisekunden, Anzahl Nutzungen eines Services)
- *costPerUnit*: geplante oder ermittelte Kosten pro Kostentreibereinheit

CostType

Das Konzept *CostType* steht stellvertretend für mögliche Kostenarten, die im Zusammenhang mit IT auftreten können. Es kann weiterführend zwischen den folgenden speziellen Kostenarten differenziert werden:

CostType -> Personnel Costs

Personalkosten sind Aufwendungen für Personal, welche für Tätigkeiten im Kontext der betrieblichen IT geleistet werden.

Bsp.: Gehälter für Angestellte in der IT-Organisation

CostType -> HardwareCosts

Hardwarekosten sind Aufwendungen, welche in ummittelbarem Zusammenhang mit Hardwarekomponenten stehen.

Bsp.: Anschaffungskosten, Supportkosten, Upgradekosten

CostType -> SoftwareCosts

Softwarekosten sind Aufwendungen, welche in ummittelbarem Zusammenhang mit Software oder Softwareprodukten stehen.

Bsp:. Anschaffungskosten, Supportkosten, Entwicklungskosten

CostType -> AccomodationCosts

Unterbringungskosten sind Aufwendungen, welche in Verbindung mit Räumlichkeiten stehen.

Bsp.: Mietkosten für Betriebsräume, Versicherungen, Strom

CostType -> NonBudgetetCosts

Nicht-budgetierte Kosten sind alle Kosten, die oftmals im Einzelnen nicht erfasst werden.

Bsp.: Verzögerungen durch ergonomische Mängel oder Systemausfälle

Die speziellen Kostenarten unterscheiden sich in den Beziehungen, die für sie zu anderen Typen zugelassen sind. Dieser Sachverhalt geht aus Abbildung 22 hervor. Die vorgesehenen Zuordnungsbeziehungen *attributed to* sind:

- *PersonnelCosts -> OrganisationalRole*
- *HardwareCosts -> HardwareDevice*
- *SoftwareCosts -> Software, SoftwareProduct*
- *AccomodationCosts -> Location*
- *NonBudgetedCosts -> Software, HardwareDevice*

Personalkosten werden Organisationsrollen zugeordnet und somit dem jeweiligen Rollenträger. Hardwarekosten beziehen sich auf beliebige Hardwarekomponenten. Softwarekosten entstehen sowohl bei Softwareprodukten als auch bei Softwareartefakten. Die Unterbringungskosten werden einem Ort zugerechnet sowie die nicht-budgetierten Kosten sowohl Hardware als auch Software.

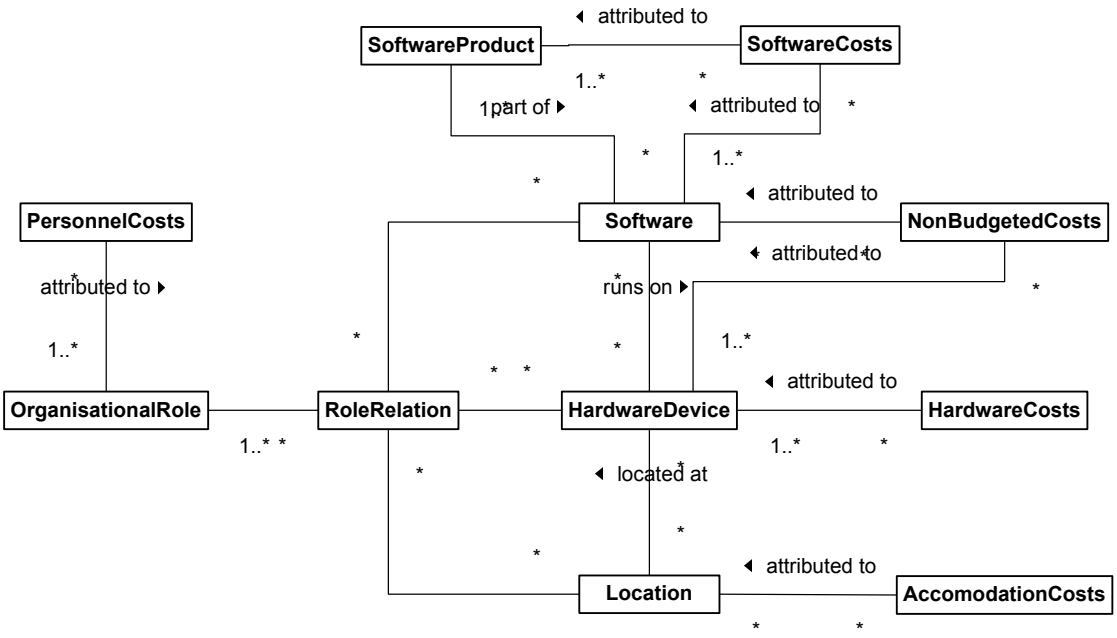

Abbildung 22: Zuordnung der speziellen Kostenarten

Eine weitere Detaillierung von Kostenaspekten auf der Ebene der Sprachdefinition erscheint nicht sinnvoll, sondern sollte auf den darunter liegenden Abstraktionsebenen erfolgen. Dazu zählt unter anderem die bedarfsgerechte Definition unternehmensspezifischer Ausprägungen von Kostentreibern, Kostenarten und Kostenträgern, die in der betrieblichen Kostenrechnung verwendet werden.

4.3.6 Informationssysteme

Ein Informationssystem ist eine höhere Abstraktion über betriebliche IT und fasst eine Reihe anderer Konzepte, wie insbesondere Soft- und Hardware logisch zusammen (vgl. Abschnitt 2.1). Wenn eine detaillierte Sicht auf diese Informationssystemkomponenten nicht gewünscht oder notwendig ist, dann bietet sich die Informationssystem-Metapher als die zentrale Abstraktion eines Modells einer IT-Landschaft an. Gerade hinsichtlich der strategischen IT-Planung steht an erster Stelle die Analyse bzw. Planung der Beziehungen von Informationssystemen mit anderen zentralen Konzepten, wie Geschäftsprozessen oder Strategien. Zu diesem Zweck ist die Service-Metapher nicht immer zur Darstellung der Geschäftsprozessunterstützung seitens der IT geeignet oder notwendig (s.u.).

Weiterhin stellt sich auf dieser Betrachtungsebene die Frage nach Qualität, Zukunftssicherheit etc. im Zusammenhang mit einer strategischen Perspektive auf ein Unternehmen. Daher ergibt sich die Notwendigkeit, die bisher vorgestellten Modellierungskonzepte um weitere zur Darstellung dieser Sachverhalte konzipierte Sprachkonstrukte zu ergänzen.

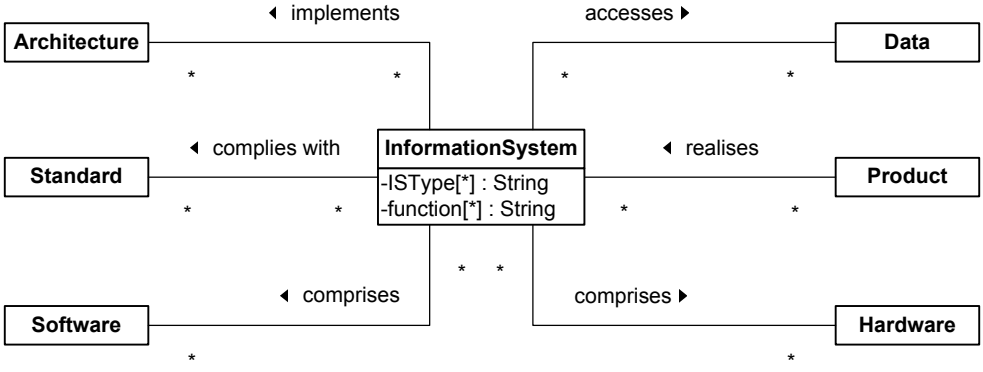

Abbildung 23: Informationssystem und technische Konzepte

InformationSystem

Der Metatyp *InformationSystem* wird hauptsächlich über seine Beziehungen mit anderen Konzepten charakterisiert. Er verfügt demnach über wenige spezielle Attribute.

