Otto-von-Guericke-Universität Magdeburg

Fakultät für Informatik
Institut für Technische und Betriebliche Informationssysteme
Arbeitsgruppe Wirtschaftsinformatik - Managementinformationssysteme

Masterarbeit:

Entwicklung eines Metamodells zur Abbildung von Systemlandschaften in kleinen und mittleren Unternehmen im Kontext von „as a Service“

Vorgelegt von:

Matthias Splieth

13. Oktober 2011

Eingereicht bei:

Prof. Dr. Hans-Knud Arndt
Prof. Dr. Klaus Turowski

Universität Magdeburg
Fakultät für Informatik
Postfach 4120, D–39016 Magdeburg
Germany

Splieth, Matthias:
Entwicklung eines Metamodells zur Abbildung von Systemlandschaften in kleinen und mittleren Unternehmen im Kontext von „as a Service“
Masterarbeit, Otto-von-Guericke-Universität Magdeburg, 2011.

Inhaltsverzeichnis

Abbildungsverzeichnis

Tabellenverzeichnis

Abkürzungsverzeichnis

ASP	Application Service Providing
CRM	Customer Relationship Management
DBMS	Datenbankmanagementsystem
ERP	Enterprise Resource Planning
EU	Europäische Union
IaaS	Infrastructure as a Service
IKT	Informations- und Kommunikationstechnologie
IKS	Informations- und Kommunikationssysteme
IT	Informationstechnologie
ITML	IT Modeling Language
JDK	Java Development Kit
JRE	Java Runtime Environment
KMU	Kleine und mittlere Unternehmen
MEMO	Multi-Perspective Enterprise Modelling
NIST	National Institute of Standards and Technology
PaaS	Platform as a Service
SaaS	Software as a Service
SLA	Service Level Agreement
SOA	Serviceorientierte Architektur
UML	Unified Modeling Language
USV	Unterbrechungsfreie Stromversorgung

Kapitel 1

Einleitung

Im Rahmen dieses Kapitels wird dem Leser verdeutlicht, welche Motivation hinter der Bearbeitung der in dieser Arbeit beachteten Problemstellung steckt. Diese wird einleitend in Abschnitt 1.1 dargelegt. Des Weiteren wird auf die wissenschaftliche Zielstellung eingegangen, welche in Abschnitt 1.2 vorgestellt wird. Abschließend wird in Abschnitt 1.3 die Struktur der Arbeit erläutert.

1.1 Motivation

In kleinen und mittleren Unternehmen (**KMU**) herrscht nach wie vor eine große Zurückhaltung bezüglich Investitionen im Bereich der Informationstechnologie (**IT**). Aufgrund der seit Jahren rückläufigen und in KMU ohnehin begrenzten IT-Budgets ist es kaum verwunderlich, dass die verfügbaren Mittel primär zur Sicherung der Wettbewerbsfähigkeit eingesetzt werden. Jedoch bietet ein zielgerichteter Einsatz der IT gerade für KMU die Möglichkeit, sich am Markt erfolgreich gegen in- und ausländische Konkurrenten durchzusetzen [Mei04]. Allerdings mangelt es KMU, im Gegensatz zu großen Unternehmen, häufig an einer eigenen IT-Abteilung, sodass das Management der IT oft zur nebensächlichen Aufgabe der Geschäftsführung verkommt. Dies ist jedoch problematisch, da es der Geschäftsführung in der Regel an den dafür notwendigen technischen Kenntnissen mangelt. Daraus resultiert beispielsweise eine Vergeudung von Potentialen [MTK10]. Die Bedeutung der Informationstechnologie nimmt gleichzeitig aber immer mehr zu und bestimmt vor allem auch den wirtschaftlichen Erfolg eines Unternehmens maßgeblich mit [Pre07]. Somit spielt natürlich auch die Systemlandschaft als Ganzes eine tragende Rolle hinsichtlich des Unternehmenserfolgs. Zwar betreiben viele Unternehmen bereits einen großen Aufwand für die Dokumentation ihrer Systemlandschaften [AS08]. Jedoch deuten hohe Funktions- und Datenredundanzen, wie sie in Unternehmen häufig vorzufinden sind [Tur99], darauf hin, dass Unternehmen kaum Kenntnisse über die eigene Systemlandschaft haben. Erschwerend kommt hinzu, dass Unternehmen nicht mehr als isolierte Einheit betrachtet werden können, sondern auch die Integration von Partnern eine immer wichtigere Rolle spielt [Die06].
Somit erwächst die Notwendigkeit einer Möglichkeit, den Ist-Zustand einer Systemlandschaft zu dokumentieren, um zunächst einen Überblick über die verwendeten Systeme zu erhalten und zu erfassen, welche fachlichen Funktionen[1] von diesen unterstützt werden.

[1] In sich abgeschlossene Tätigkeit in einem Prozess, z. B. „Kunde anlegen“

Gerade die Verknüpfung zwischen den fachlichen Funktionen und IT-Systemen ist dabei als außerordentlich wichtig anzusehen [AS08]. Daneben ist eine solche Methode beispielsweise hinsichtlich der Planbarkeit und Steuerung einer Systemlandschaft wünschenswert; aber auch Aspekte wie Effektivität und Effizienz oder Anpassungsfähigkeit und Kontinuität sind als wichtig anzusehen [Roh08].
Hinsichtlich der Gestaltung von Systemlandschaften sind dabei vor allem auch aktuelle Trends in der IT von hoher Bedeutung. Dies trifft insbesondere auf die Konzepte des *Cloud Computing*, wie etwa *Software as a Service*, zu [BB11]. Diese sind in den letzten Jahren immer relevanter geworden und stellen mittlerweile einen der wichtigsten IKT[2]-Trends dar [MTK10]. Derartige Ansätze können dabei insbesondere für KMU relevant, beispielsweise hinsichtlich der Kosten, sein. So handelt es sich bei „as a Service"-Angeboten meist um Mietmodelle, die gerade für KMU eine hohe Investitionssicherheit bringen und zudem Kosten sparen können [MTK10] und weiterhin dazu führen, dass Unternehmen selber nicht mehr über das technische Wissen für den Betrieb der Software verfügen müssen. Folglich ist die Berücksichtigung des „as a Service"-Paradigmas im Kontext der Gestaltung von Systemlandschaften notwendig.

1.2 Wissenschaftliche Zielsetzung

In der vorliegenden Arbeit wird ein Metamodell für die Abbildung von Systemlandschaften in KMU vor dem Hintergrund des „as a Service"-Paradigmas entwickelt. Wesentlich dafür ist vor allem das Erarbeiten einer Definition für den Terminus „Systemlandschaft". Auf dieser Grundlage wird dann ein Metamodell für die Abbildung von Systemlandschaften entwickelt, wobei der Fokus auf deren technischen Gegebenheiten liegt. Daneben werden aber auch andere Aspekte, wie etwa Geschäftsprozesse, einbezogen. Die Zielsetzung betrifft demnach unterschiedliche Punkte:

1. Das Modell wird alle für KMU relevanten (technischen) Bestandteile einer Systemlandschaft enthalten und deren Beziehungen untereinander aufzeigen. Dabei werden auch charakteristische Eigenschaften dieser Bestandteile festgehalten.

2. Die Komplexität des Metamodells wird möglichst klein gehalten, um auch Anwendern ohne Fachkenntnisse die Nutzung des Modells zu ermöglichen. Diese Situation ist insbesondere in KMU vorherrschend.

3. Das Modell zeigt auf, welche fachlichen Funktionen von den einzelnen Komponenten der Systemlandschaft offeriert werden.

4. Das Modell ermöglicht die Abbildung von Konzepten des Cloud Computings, die insbesondere für KMU als relevant anzusehen sind.

Diese Ziele werden als Grundlage für die Bewertung des zu entwickelnden Metamodells dienen.

[2]Informations- und Kommunikationstechnologie

1.3 Struktur der Arbeit

Um die zuvor definierte wissenschaftliche Zielstellung zu erreichen, wird folgendes Vorgehen gewählt: Einleitend werden in Kapitel 2 die theoretischen Grundlagen für das Verständnis dieser Arbeit geschaffen. Dies betrifft insbesondere die Erarbeitung von Definition für eine Reihe von Begriffen, für die entweder noch keine Definitionen vorliegen oder für die kein einheitliches Begriffsverständnis gegeben ist. Im Rahmen des Kapitels wird dabei auf Systemlandschaften, Modelle, kleine und mittlere Unternehmen sowie das Cloud Computing eingegangen.

Im Anschluss daran wird in Kapitel 3 eine Analyse bestehender Ansätze für die Modellierung von Systemlandschaften vorgenommen. Dafür wird zunächst ein Kriterienkatalog aufgestellt, auf dessen Basis dann anschließend einzelne Ansätze untersucht werden. Die Ergebnisse dieser Untersuchung werden dann für die Entwicklung das im Rahmen dieser Arbeit zu erstellende Metamodell berücksichtigt.

Die Vorstellung des Metamodells erfolgt in Kapitel 4. Zunächst werden einige von dem Modell einzuhaltende Anforderungen definiert. Auf Grundlage dieser Anforderungen und der Analyse aus Kapitel 3 wird das Metamodell dann detailliert beschrieben. Im Rahmen einer Evaluierung wird dieses im Anschluss bewertet.

Abschließend werden in Kapitel 5 die in der Arbeit gewonnenen Erkenntnisse zusammengefasst und zudem Möglichkeiten für die Fortführung dieser Arbeit aufgezeigt.

Kapitel 2

Theoretische Grundlagen und Begriffsdefinitionen

In diesem Kapitel werden die Grundlagen für das weitere Verständnis dieser Arbeit geschaffen. Daher werden im weiteren Verlauf zunächst die für das Verständnis wesentlichen Begriffe vorgestellt beziehungsweise Begriffsdefinitionen erarbeitet. Zu Beginn wird in Abschnitt 2.1 eine Definition für „Systemlandschaften“ aufgestellt, da für diese bislang kein einheitliches Begriffsverständnis gegeben ist. Im Anschluss daran wird in Abschnitt 2.2 eine Definition für den Begriff „Modell“ ausgearbeitet, der in der Wirtschaftsinformatik häufig Gegenstand von Diskussionen ist. Da KMU den Rahmen für das im Kontext dieser Arbeit zu entwickelnden Metamodells vorgeben, werden diese in Abschnitt 2.3 charakterisiert. Im darauf folgenden Abschnitt 2.4 werden die Konzepte des Cloud Computings erläutert sowie auf die Bedeutung des selbigen für KMU eingegangen. Eine Zusammenfassung schließt das Kapitel.

2.1 Systemlandschaften

Der Begriff *Systemlandschaft*, häufig auch als IT-Landschaft oder Anwendungslandschaft bezeichnet (vergleiche [RHS05, AS08, Roh08]), ist ein in der Literatur häufig verwendeter Terminus. Jedoch mangelt es bisher an einer klaren Definition. Im Rahmen der Verwendung dieses Begriffs werden häufig nur Eigenschaften von Systemlandschaften herausgestellt, ohne ein tatsächliches Begriffsverständnis zu schaffen (vergleiche beispielsweise [GGH+07]). In diesem Abschnitt soll daher anhand einer interdisziplinären Betrachtung des Begriffes „Systemlandschaft“ eine Definition hergeleitet werden, die im weiteren Verlauf der Arbeit als Verständnisgrundlage dient. Dazu werden zunächst die terminologischen Bestandteile von Systemlandschaft, „System“ und „Landschaft“, untersucht und jeweils eine Definition aufgestellt. Im Anschluss daran werden die beiden Begriffe in einen informationstechnischen Kontext eingeordnet und darauf aufbauend eine Definition von Systemlandschaften erarbeitet.