InformationSystem:

- *ISType*: Typ des Informationssystem (z.B. Decision Support System, CAD-System, Warenwirtschaftssystem)
- *function*: betriebswirtschaftliche Funktionen des Systems (z.B. Vertrieb, Beschaffung, Marketing)

Die Attribute sind jeweils mehrwertig, da oftmals eine eindeutige Zuordnung eines Informationssystems zu einem Typ oder einer Funktion nicht möglich ist. So kann ein ERP mehrere Typen von Informationssystemen zusammenfassen, was u.a. auch darin resultiert, dass mehrere betriebliche Funktionen abgedeckt werden. Die Metapher der Funktion ist im unternehmerischen Sprachgebrauch trotz der zunehmenden Prozessorientierung immer noch gegenwärtig und daher als optional anzugebende Information über ein Informationssystem vorgesehen. Die Modellierung der Funktionsrealisierung erfolgt allerdings mithilfe von Geschäftsprozessen.

InformationSystem hat eine Reihe von Beziehungen zu zuvor eingeführten technischen Konzepten (s. Abbildung 23). Somit kann abgebildet werden, auf welcher technischen Grundlage ein Informationssystem aufbaut. Hier ist zuerst die Architektur zu nennen, die implementiert wird (*InformationSystem implements Architecture*). Weiterhin basieren Informationssysteme auf einer Menge von Hard- und Softwarekomponenten (Assoziationen *comprises* zu *Hardware* und *Software*). Sie können durch ein oder mehrere Produkte realisiert werden (*Product realises InformationSystem*) und im Ganzen standardkonform sein (*InformationSystem complies with Standard*). Hier wird üblicherweise auf Produktstandards wie SAP R/3 verwiesen. Schließlich greifen Informationssysteme auf Daten zu, um ihre Aufgaben erfüllen zu können (Assoziation *accesses* zu *Data*).

Um aber eine strategische oder organisatorische Sicht auf Informationssysteme zu realisieren, werden außer den technischen noch weitere Konzepte benötigt. Diese sind in Abbildung 24 in Verbindung mit *InformationSystem* abgebildet.

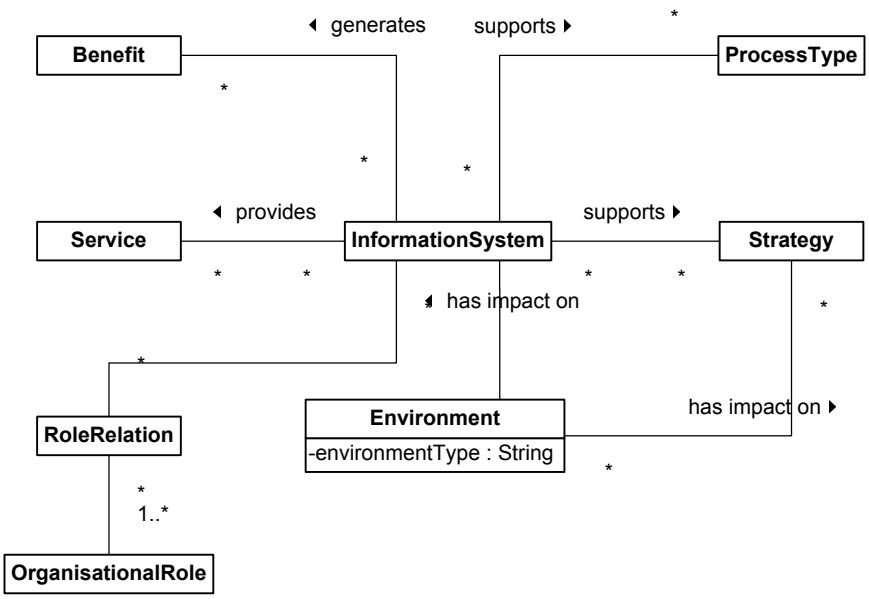

Abbildung 24: Informationssystem und strategie- bzw. organisationsrelevante Konzepte

Ein Informationssystem bietet Services an (*InformationSystem provides Services*) und unterstützt sowohl Geschäftsprozesse (*InformationSystem supports ProcessType*) als auch die Verfolgung von Strategien (*InformationSystem supports Strategy*). An dieser Stelle verbergen sich eine Reihe intendierter Redundanzen. So verfügt der Metatyp *Software* bereits über eine Assoziation zu *Service*. Damit ist es möglich, über die Beziehungen zwischen Softwaretypen und Services sowie Softwaretypen und Informationssystemen potenzielle Verbindungen zwischen Informationssystemen und Services abzuleiten. Dies ist aber aus zwei Gründen problematisch. Erstens sind nicht alle Services, die von einem Softwaretypen angeboten werden, relevant für die Sicht auf ein Informationssystem. Auf dieser Ebene werden typischerweise nur Dienstleistungen aufgezeigt, die nach außen hin angeboten und berechnet werden. Zweitens ist dem Sprachanwender im Rahmen einer Analyse zur Ermittlung des strategischen Potenzials eines Informationssystems oder der Planung von Einsatzszenarien nicht zuzumuten, Services nur in Verbindung mit den realisierenden Softwarekomponenten angeben zu können.

Eine weitere Redundanz liegt in der Verbindung des Informationssystems zu den Geschäftsprozessen und den Strategien. Hier gilt eine ähnliche Argumentation wie im Falle der Services. In der Analyse und Planung von Informationssystemen ist es notwendig, direkt Zusammenhänge zu diesen zentralen Konzepten aufzuzeigen, ohne auf Mittlerkonzepte zurückgreifen zu müssen. Zu beachten ist dabei, dass bei einer Verfeinerung des Modells um ebendiese Konzepte keine Inkonsistenzen entstehen. So dürfen z.B. seitens eines Informationssystems keine Services angeboten werden, die nicht an anderer Stelle von einer zugeordneten Softwarekomponente implementiert sind. Weiterhin müssen so angebundene Services mindestens einen der vom Informationssystem unterstützen Geschäftsprozesstypen ebenfalls unterstützen, sowie Geschäftsprozesse die so verbundenen Strategien realisieren. Diese Einschränkungen können genutzt werden, um die Planung eines Informationssystems mit der zugrunde liegenden technischen Realisierung abzugleichen.

InformationSystem verfügt weiterhin über eine Beziehung *generates* zu *Benefit*. Hier ergibt sich eine potenzielle Redundanz dahingehend, dass Services, welche von Informationssystemen angeboten werden, ebenfalls mit dem Nutzenkonzept in Beziehung stehen können. Die Art des Nutzens, welcher im Zusammenhang mit einem Informationssystem von Interesse ist, unterscheidet sich aber in der Regel von der eines Servicenutzens. Die unterschiedlichen Dimensionen, die sich hier über den Begriff aufspannen, decken bzgl. Informationssystemen eine größere Bandbreite ab, die sich von monetären (tangiblen) bis hin zu kulturellen (intangiblen) Aspekten erstreckt. Der Servicenutzen ist in der Regel konkreter, da er oftmals direkt in Wirtschaftlichkeitsbetrachtungen sowie die Preisgestaltung von Services mit eingeht. Bei der Erzeugung von Nutzentypen auf Modellebene obliegt es in der Hauptsache dem Anwender, das jeweils angemessene Abstraktionsniveau zu finden.

Die Beziehungen zu Organisationsrollen sind nicht redundant, da hier eher die Gesamtverantwortung für komplette Systeme als bspw. die Wartung von Subsystemen abgebildet wird. Dies verdeutlicht folgendes Beispiel eines Rollenbeziehungstyps:

RoleRelation -> InformationSystem: IS Overall Responsibility

- *taskDescription*: attached role has the overall responsibility for all aspects regarding the information system
- *restriction*: none

Rollentyp z.B. *CIO*

Der Metatyp *InformationSystem* verfügt noch über eine weitere Beziehung zu dem bisher noch nicht vorgestellten Konzept *Environment*.

Environment

Der Metatyp *Environment* beschreibt diverse Arten von Umwelteinflüssen auf das Unternehmen. So können Gesetze bestimmte Einschränkungen oder zusätzlich benötigte Funktionalitäten für Informati-

onssysteme implizieren. Dies trifft analog für kulturelle oder marktspezifische Faktoren einer Region zu. Diese Einflüsse werden durch die Assoziation *has impact on* zwischen *Environment* und *InformationSystem* abgebildet. Eine weitere Beziehung existiert zu *Strategy* (*has impact on*). So können Umwelteinflüsse auf Strategien ohne Berücksichtigung von Informationssystemen betrachtet werden. Einflussfaktoren, welche die Strategieformulierung beeinflussen, müssen nicht zwingend auf die Gestaltung des Informationssystems einwirken. So ist die Ausrichtung auf bestimmte Produkte in einer Region, bedingt durch die dort vorherrschenden Präferenzen, für ein ERP-System nicht von Belang, da hierfür nicht zwingend Modifikationen vorgenommen werden müssen.

Environment:

- *environmentType*: Art des Umweltfaktors (z.B. Markt, Gesetz, Kultur)

Ein weiterer zentraler Aspekt bei der Betrachtung von Informationssystemen ist das Management von Risiken, d.h. das Antizipieren von potenziellen Stör- und Zwischenfällen sowie die Planung und Implementierung pro- und reaktiver Schutz- und Gegenmaßnahmen. Abbildung 25 zeigt die von dieser Sicht betroffenen Modellierungskonzepte der ITML.

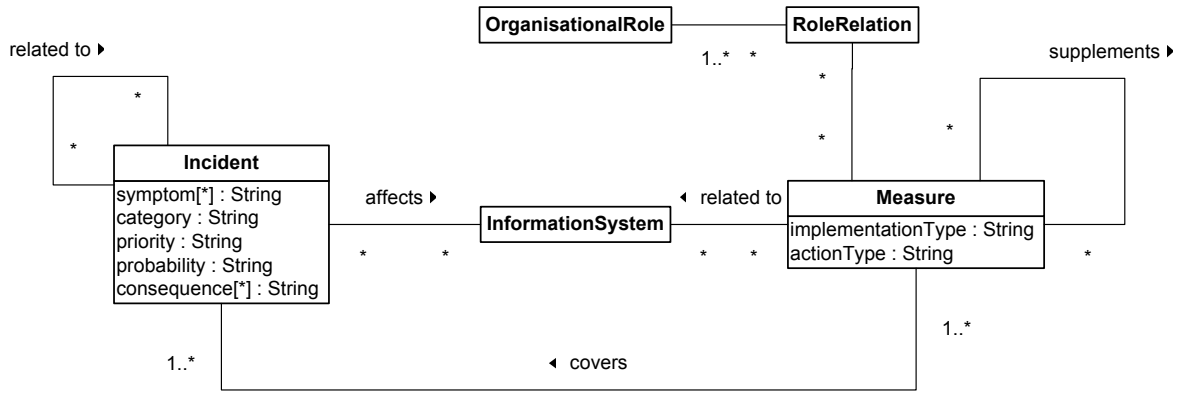

Abbildung 25: Risikokontext von Informationssystemen

Incident

Ein Zwischenfall (*Incident*) steht stellvertretend für jede denkbare Störung, die auf ein Informationssystem einwirken und den planmäßigen Betrieb beinträchtigen kann. Zur Beschreibung eines Zwischenfalltyps stehen die folgenden Attribute zur Verfügung:

Incident:

- *symptom*: eine Beschreibung der möglichen Symptome, anhand derer das Eintreten eines Zwischenfalls erkannt werden kann (z.B. höhere Bearbeitungszeiten, Totalausfall)
- *category*: Kategorie des Zwischenfalls (z.B. Manipulation, technischer Ausfall)
- *priority*: Einschätzung der Wichtigkeit eines Zwischenfalls (z.B. hoch, 3/10)
- *probability*: Wahrscheinlichkeit des Eintritts eines Zwischenfalls (z.B. 0,001%)
- *consequence*: Liste der möglichen Konsequenzen, die ein Zwischenfall nach sich ziehen kann (z.B. Ausfall eines Vertriebskanals, Ausfall der Produktionsanlagen)

Zwischenfälle wirken auf Informationssysteme ein (Assoziation *affects* zu *InformationSystem*) und können miteinander in Beziehung stehen (*relates to*). Somit können Kausalitäten oder Korrelationen zwischen Zwischenfällen abgebildet werden.

Das geschätzte Risiko, welches bspw. als Produkt aus Eintrittswahrscheinlichkeit eines Zwischenfalls und (quantifizierter) Konsequenz (z.B. Einnahmeverlust) interpretiert werden kann, dient als Anhaltspunkt zur Festlegung des Werts für *priority*.

Measure

Eine Maßnahme (*Measure*) dient der Verhinderung eines Zwischenfalls bzw. der Behandlung der Konsequenzen nach dessen Eintreten.

Measure:

- *implementationType*: Art der Ausführung (z.B. automatisch/teilautomatisch/manuell)
- *actionType*: Art der Aktivität einer Maßnahme (z.B. proaktiv/reaktiv)

Maßnahmen können verschiedene Grade der Automatisierung vorweisen und proaktiv oder reaktiv sein. Bspw. ist die Verwendung eines Backup-Servers eine proaktive und automatische oder teilautomatische Maßnahme, da die Konsequenzen von Zwischenfällen damit größtenteils vorbeugend kompensiert werden können. Backup-Server können Sicherungen zeitgesteuert automatisch oder durch manuelles Anstoßen durchführen.

Das ortsnahe Vorhalten von Ersatzteilen zum Austausch fehlerhafter Komponenten ist im Gegensatz zum ersten Beispiel eine reaktive, manuelle Maßnahme. Die Konsequenz eines Zwischenfalls, wie bspw. ein temporärer Systemausfall, kann nicht gänzlich kompensiert werden, allerdings wird die Reaktions- und Fehlerbehebungszeit deutlich verkürzt. Die Durchführung des Austauschs erfolgt händisch durch einen Mitarbeiter.

Für jede Maßnahme existiert eine zuständige Organisationsrolle (Beziehung zu *OrganisationalRole*). Dies kann z.B. folgendermaßen ausgeprägt sein:

RoleRelation -> Measure: Measure Execution

- *task*: attached role is responsible for the execution of the measure
- *restriction*: attached role is not allowed to modify the measure specification

Rollentyp z.B. *Administrator*

Jede Maßnahme deckt einen oder mehrere mögliche Zwischenfälle ab, wobei ein dokumentierter Zwischenfall in mindestens einer Maßnahme berücksichtigt sein muss (*Measure covers Incident*).

Die hier genannten Konzepte unterstützen ein betriebliches Risikomanagement dahingehend, dass eine initiale Beschreibungsstruktur für Zwischenfälle und Maßnahmen vorgegeben wird. Somit wird ein größeres Bewusstsein für Bedrohungen der Geschäftsfähigkeit durch Ausfall oder Beeinträchtigung der zum Betrieb nötigen IT gefördert und die Planung möglicher Schutzmechanismen unterstützt. Diesbezügliche Zuständigkeiten können festgelegt und betroffene Informationssysteme identifiziert werden.