2.1.1 Der Systembegriff

Bei dem Begriff *System* (vom griechischen *systema = Zusammenstellung, geordnetes Ganzes*) handelt es sich um eine Bezeichnung, die nicht allein in der Informatik oder

Wirtschaftsinformatik Verwendung findet. Vielmehr ist in verschiedenen (wissenschaftlichen) Disziplinen eine Vielzahl von Definitionen entstanden (vergleiche beispielsweise [HF56, Bos92, KN00]). Ein homogenes Begriffsverständnis der verschiedenen Definitionen besteht aber dahingehend, dass ein System aus verschiedenen Komponenten und Relationen besteht. So ist ein System nach ISO/IEC als ein „*Objekt aus Elementen, die miteinander verbunden sind oder interagieren*“ [ISO00] definiert. Auch nach Haber besteht ein System aus „*Elementen und Beziehungen*“ [HS09, S. 9]. Zudem wird davon ausgegangen, dass die Systemelemente über Eigenschaften verfügen [Kle07].

Zwar ist das Begriffsverständnis von Systemen hinsichtlich bestimmter Aspekte homogen. Eine Veränderung dieses Verständnisses erfolgte jedoch hinsichtlich der Wandlung systemtheoretischer Paradigmen: Früher wurde ein System als Teil/Ganzes-Schema gesehen. Diese Wahrnehmung hat sich heute über System/Umwelt zu Identität/Differenz gewandelt und so neue Paradigmen im Rahmen der Systemtheorie etabliert [Kle07]. Auf deren ausführliche Darstellung wird an dieser Stelle jedoch Komplexitätsgründen verzichtet. Stattdessen werden im Folgenden die wichtigsten Eigenschaften von Systemen herausgestellt, die insbesondere durch die Paradigmenwechsel entstanden sind. Für detailliertere Beschreibungen empfiehlt sich beispielsweise das Studium von [HF56, vB68, MV80, Luh84, Kle07].

Die Paradigmenwechsel haben dazu geführt, dass sich die Sichtweise auf Systeme verändert hat. Wurde ein System ursprünglich als isoliertes Objekt betrachtet, gehen beispielsweise [KR72, Kle07] heute davon aus, dass auch Elemente außerhalb des Systems existieren müssen: Ein System ist immer in eine *Systemumgebung* (seine Umwelt) eingebettet.

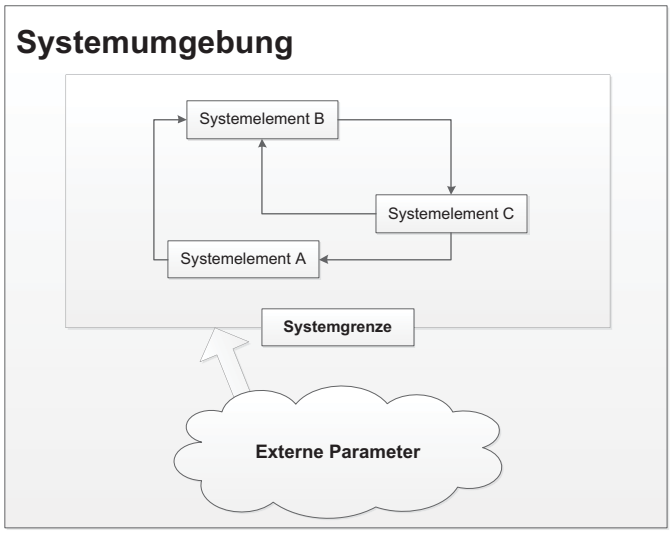

Abbildung 2.1: Aufbau eines Systems

Die wesentliche Eigenschaft eines Systems ist nach Luhmann das Vorhandensein von Operationen in dem System [Luh84]. Diese Operationen sind es, die das System von seiner Umwelt differenzieren. Der Schnittpunkt, an dem das betrachtete System und seine Umwelt aufeinander treffen, wird als *Systemgrenze* bezeichnet [LSK04, Kle07]. Die Grenze ist relevant, um entscheiden zu können, welche Elemente Bestandteile eines

Systems sind beziehungsweise nicht. Ein System kann demnach über Eingaben und Ausgaben verfügen. Daher wird durch das Ziehen der Systemgrenze grundsätzlich zwischen *offenen* (mit Ein- und Ausgabe) und *geschlossenen* (ohne Ein- und Ausgabe) Systemen unterschieden. Abbildung 2.1 verdeutlicht die zuvor genannten Komponenten und Eigenschaften eines Systems.

Die Gesamtheit aller Elemente eines Systems wird als *Systemstruktur* bezeichnet. Diese muss nicht fix sein, da das *Systemverhalten* die Struktur ändern oder Änderungen zumindest stimulieren kann[Kle07]. Aus systemtheoretischer Sicht können Systeme noch weitere Merkmale aufweisen. Dazu zählen beispielsweise:

- *Systemziele*, die durch den Anwendungsbereich bestimmt werden, oder
- die *Integrität* eines Systems, die durch seine Elemente bestimmt wird.

Diese Aufzählung zeigt dabei lediglich einen kleinen Ausschnitt weiterer Kennzeichnung von Systemen und soll lediglich die Komplexität des Systembegriffs verdeutlichen.

Auf Basis der gewonnenen Erkenntnisse kann nun eine Definition des Systembegriffs aufgestellt werden:

> *„Ein (offenes) System beschreibt eine sich durch seine Operationen von seiner Umwelt abgegrenzte Gesamtheit, die über eine Systemschnittstelle mit dieser kommuniziert. Ein System besteht dabei aus einer Menge von Komponenten, die über bestimmte Eigenschaften verfügen und untereinander in Beziehung stehen. Seine Struktur ist durch das Systemverhalten veränderbar.“*

2.1.2 Der Landschaftsbegriff

Bei „Landschaft“ handelt es sich um einen Begriff, für den kein einheitliches Begriffsverständnis gegeben ist (vergleiche beispielsweise [Buc05, Küh08, HS09]). Im Gegensatz zum Systembegriff zeigt sich beim Landschaftsbegriff jedoch eine hohe Diversifizierung hinsichtlich des Begriffsverständnisses (siehe [HS09, Hok09b, Hok09a, Küh08]). Ein Grund hierfür ist die Tatsache, dass „Landschaft“ in vielen verschiedenen wissenschaftlichen Disziplinen, beispielsweise Geographie und Philosophie, von Bedeutung ist und daher vielschichtig diskutiert wird [Küh08, Hok09b]. Landschaft wird dabei als „komplexes Objekt“ [Buc05, Küh08] aus Landschaftselementen, -strukturen und -funktionen gesehen [Ben03].

In der Vergangenheit wurde Landschaft als einfaches Bild eines Raumes betrachtet. Diese Vorstellung gilt aber mittlerweile als überholt [Buc05]. So charakterisiert Ipsen in [Ips06] zwei wesentliche Bedeutungen des Landschaftsbegriffs. Zum einen stellt er die Materialität des Raumes heraus. Zum anderen weist er auf die Konstruktion eines Bildes von Räumen hin. Landschaften beschreiben demnach einerseits eine Beziehung von Menschen und andererseits durch Natur und Arbeit entstandenen Raum [Ips06]. Folglich stellen Landschaften durch Menschen gestaltete Räume dar, die als Konstrukte in den Köpfen der Menschen entstehen [Buc05]. Das Einbeziehen des Faktors Mensch ist dementsprechend ein wesentlicher Schritt hin zum heutigen Begriffsverständnis von Landschaften. Häufig wird in diesem Kontext auch von Kulturlandschaften gesprochen.

Kulturlandschaften bezeichnen Landschaften, die durch menschliches Handeln verändert werden [Gem07]. Da nach dem gegenwärtigen Landschaftsbegriff die Gestaltung von Landschaften bereits durch Menschen erfolgt, ist der Begriff „Kulturlandschaft“ tautologisch. Er soll dabei lediglich *„ein besonderes Interesse am kulturhistorischen Gehalt von Räumen“* [Ben03, S. 122] aufzeigen. Zwar werden die beiden Begriffe teilweise durch unterschiedliche semantische Bedeutungen differenziert (wie in [Hok09a] gezeigt wird), jedoch ist eine Unterscheidung im Rahmen dieser Arbeit nicht relevant. Das Kompositum „Kulturlandschaft“ ist somit für das Verständnis nicht notwendig und folglich kann „Kultur“ als Bestimmungswort entfallen. Für das Verständnis von Landschaften ist nur von Bedeutung, dass Landschaften beiläufig und im Laufe der Zeit entstehen (siehe [Gem07, Küh08]). Um dabei die nachhaltige Entwicklung einer Landschaft zu ermöglichen, ist es nach Lutze et al. wichtig, den Ist-Zustand der Landschaften abzubilden [LSK04]. Die Zustandsbeschreibung der Landschaft sollte dabei durch die Abbildung der Landschaftskomponenten geschehen.

Auf Basis der zuvor gewonnenen Erkenntnisse kann nun abschließend folgende Definition von „Landschaft“ im Kontext dieser Arbeit festgehalten werden:

> *„Eine Landschaft ist ein durch Subjekte ausgestaltetes und durch eine Dynamik gekennzeichnetes (komplexes) Konstrukt aus Landschaftselementen, -strukturen und -funktionen, das historisch gewachsen ist.“*

2.1.3 Systemlandschaft

Nachdem die Begriffe „System“ und „Landschaft“ definiert wurden, ist es nun notwendig, dem Kompositum „Systemlandschaft“ eine Bedeutung zu geben. Dafür werden „System“ und „Landschaft“ im Folgenden kombiniert und in einen informationstechnischen Kontext eingeordnet.

Die in Abschnitt 2.1.1 vorgestellte Definition eines Systems lässt sich gut mit Konzepten der Informatik in Einklang bringen. Im Wesentlichen kann unter einem System im Kontext dieser Arbeit ein Informations- und Kommunikationssystem (**IKS**) verstanden werden. Ein System im allgemeinen Sinne besteht, wie zuvor gezeigt, aus *Komponenten*. Bei IKS sind diese Komponenten Menschen, Maschinen und Anwendungen [Gab08]. Diese Komponenten stehen untereinander in Beziehung, um ihre Aufgabe der Leistungserbringung beziehungsweise Leistungsunterstützung zu erfüllen. Sie verfügen über Operationen und sind in eine Umgebung, ihre Umwelt, eingebettet.

Folglich ist ein IKS ein soziotechnisches System, das die betriebliche Leistungserstellung durch die Erfüllung von Aufgaben mit Informations- und Kommunikationsbezug unterstützen soll [WKW94]. Derartige Systeme lassen sich in den Dimensionen *Mensch*, *Aufgabe* und *Technik* beschreiben [HHR04], die untereinander in Beziehung stehen [Gab08]. Dieses Mensch-Aufgabe-Technik-Schema ist in Abbildung 2.2 aufgeführt.