4.4 Konzepte zur Wiederverwendung

Die ITML unterstützt sowohl Typbildung/Kapselung als auch Generalisierung/Spezialisierung/Vererbung als Wiederverwendungskonzepte (s. auch Abschnitt 3.2). Durch Typbildung werden die gemeinsamen Eigenschaften von Instanzen zusammengefasst und in einem Typen gekapselt. Dies wird bei jeder Verwendung der Sprache implizit genutzt und muss hier nicht weiter ausgeführt werden.

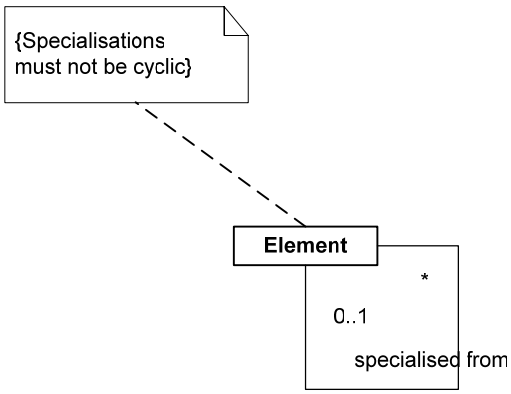

Abbildung 26: Spezialisierung im Metamodell der ITML

Die Spezialisierung hingegen bedarf einiger Erläuterungen. Sie basiert auf der Erzeugung von generellen Typen und erlaubt auf deren Basis die Definition von weiteren, speziellen Typen. Die Spezialisierung ist für alle Subtypen des Metatyps Element zulässig (s. Abbildung 26). Zyklische Vererbungshierarchien sind ebenso wie Mehrfachvererbung nicht erlaubt. Die Probleme, welche letzteres birgt - hier ist vor allem an Vererbung widersprüchlicher Definitionen von Typeigenschaften zu denken – sprechen gegen eine Umsetzung in der ITML.

Folgendes Beispiel illustriert eine mögliche Spezialisierungsbeziehung zweier ITML-Typen. So sei der Ortstyp *Office* eine Instanz des Metatyps *Location* und hinsichtlich der Unterbringung von Hardwaregeräten definiert. Ein benutzerdefiniertes Name/Werte-Paar ist *maxEmployees = 4*. Hiermit soll ausgedrückt werden, dass in einem Büro maximal vier Angestellte untergebracht werden können. Nun kann dieser Ortstyp in *ExecutiveOffice* spezialisiert werden, welcher einige zusätzliche Einschränkungen bzw. Eigenschaften aufweist. So erhält *maxEmployees* den Wert *1*, was eine Verschärfung des generellen Werts *4* bedeutet. Ein zusätzliches benutzerdefiniertes Attribut *minDimension = 5x5 m* beschreibt Mindestmaße für ein repräsentatives Büro. Von der objektorientierten Interpretation der Spezialisierung ausgehend müsste das Prinzip der Ersetzbarkeit gelten[1]. D.h. überall dort wo der der generelle Typ *Office* in einem Modell verwendet wird, müsste auch der spezielle Typ *ExecutiveOffice* Verwendung finden können. Durch die intendierte Semantik der beiden Typen würde dies allerdings zu logischen Inkonsistenzen in einem Modell führen und ist daher nicht vorzusehen. Ein Chefbüro erfüllt zwar formal die Restriktionen eines Büros, findet in der Realität allerdings kaum Verwendung als Mitarbeiterbüro. Nicht zuletzt die Einschränkung auf einen (im konkreten Falle hochrangigen) Mitarbeiter verhindert dies.

Als Konsequenz der Problematik, die durch das o.g. Beispiel verdeutlicht wird, leitet sich die Entwurfsentscheidung ab, Ersetzbarkeit nicht als Bestandteil des Spezialisierungskonzepts der ITML aufzunehmen. Stattdessen wird Spezialisierung auf die Wiederverwendung von Typeigenschaften reduziert, was im Grunde dem Prinzip der Vererbung entspricht. Vererbt werden dürfen die ITML-Konzepte

- Metaattribut,
- Attribut,
- Name/Werte-Paar,
- Einschränkung und
- Beziehung.

[1] Das Prinzip der Ersetzbarkeit beschreibt den Sachverhalt, dass jede Instanz eines speziellen Typs, der durch eine *is-a*-Beziehung (d.h. Spezialisierungsbeziehung) mit einem anderen Typen verbunden ist, gleichzeitig eine Instanz des generellen Typs ist. Man bezeichnet die beteiligten Typen auch als *Subtyp* und *Supertyp*. S. [Lisk88].

Im Folgenden werden einige Regeln definiert, welche die Verwendung von Spezialisierung dahingehend anleiten, dass die Konsistenz von Modellen mit Spezialisierungsbeziehungen erhalten bleibt. Darüber hinaus ist der Sprachanwender angehalten, die Semantik der Konzepte bei der Erzeugung spezieller Typen zu berücksichtigen und somit potenziell problematische Spezialisierungen zu vermeiden.

Regel 1

Das Löschen bzw. Unterdrücken von Metaattributen, Attributen, Name/Werte-Paaren, Beziehungen sowie Einschränkungen des generellen Typs im speziellen Typ ist nicht erlaubt.

Regel 2

Das Hinzufügen von Attributen, Name/Werte-Paaren, Beziehungen sowie Einschränkungen ist beliebig innerhalb der Vorgaben durch das Metamodell erlaubt.

Regel 3

Das Überschreiben von Metaattributen des generellen Typs im speziellen Typ ist eingeschränkt erlaubt. Somit darf das Metaattribut im speziellen Typ einen neuen Wert erhalten. Beim Überschreiben des Werts muss der Modellierer darauf achten, dass der neue Wert eine gleich scharfe oder schärfere Einschränkung repräsentiert als der ursprünglich definierte. Auf diese Weise wird sichergestellt, dass die Menge der Instanzen des speziellen Typs eine Teilmenge der Menge der Instanzen des generellen Typs ist und nicht noch weitere Instanzen hinzugefügt werden[1].

Beispiel: *HardwareInterface*

Erlaubt: *minBandwidth = 10 MBit/s* wird zu *minBandwidth = 100 MBit/s*

Verboten: *synchronous = false* wird zu *synchronous = true*

Regel 4

Das Überschreiben von Attributen des generellen Typs im speziellen Typ ist nicht erlaubt. Weder Name noch Typ dürfen geändert werden.

Regel 5

Das Überschreiben von Name/Werte-Paaren des generellen Typs im speziellen Typ ist eingeschränkt erlaubt. Der Name darf dabei nicht geändert werden, jedoch kann der zugeordnete Wert ein anderer sein. Beim Überschreiben des Werts muss der Modellierer darauf achten, dass der neue Wert eine gleich scharfe oder schärfere Einschränkung repräsentiert als der ursprünglich definierte. Analog zu Regel 3 wird so sichergestellt, dass die Menge der Instanzen des speziellen Typs eine Teilmenge der Menge der Instanzen des generellen Typs ist und nicht noch weitere Instanzen hinzugefügt werden.

Beispiel: *Location*

Erlaubt: *maxEmployees = 4* wird zu *maxEmployees = 1*

Verboten: *maxEmployees = 4* wird zu *maxEmployees = 6*

Regel 6

Das Überschreiben von Einschränkungen des generellen Typs im speziellen Typ ist eingeschränkt erlaubt. Analog zu Regel 4 darf der Identifikator nicht geändert werden, lediglich der Ausdruck ist

[1] Das Überschreiben eines Wertes mit einer leeren Menge ist nicht gleichbedeutend mit dem Löschen eines Metaattributs. So wird im ersten Fall darauf hingewiesen, dass keine Ausprägungen mit konkreten Werten existieren. Im zweiten Fall entsteht ein undefinierter Zustand. Dies gilt analog für Name/Werte-Paare.

zu überschreiben. Es muss sichergestellt sein, dass der neue Ausdruck gleich scharf oder schärfer ist als der ursprünglich definierte.