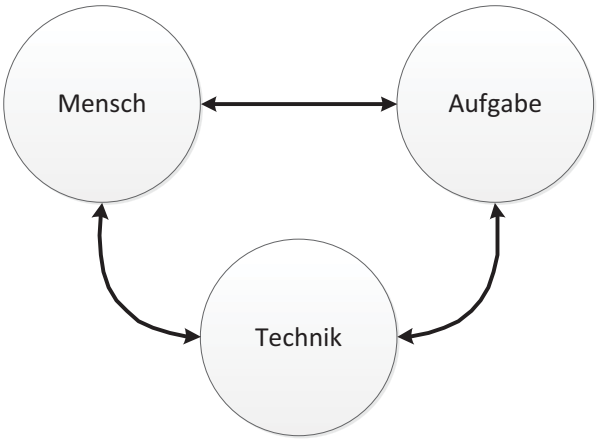

Abbildung 2.2: Dimensionen eines Informations- und Kommunikationssystems

Nach Heinrich et al. sind diese Dimensionen wie folgt charakterisiert (vergleiche [HHR04]):

- Der *Mensch* stellt eine wesentliche Komponente eines IKS dar, die in der Wirtschaftsinformatik jedoch häufig nicht oder nur kaum beachtet wird. Die Relevanz des Menschen ist dabei jedoch offensichtlich, denn ein IKS ist *„letztendlich auf die Absichten des Menschen zurückzuführen“* [HHR04, S. 14]. Der Mensch ist folglich als ein die Systemkonstruktion stark beeinflussender Faktor zu sehen.
- Die *Aufgabe* leitet sich aus den Geschäftsprozessen einer Organisation beziehungsweise deren Gesamtaufgabe ab. Aufgaben werden dabei verschiedenen Aufgabenkategorien/-typen zugeordnet und werden durch sechs verschiedene Merkmale charakterisiert:
  1. Verrichtung der Aufgabe (Verrichtungsprinzip),
  2. das Objekt, an dem die Aufgabe verrichtet wird,
  3. der Aufgabenträger, der die Aufgabe verrichtet,
  4. die Hilfsmittel zur Verrichtung durch den Aufgabenträger,
  5. der Ort (nach Heinrich et al. „Raum“) der Erfüllung sowie
  6. die Zeit der Aufgabenerfüllung.
- Unter *Technik* sind Hardware und Software und deren Zusammenspiel zu verstehen.

Grabski und Krüger wie auch Heinrich et al. sehen dabei nicht nur die einzelnen Dimensionen als wesentlich für Forschungsfragen an, sondern vor allem auch deren Beziehungen zueinander [GK08, HHR04]:

- *Mensch/Aufgabe-Beziehungen* (Management) behandeln beispielsweise das Maß der optimalen Zuordnung von Aufgaben zu Menschen.

- *Mensch/Technik-Beziehungen* (Technologie) bezeichnen die *„Konstruktion, Analyse, Anwendung und Veränderung von Technik“* [GK08, S. 1884].
- *Aufgabe/Technik-Beziehungen* (Automatisierung) bezeichnen die Verrichtung von Aufgaben, die vormals durch Menschen erledigt wurden und nun durch Technik erfüllt werden.

Wie in Abschnitt 2.1.2 gesehen, besteht eine Landschaft aus Landschaftselementen, -strukturen und -funktionen. Bei den *Landschaftselementen* handelt es sich im Kontext von IT um die zuvor beschriebenen Informations- und Kommunikationssysteme. Unter der *Landschaftsstruktur* ist folglich die Verbindung der einzelnen Systeme untereinander zu verstehen. Diese Vernetzung der einzelnen IKS wird durch die *IT-Infrastruktur* realisiert. Nach Patig et al. besteht die IT-Infrastruktur aus den in Abbildung 2.3 aufgeführten Komponenten [PHZ09]:

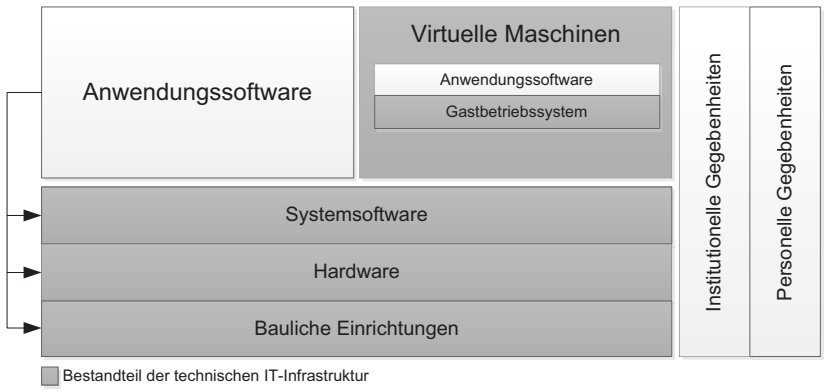

Abbildung 2.3: Bestandteile der IT-Infrastruktur (in Anlehnung an [PHZ09])

Es werden dabei verschiedene Sichten auf eine IT-Infrastruktur (vergleiche [PHZ09]) unterschieden, denen die einzelnen Elemente der Infrastruktur zugeordnet werden können:

- Die *IT-Infrastruktur aus technischer Sicht* umfasst alle Arten von *„Hardware, Software sowie [bauliche] Einrichtungen für den Betrieb von (Anwendungs-) Software“* [PHZ09]. Zur Hardware zählen Rechentechnik, Netzwerktechnik und Peripheriegeräte sowie weitere Komponenten für den Betrieb der Hardware. Zudem sind Systemsoftware und bauliche Einrichtungen Bestandteil der technischen IT-Infrastruktur. Die Komponenten der technischen Sicht auf die Infrastruktur sind in Abbildung 2.3 grau hinterlegt.
- Die *IT-Infrastruktur aus Sicht des Informationsmanagements* (auch als „Informationsinfrastruktur“ bezeichnet) erweitert die IT-Infrastruktur um *„um institutionelle und personelle Gegebenheiten“* [PHZ09]. Diese umfassen diverse Einflüsse auf die IT-Infrastruktur, wie beispielsweise Gesetze und Normen oder Anzahl der Mitarbeiter und deren Wissen.

Die Landschaftselemente in einer Systemlandschaft werden folglich durch IKS repräsentiert, die durch die IT-Infrastruktur betrieben werden.

Unter der *Landschaftsfunktion* wird im Kontext dieser Arbeit die Gesamtaufgabe des Unternehmens verstanden. Diese Gesamtaufgabe kann in kleine Teilaufgaben zerlegt werden, die durch einzelne IKS verrichtet beziehungsweise unterstützt werden (Dimension „Aufgabe“ eines IKS).

Neben dem Aufbau einer Landschaft, wie er zuvor in Abschnitt 2.1.2 dargestellt wurde, gelten die in der generischen Definition von „Landschaft“ aufgeführten Charakteristika ebenso für Systemlandschaften (vergleiche [GGH+07, GK08, Mat08]): Sie sind *durch Menschen gestaltet* worden und *mit der Zeit gewachsen*. Diese Veränderungen über einen längeren Zeitraum kennzeichnen die *Dynamik* einer Systemlandschaft. Diese wird auch zukünftig zu einer Veränderung des Landschaftsbildes führen. Ein Fakt, der nicht direkt durch die allgemeine Definition von „Landschaft“ abgedeckt wird, ist die *Komplexität* von Systemlandschaften. Diese Komplexität, die in der betrieblichen Praxis häufig vorherrscht [GGH+07], ist vor allem durch die komplexen Methoden zur Integration der einzelnen Systeme innerhalb der Systemlandschaft bedingt. Die Komplexität wird zudem und vor allem auch durch die *Heterogenität* der IKS und der ihnen zugrundeliegenden Infrastruktur bedingt. Die Heterogenität lässt sich vor allem auf die technologische Entwicklung zurückführen, die sich wiederum auf dem historischen Wachstum beruht.

Auf Basis der vorhergegangenen Untersuchung kann der Begriff Systemlandschaft demnach wie folgt definiert werden:

> *„Eine Systemlandschaft ist ein komplexes, heterogenes Konstrukt aus untereinander integrierten Informations- und Kommunikationssystemen, das in den Dimensionen Mensch, Aufgabe und Technik beschrieben wird. Sie ist historisch gewachsen und unterliegt einer veränderlichen Dynamik, die ihre Struktur immer wieder variiert.“*

Neben den bereits in die Definition eingeflossenen Charakteristika zeichnet sich eine Systemlandschaft durch weitere Eigenschaften aus:

- Eine Systemlandschaft ist *kein Selbstzweck*, sondern *„dient der Umsetzung und Unterstützung des Geschäfts eines Unternehmens“* [HVH06, S. 396].
- Die Komponenten einer Systemlandschaft weisen häufig eine *hohe Kopplung* auf („Spaghetti-Integration“) [GGH+07].
- Eine *langfristige Perspektive* muss vorhanden sein, um Ziele für die Zukunft definieren zu können und somit ein planerisches Vorgehen bei der Entwicklung der Systemlandschaft zu ermöglichen [Mat08].
- Da sich die Perspektive und folglich die mit der Systemlandschaft verbundenen Ziele mit der Zeit *wandeln*, bleiben diese unerreichbar [Mat08].
- Systemlandschaften enthalten *hohe Funktions- und Datenredundanzen*. Ein Teil dieser Redundanzen ist zwar unvermeidbar, jedoch Großteils ungewollt [Sie07].

Anhand der vorausgegangenen Vorstellung von Systemlandschaften kann festgehalten werden, dass auch die anderen Dimensionen aufgrund ihrer Bedeutung im Metamodell zumindest berücksichtigt werden sollten. Da der Faktor Mensch, wie zuvor dargelegt, dabei eine wesentliche Rolle einnimmt, erscheint es sinnvoll, diesen hinsichtlich seiner Interaktion mit der Systemlandschaft zu berücksichtigen, weil sein Einfluss auf die Modellbildung nur schwer einbezogen werden kann. Hinsichtlich der Dimension „Aufgabe“ erscheint es sinnvoll, Geschäftsprozesse bei der Abbildung von Systemlandschaften einzubeziehen, da sich die Aufgabe eines Unternehmens aus dessen Geschäftsprozessen ableitet.

2.2 Modell

Ein Modell ist das Ergebnis des Vorgangs der Modellbildung, bei dem Systeme auf Modelle abgebildet werden [Frö09]. Der Terminus „Modell“ wird in der Wissenschaft jedoch nicht einheitlich verwendet und ist auch in der Wirtschaftsinformatik häufig Gegenstand von Diskussionen [vB03, Tho05]. Im Allgemeinen können jedoch vor allem zwei Begriffsbedeutungen herausgestellt werden [Zsc95]: Zum einen ist dies der axiomatische Modellbegriff [Tar77], zum anderen die allgemeine Modelltheorie [Sta73] nach Stachowiak. Der axiomatische Modellbegriff ist in der Wirtschaftsinformatik von geringer Bedeutung, da in der Wirtschaftsinformatik vor allem realwissenschaftliche Methoden relevant sind.

Um nun ein einheitliches Verständnis des Begriffs „Modell“ im Kontext dieser Arbeit zu gewährleisten, wird im Folgenden eine Arbeitsdefinition aufgestellt. Dazu wird in Abschnitt 2.2.1 anhand der allgemeinen Modelltheorie zunächst erläutert, was sich hinter dem Begriff „Modell“ verbirgt. Im Anschluss dran wird in Abschnitt 2.2.2 darauf eingegangen, welche Ausprägungen des Modellbegriffs in der Wirtschaftsinformatik diskutiert werden. Darauf aufbauend erfolgt in Abschnitt 2.2.3 die Aufstellung einer Arbeitsdefinition, bevor in Abschnitt 2.2.4 abschließend auf Metamodelle eingegangen wird.