Beispiel: beliebiger Typ

Erlaubt: *maxNoOfInstances <= 50* wird zu *maxNoOfInstances <= 30*

Verboten: *maxNoOfInstances <= 50* wird zu *maxNoOfInstances <= 70*

Erlaubt: *„Nur Mitarbeiter dürfen Räume diesen Typs betreten"* wird zu *„Nur Abteilungsleiter dürfen Räume diesen Typs betreten"*

Verboten: *„Nur Mitarbeiter dürfen Räume diesen Typs betreten"* wird zu *„Zu Räumen diesen Typs besteht uneingeschränkter Zugang"*

Regel 7

Das Überschreiben von Beziehungen eines generellen Typs zu einem anderen Typ ist für den speziellen Typ eingeschränkt erlaubt. Gegeben seien die Typen *A*, *B* und *C*, wobei *B* eine Spezialisierung von *A* ist und *A* eine Beziehung zu *C* hat. Die Kardinalitäten der Beziehungen müssen so festgelegt werden, dass die Menge der Instanzen der Beziehung zwischen *B* und *C* eine Teilmenge der Menge der Instanzen der Beziehung zwischen *A* und *C* ist. Auf diese Weise wird sichergestellt, dass keine zusätzlichen Ausprägungen dieser Beziehung mehr hinzukommen.

Beispiel: Beziehung zwischen einem *HardwareDevice* und einem *CommunicationProtocol*, wobei *HardwareDevice* spezialisiert wird

Erlaubt: *„Ein Hardwaregerätetyp sendet mittels fünf Kommunikationsprotokollen, ein Kommunikationsprotokoll wird von beliebig vielen Hardwaregerätetypen verwendet"* wird zu *„Ein Hardwaregerätetyp sendet mittels einem Kommunikationsprotokoll, ein Kommunikationsprotokoll wird von beliebig vielen Hardwaregerätetypen verwendet"*

Verboten: *„Ein Hardwaregerätetyp sendet mittels einem Kommunikationsprotokoll, ein Kommunikationsprotokoll wird von beliebig vielen Hardwaregerätetypen verwendet"* wird zu *„Ein Hardwaregerätetyp sendet mittels zwei Kommunikationsprotokollen, ein Kommunikationsprotokoll wird von beliebig vielen Hardwaregerätetypen verwendet"*

Die Vererbung von Beziehungen zwischen Typen bringt einige Probleme mit sich, die an dieser Stelle nicht weiter vertieft werden sollen[1]. Ungeachtet dessen soll zugunsten einer möglichst komfortablen Wiederverwendung von Modellelementen nicht auf sie verzichtet werden. Generell lassen sich die Regeln, welche eingeschränkte Änderungen an Typeigenschaften erlauben, nicht durch die Beschränkung auf eine rein syntaktische Betrachtung der Modelle bzw. Typen überprüfen. Hier ist der Modellierer besonders gefordert, die Konsistenz des Modells zu überwachen und die Sinnhaftigkeit der von ihm definierten Typen zu hinterfragen.

Die Generalisierung von Typen ist in der ITML ebenfalls erlaubt. Dabei ist zu beachten, dass bei der Erzeugung eines generellen Supertypen dieser alle gemeinsamen Eigenschaften seiner speziellen Subtypen aufweist. *Gemeinsam* bedeutet im Falle von Metaattributen, Attributen, Name/Werte-Paare, Beziehungen und Einschränkungen im Idealfall eine vollkommene syntaktische Übereinstimmung in den jeweiligen Subtypen. Allerdings muss dies vor allem im Falle von Einschränkungen im Einzelfall auch auf inhaltliche, also semantische Korrespondenz bezogen werden. Dies gilt bedingt durch die Möglichkeit des Verwendens von natürlichsprachlichen Formulierungen sowie unscharfer Semantik ggf. auch in Bezug auf die anderen Konzepte.

[1] Hier sind bspw. die Ko- und Kontravarianz zu nennen (s. [Pras02]).

5 Zusammenfassung und Ausblick

In diesem Beitrag wurde die Spezifikation einer domänenspezifischen Modellierungssprache für die Anwendung im Rahmen einer Methode für das IT-Management vorgestellt. Die Herleitung der Sprache basiert auf den zuvor identifizierten Aufgaben des IT-Managements sowie den in der Folge formulierten Anforderungen an eine derartige Sprache. Die Sprachspezifikation erfolgt semiformal durch ein Metamodell. Dieses beschreibt die abstrakte Syntax sowie in gewissem Umfang die Semantik der Sprachkonzepte für verschiedene Teilbereiche des IT-Managements. Zunächst sind hierbei die hard- und softwarespezifischen Konzepte zur Darstellung der technischen Basis einer IT-Landschaft zu nennen. Weitere Sprachkonzepte decken organisationsspezifische Aspekte ab. Diese betreffen vor allem die Aufbau- und Ablauforganisation, Strategien sowie von der Organisation im Rahmen ihrer Aufgaben angebotene Services und deren Beziehungen zur betrieblichen IT. Das Informationssystem als höhere Abstraktion über IT sowie damit zusammenhängend Aspekte der Sicherheit im Rahmen einer Continuity-Planung wurden ebenfalls berücksichtigt. Die Wiederverwendungskonzepte Typisierung/Kapselung und Generalisierung/Spezialisierung/Vererbung sollen die Wirtschaftlichkeit der Modellierungstätigkeiten mit der Sprache grundlegend fördern und die Modellqualität insgesamt positiv beeinflussen. Die in der Sprache enthaltenen Konstrukte sind sowohl für Analysen von IT-Landschaften als auch für deren Planung im Rahmen des IT-Managements verwendbar. Integritätsbedingungen in Form von Kardinalitäten dienen der genaueren Spezifizierung der möglichen Interaktionen mit Organisationsrollen (Kunden, Mitarbeiter, Abteilungen, externe Partner etc.) mit der betrieblichen IT. Schließlich wird durch die Integration mit bereits existierenden DSLs innerhalb von MEMO eine breitere Anwendbarkeit der Sprache erreicht sowie eine Reihe attraktiver Verwendungsmöglichkeiten im Rahmen der Unternehmensmodellierung eröffnet.

Für einen zielführenden Einsatz der Sprache fehlen zum jetzigen Zeitpunkt allerdings noch einige grundlegende Bestandteile, deren Entwicklung Gegenstand zukünftiger Forschungstätigkeiten sein wird. Zunächst ist eine möglichst ansprechende und anschauliche Notation zu erstellen. Die Abbildung der Sprachkonzepte auf die Notationssymbole muss nachvollziehbar sein und eine einfache Sprachanwendung fördern. Darüber hinaus müssen Regeln definiert werden, welche die Verwendung der Sprachkonzepte in unterschiedlichen Kontexten festlegen. Dies kann mittels eines Vorgehensmodells erfolgen. Dieses sollte die Integration in die MEMO-Vorgehensmodelle einerseits und die Verwendung innerhalb von Best Practice-Ansätzen für das IT-Management andererseits thematisieren. Somit können die Potenziale aller beteiligten Ansätze angemessen berücksichtigt werden. Vor diesem Hintergrund bieten Referenzmodelle eine weitere Unterstützung, da sie zum einen exemplarisch aufzeigen, wie die Sprache zu verwenden ist. Zum anderen bilden sie idealtypisch Teile einer Domäne ab und können in anderen Kontexten wiederverwendet werden.

Zuletzt ist noch der Bedarf an Werkzeugunterstützung zu erwähnen. Ein Modellierungswerkzeug ist erforderlich, um die Sprache komfortabel anzuwenden. Darüber hinaus kann durch die Integration von Auswertungsvorschriften in das Werkzeug die Analyse einer IT-Landschaft auf Modellebene nachhaltig unterstützt werden. Auch die Anbindung an eine Datenbank, welche den Bestand der konkreten IT-Komponenten verwaltet (z.B. in Form einer *Configuration Management Database*, CMDB) ist somit möglich. Die Implementierung der Sprache kann zunächst durch ein Metamodellierungswerkzeug erfolgen, in einer späteren Phase auch durch ein dediziertes Werkzeug.

Der methodische Rahmen, der aus der Modellierungssprache und den noch fehlenden Bestandteilen gebildet wird, ist in Zukunft praxisnah zu evaluieren und ggf. anwendungsbegleitend anzupassen. Er verspricht jedoch schon zum jetzigen Zeitpunkt der Entwicklung die Lücke, welche zwischen den bisher zur Verfügung stehenden Ansätzen für das IT-Management und Modellierungsansätzen besteht, auszufüllen und somit das IT-Management im Unternehmen insgesamt anzureichern und zum Erfolg zu führen.

6 Literatur

[ACB89] Ashurst, C.; Doherty, N.F.: Towards the Formulation of a 'Best Practice' Framework for Benefits Realisation in IT Projects. In: Electronic Journal of Information Systems Evaluation (EJISE), Vol. 6, Issue 2, 2003

[Balz00] Balzert, H.: Lehrbuch der Software-Technik: Software-Entwicklung. 2. Aufl., Spektrum, Heidelberg et al., 2000

[Bart02] Barton, N.: Measuring I.T. Performance – From Functional Models to Service Models. In: Brown, A.; Remenyi, D. (Eds.): Proceedings of the Ninth European Conference on Information Technology Evaluation, S. 55-61, Paris, 2002

[Böhm04] Böhmann, T.: Modularisierung von IT-Dienstleistungen: Eine Methode für das Service-Engineering. Gabler, Wiesbaden, 2004

[Brau96] Braun, S.: Die Prozeßkostenrechnung. 2. Aufl., Verlag Wissenschaft und Praxis, Sternenfels et al., 1996

[Bren94] Brenner, W.: Grundzüge des Informationsmanagements. Springer, Berlin et al., 1994

[BRS02] Brenner, M.; Radisic, I.; Schollmeyer, M.: A Criteria Catalog Based Methodology for Analyzing Service Management Processes. In: Freidun, M.; Kropf, P.; Babin, G.: (Eds.): Proceedings of the 13th IFIP/IEEE International Workshop on Distributed Systems: Operations & Management (DSOM 2002). LNCS 2506, Montreal, 2006

[Bryn93] Brynjolfsson, E.: The productivity paradox of information technology. In: Communications of the ACM, Vol. 36, No. 12, S. 67-77, 1993

[Chen76] Chen, P.P.: The Entity-Relationship Model – Toward a Unified View of Data. In: ACM Transactions on Database Systems, Vol. 1, No. 1, S. 9-36, 1976

[Cobi06] o.V.: CobiT 4.0: Control Objectives, Management Guidelines, Maturity Models. ITGI, 2006

[DoDA04] o.V.: DoD Architecture Framework Version 1.0 - Deskbook. DoD Architecture Framework Working Group, 2004

[Eise02] Eisele, W.: Technik des betrieblichen Rechnungswesens. 7. Aufl., Vahlen, München, 2002

[Erl05] Erl, T.: Service-Oriented Architecture: Concepts, Technology and Design. Prentice Hall, New York et al., 2005

[EsJa01] Esser, R.; Jannek, J.W.: A Framework for Defining Domain-Specific Visual Languages, In: Proceedings of the Workshop on Domain Specific Visual Languages at OOPSLA 2001, Tampa Bay, 2001

[FeSi98] Ferstl, O. K., Sinz E.J.: SOM – Modeling of Business Systems. In: Bernus, P.; Mertins, K.; Schmidt, G. (Eds.): Handbook on Architectures of Information Systems. S. 339-358, Springer, Berlin et al., 1998

[Fran94] Frank, U.: Multiperspektivische Unternehmensmodellierung: Theoretischer Hintergrund und Entwurf einer objektorientierten Entwicklungsumgebung. Oldenbourg, München, 1994

[Fran98a] Frank, U.: The MEMO Meta-Metamodel. Arbeitsberichte des Instituts für Wirtschaftsinformatik, Nr. 9, Koblenz, 1998

[Fran98b] Frank, U.: The MEMO Object Modelling Language (MEMO-OML). Arbeitsberichte des Instituts für Wirtschaftsinformatik, Nr. 10, Koblenz, 1998

[Fran98c] Frank, U.: Applying the MEMO-OML: Guidelines and Examples. Arbeitsberichte des Instituts für Wirtschaftsinformatik, Nr. 11, Koblenz, 1998

[Fran99] Frank, U.: Memo: Visual Languages for Enterprise Modelling. Arbeitsberichte des Instituts für Wirtschaftsinformatik, Nr. 18, Koblenz, 1999

[Fran02] Frank, U.: Multi-Perspective Enterprise Modelling (MEMO) - Conceptual Framework and Modelling Languages. In: Proceedings of the Hawaii International Conference on System Sciences (HICSS-35), S. 10 ff., Honolulu, 2002

[Frie04] Friedl, B.: Kostenrechnung. Oldenbourg, München et al., 2004

[FrLa03] Frank, U.; van Laak, B. L.: Anforderungen an Sprachen zur Modellierung von Geschäftsprozessen. Arbeitsberichte des Instituts für Wirtschaftsinformatik, Nr. 34, Koblenz, 2003

[FrLa06] Frank, U.; Lange, C.: E-MEMO: A Method to support the Development of customized Electronic Commerce Systems. In: Information Systems and E-Business Management (ISeB), Springer, Berlin et al., 2006

[GHG04] van Grembergen, W.; De Haes, S.; Guldentops, E.: Structures, Processes and Relational Mechanisms for IT Governance. In: van Grembergen, W. (Ed.): Strategies for Information Technology Governance. Idea Group, Hershey et al., 2004

[GHJ+95] Gamma, E.; Helm, R.; Johnson, R.; Vlissides, J.: Design Patterns – Elements of Reusable Object-oriented Software. Addison-Wesley, Reading, 1995

[GiHi05] Gitzel, R.; Hildenbrand, T.: A Taxonomy of Metamodel Hierarchies. Research Report, Department of Information Systems, Universität Mannheim, 2005

[GKP98] Geisler, R.; Klar, M.; Pons, C.: Dimensions and Dichotomy in Metamodelling. Bericht-Nr. 98-5, Institut für Kommunikations- und Softwaretechnik, TU Berlin, 1998

[GRB04] Grob, H.L.; Reepmeyer, J.-A.; Bensberg, F.: Einführung in die Wirtschaftsinformatik. 5. Aufl., Vahlen, München, 2004

[Hand92] Hand, M.: Managing strategic investment in information systems. In: Dixon, R.; Franks, R. (Hrsg.): IT Management Handbook, S. 7-27, Butterworth – Heinmann, Oxford et al., 1992

[HBS06] Huppertz, P.G., Bause, M.; Swidlowski, S.: IT-Service – Der Kern des Ganzen. ITSM Advanced Pocket Book, Band 6, Serview GmbH, Bad Homberg, 2006

[Hein99] Heinrich, L.J.: Informationsmanagement. 6. Aufl., Oldenbourg, München et al., 1999

HeKu00] Heemstra, F.J.; Kusters, R.J.: Assessing IT-investments: costs, benefits, risks. In: Kusters, R.; Maxwell, K.D.; Cowderoy, A. (Hrsg.): Proceedings of ESCOM-SCOPE, S. 307-315, München, 2000

[HMT02] Heberling, M.; Maier, Ch., Tensi, T.: Visual Modeling and Managing the Software Architecture Landscape in a Large Enterprise by an Extension of the UML, In: Proceedings of the Second Workshop on Domain-Specific Visual Languages – OOPSLA 2002, Seattle, 2002