2.2.1 Allgemeiner Modellbegriff

Nach Stachowiak handelt es sich bei einem Modell um eine Repräsentation eines *Originals* für ein bestimmtes *Subjekt* bezüglich eines bestimmten *Zwecks* innerhalb einer bestimmten *Zeitspanne* [Sta73]. Abbildung 2.4 illustriert diesen Sachverhalt.

Stachowiaks Definition eines Modells ist auf die in seiner Begriffsanalyse identifizierten drei Hauptmerkmale des allgemeinen Modellbegriffs zurückzuführen (vergleiche nachfolgend [Sta73]):

1. *Abbildungsmerkmal*: Bei Modellen handelt es sich immer um Repräsentationen von natürlichen oder künstlichen Originalen. Modelle können aber wiederum andere Modelle repräsentieren.

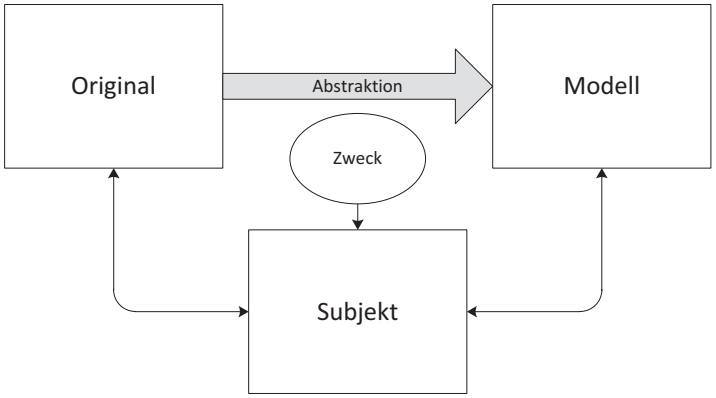

Abbildung 2.4: Modell in Anlehnung an [Sta73]

2. *Verkürzungsmerkmal*: Modelle erfassen Originale in der Regel nicht in ihrer Gesamtheit, sondern zeigen immer nur die für das Subjekt relevanten Teile des Originals. Das führt dazu, dass die abzubildenden Originaleigenschaften nie ganzheitlich bestimmt werden können. Folglich impliziert dieses Merkmal, dass zu einem Original mehrere verschiedene Modelle existieren.

3. *Pragmatisches Merkmal*: Modelle sind in dreierlei Hinsicht pragmatisch: Sie erfüllen einen bestimmten *Zweck* für einen bestimmtes *Subjekt* in einer bestimmten *Zeit*.

Abbildung 2.5 zeigt das Modellverständnis der allgemeinen Modelltheorie unter Berücksichtigung der drei Hauptmerkmale.

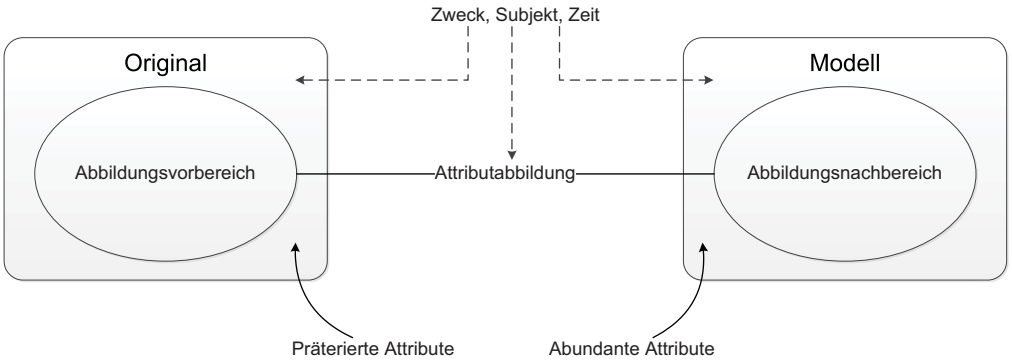

Abbildung 2.5: Original-Modell-Abbildung

Demnach werden die Attribute des Originals auf den sogenannten abbildungsrelevanten Abbildungsvorbereich reduziert und durch die Attributabbildung bijektiv auf das Modell in den Abbildungsnachbereich abgebildet [Sta73]. Dabei entstehen zwei Klassen von Attributen: *präterierte* sowie *abundante* Attribute. Präterierte Attribute sind die Attribute des Originals, die nicht auf das Modell abgebildet werden. Als abundante Attribute werden die Attribute bezeichnet, die sich zwar Bestandteil des Modells sind, sich aber nicht im Original wiederfinden.

2.2.2 Modelle in der Wirtschaftsinformatik

Innerhalb der Wirtschaftsinformatik sind insbesondere zwei Sichtweisen (vergleiche [vB03]) auf den allgemeinen Modellbegriff relevant: Die *Abbildungsorientierung* sowie die *Konstruktionsorientierung*.
Die Abbildungsorientierung stellt das Abbildungsmerkmal der allgemeinen Modelltheorie in den Vordergrund. Der Fokus der Abbildungsorientierung liegt dabei auf der Abbildung zwischen Original und Modell. Für abbildungsorientierte Modelldefinitionen ist vor allem die Art der Abbildung von Bedeutung [Tho05]. Sie kann entweder homomorph oder isomorph sein. Thomas äußert die Vermutung, dass das abbildungsorientierte Modellverständnis das in der Wirtschaftsinformatik am weitesten verbreitete sei [Tho05]. Vom Brocke teilt diese Ansicht, da er Modelle als *„immaterielle und abstrakte Abbilder der Realität für Zwecke eines Subjekts“* [vB03, S. 10] sieht. Im Gegensatz zur Definition nach Stachowiak unterstellt die Abbildungsorientierung also explizit einen Realitätsbezug. Abbildung 2.6 verdeutlicht die Begriffe der Abbildungsorientierung und deren Relationen (vergleiche dafür [Tho05, RS02, vB03]).