[Horv03] Horváth, P.: Controlling. 9. Aufl., Vahlen, München, 2003

[ISO94] o.V.: ISO/IEC 7498-1: Information Technology - Open Systems Interconnection - Basic Reference Model: The Basic Model, ISO, 1994

[ISO99] o.V.: ISO/IEC 14598: Information Technology - Software Product Evaluation - Part 1: General Overview, ISO, 1999

[ITIL02] o.V.: ICT Infrastructure Management. CD-ROM Version, Office of Government Commerce, Norwich, 2002

[Jung02] Jung, J.: Some Reflections on the Basic Conceptualisation of a Resource Modelling Language for Business Process Modelling - Concepts, Requirements and Open Research Questions. Arbeitsberichte des Instituts für Wirtschaftsinformatik, Nr. 35, Koblenz, 2002

[Jung03] Jung, J.: Basic Conceptualisation of a Resource Modelling Language. In: Khosrow-Pour, M (Hg.): Information Technology and Organizations: Trends, Issues, Challenges and Solutions. Proceedings of the 2003 Information Resources Management Association International Conference Philadelphia, S. 827-831, Philadelphia, 2003

[Jung06] Jung, J.: Supply Chains in the Context of Enterprise Modelling. In: Mayr, H.; Breu, R. (Hrsg.): Proceedings der Modellierung 2006, S.193-202, Innsbruck, 2006

[Jung07] Jung, J.: Entwurf einer Sprache für die Modellierung von Ressourcen im Kontext der Geschäftsprozessmodellierung. Dissertation, Universität Duisburg-Essen, Essen, erscheint 2007

[KaCo99] Kaplan, R.; Cooper, R.: Prozesskostenrechnung als Managementinstrument. Campus, Frankfurt/Main et al., 1999

[KiJu06] Kirchner, L.; Jung, J.: A Framework for the Evaluation of Meta-Modelling Tools. In: Remenyi, D.; Brown, A. (Eds.): Proceedings of the 13th European Conference on Information Technology Evaluation (ECITE 2006), S. 307-315, Genua, 2006

[Kirc05] Kirchner, L.: Cost Oriented Modelling of IT-Landscapes: Generic Language Concepts of a Domain Specific Language. In: Desel, J.; Frank, U. (Eds.): Proceedings of the Workshop on Enterprise Modelling and Information Systems Architectures (EMISA ´05), S. 166-179, Klagenfurt, 2005

[Kirc06] Kirchner, L.: Unterstützung der Analyse von betrieblichen Informationssystemen im Rahmen der strategischen IT-Planung durch eine domänenspezifische Modellierungssprache. In: Schelp, J.; Winter, R.; Frank, U.; Rieger, B.; Turowski, K.: Integration, Informationslogistik und Architektur. Proceedings der DW2006, S. 225- 246, Friedrichshafen, 2006

[Kirs99] Kirsch, J.: Prozessorientiertes Management von Client-Server-Systemen, Deutscher Universitätsverlag, Wiesbaden, 1999

[Koeh05] Köhler, P.T.: ITIL. Springer, Berlin et al., 2005

[Krcm00] Krcmar, H.; Informationsmanagement. 2. Aufl., Springer, Berlin et al., 2000

[Kuet03] Kütz, M.: Balanced Scorecard im IT-Controlling. In: Blomer, R.; Bernhard, M.G. (Hrsg.): Balanced Scorecard in der IT. S. 21-35, symposion, Düsseldorf, 2003

[KuLo95] Küting, K.; Lorson, P.: Stand, Entwicklung und Grenzen der Prozeßkostenrechnung. In: Männel, W. (Hrsg.): Prozeßkostenrechnung. Gabler, Wiesbaden, 1995

[LHM95] Lehner, F.; Hildebrand, K.; Maier, R.: Wirtschaftsinformatik: Theoretische Grundlagen. Hanser, München et al., 1995

[LHP01] Lillrank, P.; Holopainen, S.; Paavola, T.: Catching Intangible IT Benefits, In: Electronic Journal of Information Systems Evaluation (EJISE), Paper 4, Issue 4, 2001

[Lisk88] Liskov, B.: Data abstraction and hierarchy. ACM SIGPLAN Notices, 23(5), S. 17-34, 1988

[LKT04] Luoma, J.; Kelly, S.; Tolvanen, J.-P.: Defining Domain-Specific Modeling Languages: Collected Experiences. Proceedings of the 4th OOPSLA Workshop on Domain-Specific Modeling (DSM'04), Vancouver, 2004

[Maen95] Männel, W.: Zur Bedeutung der Prozesskostenrechnung. In: Männel, W. (Hrsg.): Prozeßkostenrechnung. Gabler, Wiesbaden, 1995

[MBK+05] Mertens, P.; Bodendorf, F.; König, W.; Picot, A.; Schumann, M.; Hess, Th.: Grundzüge der Wirtschaftsinformatik. 9. Aufl., Springer, Berlin et al., 2005

[MEH01] Maier, M.W.; Emery, D.; Hilliard, R.: Software Architecture: Introducing IEEE Standard 1471. In: Computer, Vol. 34, No. 4, S. 107-109, IEEE, 2001

[Mens98] Mensch, G.: Kosten-Controlling. Oldenbourg, München, 1998

[Mert01] Mertens, P. (Hrsg.): Lexikon der Wirtschaftsinformatik. 4. Aufl., Springer, Berlin et al., 2001

[MuOe99] Muschter, S.; Österle, H.: Investitionen in Standardsoftware: Ein geschäftsorientierter Ansatz zur Nutzenmessung und –bewertung. In: Scheer, A.-W.; Nüttgens, M. (Hrsg.): Proceedings zur 4. internationalen Tagung Wirtschaftsinformatik 1999, Heidelberg

[MZK03] Meyer, M.; Zarnekow, R.; Kolbe. L.M.: IT Governance: Begriff, Status quo und Bedeutung. In: Wirtschaftsinformatik. Band 45, Nr. 4, S. 445-448, Vieweg, Wiesbaden, 2003

[NCT+05] Niessink, F.; Clerc, V.; Tijdink, T.; van Vliet, H.: The IT Service Capability Maturity Model. Version 1.0, RC1. Vrije Universiteit, Amsterdam, 2005

[Niem06] Niemann, K.D.: From Enterprise Architecture to IT-Governance: Elements of Effective IT-Management. Vieweg, Wiesbaden, 2006

[Nijl04] Nijland, M.: IT cost benefit management improvement from a critical perspective. In: Electronic Journal of Information Systems Evaluation (EJISE), Vol. 7, Issue 1, 2004

[OBH92] Österle, H.; Brenner, W.; Hilbers, K.: Unternehmensführung und Informationssystem. Teubner, Stuttgart, 1992

[Olbr04] Olbrich, A.: ITIL kompakt und verständlich. 2. Aufl., Vieweg, Wiesbaden, 2004

[OMG03] o.V.: UML 2.0 OCL Specification. Object Management Group, 2003

[OMG04] o.V.: UML 2.0 Infrastructure Specification. Object Management Group, 2004

[OMG05] o.V.: UML Superstructure Specification. Object Management Group, 2005

[Pete04] Peterson, R.R.: Integration Strategies and Tactics for Information Technology Governance. In: van Grembergen, W. (Ed.): Strategies for Information Technology Governance. Idea Group, Hershey et al., 2004

[PMK04] Pietsch, Th.; Martiny, L.; Klotz, M.: Strategisches Informationsmanagement. 4. Aufl., Erich Schmidt Verlag, Berlin et al., 2004

[Port85] Porter, M.: Competitive Advantage: Creating and Sustaining Superior Performance. The Free Press, New York, 1985

[Pott98] Potthof, I.: Empirische Studien zum wirtschaftlichen Erfolg der Informationsverarbeitung. In: Wirtschaftsinformatik, Ausgabe 1-1998, S. 54-65, Vieweg, Wiesbaden, 2002

[Pras02] Prasse, M.:: Entwicklung und Formalisierung eines objektorientierten Sprachmodells als Grundlage für MEMO-OML. Dissertation, Universität Koblenz-Landau, Koblenz, 2002

[Reme00] Remenyi, D.: The Elusive Nature of Delivering Benefits from IT Investment. In: Electronic Journal of Information Systems Evaluation (EJISE), Vol. 3, Issue 1, 2000

[Sche95] Scheer, A.-W.: Wirtschaftsinformatik – Referenzmodelle für industrielle Geschäftsprozesse. Springer, Berlin et al., 1995

[Sche02] Scheer, A.-W.: ARIS – Vom Geschäftsprozess zum Anwendungssystem. 4. Aufl., Springer, Berlin et al., 2002

[Schm99] Schmidt, G.: Informationsmanagement. 2. Aufl., Springer, Berlin et al., 1999

[StHa05] Stahlknecht, P.; Hasenkamp, U.: Einführung in die Wirtschaftsinformatik. 11. Aufl., Springer, Berlin et al., 2005

[Stic01] Stickel, E.: Informationsmanagement. Oldenbourg, München et al., 2001

[Tane05] Tanenbaum, A.: Computernetzwerke. 4. Ed, Pearson, München, 2005

[Tens05] Tensi, T.: Anwendungslandschaft: Nachverfolgung von IT-Modellen. In: Breu, R.; Matzner, T.; Nickl, F.; Wiegert, O. (Hrsg.): Software-Engineering. S. 5-26, Oldenburg, 2005

[TOGA04] o.V.: TOGAF Documentation. Catalog Number I917, Free HTML Version, The OPEN Group, 2004

[WBW03] Winter, A.; Brigl, B.; Wendt, Th.: 3LGM² Methode und Werkzeug zur Modellierung von Unternehmensarchitekturen im Krankenhaus, In: Proceedings der Frühjahrskonferenz 2003 des GI Arbeitskreises Enterprise Architecture, S. 20-32, St. Gallen, 2003

[WHB+04] Wendt Th.; Häber, A.; Brigl, B.; Winter, A.: Modeling Hospital Information Systems (Part 2): Using the 3LGM2 Tool for Modeling Patient Record Management. Methods of Information in Medicine 43(3), S. 256-67, 2004

[WeRo04] Weill, P.; Ross, J.W.: IT Governance: How Top Performers Manage IT Decision Rights for Superior Results. Harvard Business School Press, Boston, 2004

[WoMü03] Wolle, B.; Müller, V.: Prozessorientiertes IT-Qualitätsmanagement. In: Brenner, W.; Meier, A.; Zarnekow, R. (Hrsg): Strategisches IT-Management. Praxis der Wirtschaftsinformatik Nr. 232, HMD, S. 66-78, dpunkt, Heidelberg, 2003

[ZaBr03] Zarnekow, R.; Brenner, W.: Auf dem Weg zu einem produkt- und diensleistungsorientierten IT-Management. In: Brenner, W.; Meier, A.; Zarnekow, R. (Hrsg): Strategisches IT-Management. Praxis der Wirtschaftsinformatik Nr. 232, HMD, S. 7-16, dpunkt, Heidelberg, 2003

Previously published ICB - Research Reports

2007

No 11 (February 2007)
Kirchner, Lutz: "Entwurf einer Modellierungssprache zur Unterstützung der Aufgaben des IT-Managements – Grundlagen, Anforderungen und Metamodell"

No 10 (February 2007)
Schauer, Carola; Strecker, Stefan: "Vergleichende Literaturstudie aktueller einführender Lehrbücher der Wirtschaftsinformatik: Bezugsrahmen und Auswertung"

No 9 (February 2007)
Strecker, Stefan; Kuckertz, Andreas; Pawlowski, Jan M.: "Überlegungen zur Qualifizierung des wissenschaftlichen Nachwuchses: Ein Diskussionsbeitrag zur (kumulativen) Habilitation"

No 8 (February 2007)
Frank, Ulrich; Strecker, Stefan; Koch, Stefan: "Open Model – Ein Vorschlag für ein Forschungsprogramm der Wirtschaftsinformatik (Langfassung)"

2006

No 7 (December 2006)
Frank, Ulrich: "Towards a Pluralistic Conception of Research Methods in Information Systems Research"

No 6 (April 2006)
Frank, Ulrich: "Evaluation von Forschung und Lehre an Universitäten – Ein Diskussionsbeitrag"

No 5 (April 2006)
Jung, Jürgen: "Supply Chains in the Context of Resource Modelling"

No 4 (February 2006)
Lange, Carola: "Development and status of the Information Systems / Wirtschaftsinformatik discipline: An interpretive evaluation of interviews with renowned researchers, Part III – Results Wirtschaftsinformatik Discipline"

2005

No 3 (December 2005)
Lange, Carola: "Development and status of the Information Systems / Wirtschaftsinformatik discipline: An interpretive evaluation of interviews with renowned researchers, Part II – Results Information Systems Discipline"

No 2 (December 2005)
Lange, Carola: "Development and status of the Information Systems / Wirtschaftsinformatik discipline: An interpretive evaluation of interviews with renowned researchers, Part I – Research Objectives and Method"

No 1 (August 2005)
Lange, Carola: „Ein Bezugsrahmen zur Beschreibung von Forschungsgegenständen und -methoden in Wirtschaftsinformatik und Information Systems"

The Institute for Computer Science and Business Information Systems (ICB), located at the Essen Campus, is dedicated to research and teaching in Applied Computer Science, Information Systems as well as Information Management. The ICB research groups cover a wide range of expertise:

Research Group	Core Research Topics
Prof. Dr. H. H. Adelsberger Information Systems for Production and Operations Management	E-Learning, Knowledge Management, Skill-Management, Simulation, Artificial Intelligence
Prof. Dr. P. Chamoni MIS and Management Science / Operations Research	Information Systems and Operations Research, Business Intelligence, Data Warehousing
Prof. Dr. F.-D. Dorloff Procurement, Logistics and Information Management	E-Business, E-Procurement, E-Government
Prof. Dr. K. Echtle Dependability of Computing Systems	Dependability of Computing Systems
Prof. Dr. S. Eicker Information Systems and Software Engineering	Process Models, Software-Architectures
Prof. Dr. U. Frank Information Systems and Enterprise Modelling	Enterprise Modelling, Enterprise Application Integration, IT Management, Knowledge Management
Prof. Dr. M. Goedicke Specification of Software Systems	Distributed Systems, Software Components, CSCW
Prof. Dr. T. Kollmann E-Business and E-Entrepreneurship	E-Business and Information Management,E-Entrepreneurship/ E-Venture, Virtual Marketplaces and Mobile Commerce, Online-Marketing
Prof. Dr. B. Müller-Clostermann Systems Modelling	Performance Evaluation, Modelling and Simulation, SAP Capacity Planning for R/3 and mySAP.com, Tools for Queueing Network Analysis and Capacity Planning, Communication Protocols and Distributed Systems, Mobile Systems
Prof. Dr. K. Pohl Software Systems Engineering	Requirements Engineering, Software Quality Assurance, Software-Architectures, Evaluation of COTS/Open Source-Components
Prof. Dr.-Ing. E. Rathgeb Computer Networking Technology	Computer Networking Technology
Prof. Dr. R. Unland Data Management Systems and Knowledge Representation	Data Management, Artificial Intelligence, Software Engineering, Internet Based Teaching
Prof. Dr. S. Zelewski Institute of Production and Industrial Information Mangement	Industrial Business Processes, Innovation Management, Information Management, Economic Analyses

For more information visit us on the Web: **http://www.icb.uni-due.de**

ISSN 1860-2770