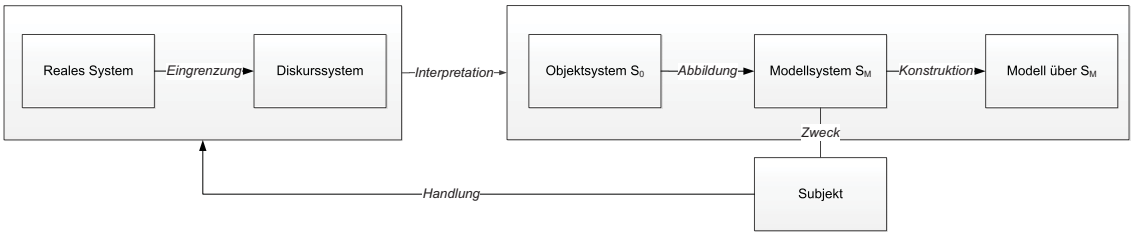

Abbildung 2.6: Der abbildungsorientierte Modellbegriff

Grundlegend unterscheidet die Abbildungsorientierung zwischen *realem System* und *Modellsystem*. Der realweltliche Abschnitt, der modelliert werden soll, wird als *Diskurssystem* (oder auch als *Diskurswelt*) bezeichnet. Die Interpretation dieses Diskurssystems wird auf ein *Objektsystem* S_0 abgebildet. Da diese Abbildung jedoch keine hinreichende Vereinfachung des Diskurssystems darstellt, wird eine Komplexitätsreduktion durch die Überführung des Objektsystems S_0 in das *Modellsystem* S_M vorgenommen. Die Relation *Zweck* macht deutlich, dass ein Modell immer mit einer bestimmten Intention (der des *Subjekts*) erstellt wird. Aus dem Modellsystem wird abschließend das *Modell* konstruiert. Die Relation *Handlung* bezeichnet den Vorgang der *Modellbildung*. In [RS02] und [Tho05] wird davon ausgegangen, dass aus dem Modellsystem kein Modell erstellt, sondern ein *Metamodell* zur Konstruktion des Modellsystems verwendet wird.

Bei der Konstruktionsorientierung (vergleiche nachfolgend [vB03, Tho05]) wird, anders als bei der Abbildungsorientierung, davon ausgegangen, dass die Realität immer subjektgebunden und damit nicht objektiv ist. Ein Unterschied zur Abbildungsorientierung besteht auch dahingehend, dass die Art der Abbildung nicht beziehungsweise weniger von Bedeutung ist. Statt der Forderung nach einem Homomorphismus beziehungsweise Isomorphismus rückt bei der Konstruktionsorientierung die Betrachtung von Problemdefinitionen in den Vordergrund. Unter Problemen werden dabei *„Abweichungen zwischen Erreichtem und Erwünschtem“* [Tho05, S. 18] verstanden. Nach Thomas ist der *„Prozess der Modellbildung [...] daher ein Strukturgebungsprozess“* [Tho05, S. 18], bei dem das Ziel jedoch nicht die Rekonstruktion von Strukturkomplexen ist. Vielmehr

ist die eigentliche Konstruktion als „gedankliche Leistung“ zu verstehen. Dies geht vor allem auf die Modelldefinition von Schütte zurück, die im Rahmen der Wirtschaftsinformatik maßgeblich für die Begriffsprägung verantwortlich ist. Er beschreibt ein Modell als *„das Ergebnis einer Konstruktion eines Modellierers, der für Modellnutzer eine Repräsentation eines Originals zu einer Zeit als relevant mit Hilfe einer Sprache deklariert“* [Sch98, S. 59]. Abbildung 2.7 illustriert dieses Modellverständnis:

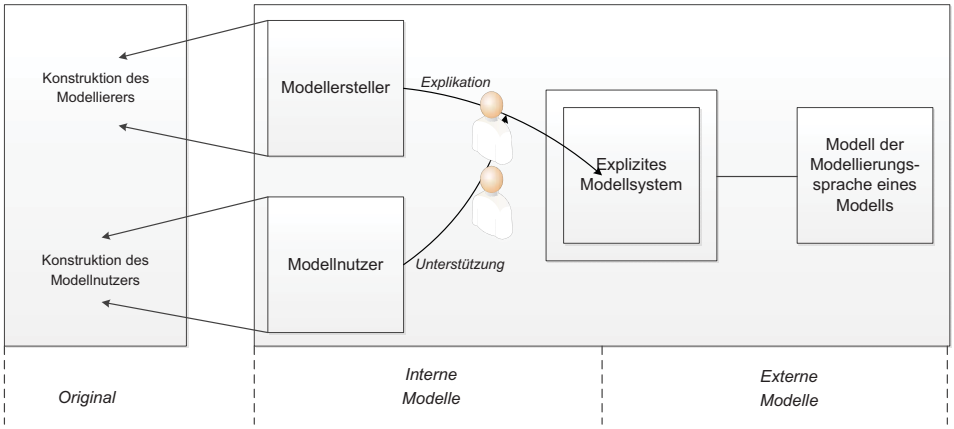

Abbildung 2.7: Der konstruktionsorientierte Modellbegriff

Die Subjekte *Modellersteller* und *Modellnutzer* bringen gleichermaßen ihr subjektives Verständnis des *Originals* in Form von *Konstruktionen* in die Modellierung ein. Das Original kann dabei ein beliebiges zu modellierendes Problem sein. Unter Nutzung einer konkreten Modellierungssprache führt der Modellersteller eine Explizierung der Konstruktion durch. Dabei wird von ihm insbesondere die Konstruktion des Modellnutzers berücksichtigt, da dieser den Zweck des Modells vorgibt. Das *explizite Modellsystem* stellt das Ergebnis der Modellierung dar.

2.2.3 Abschließende Definition

Wie im vorherigen Abschnitt aufgezeigt wurde, bestehen augenscheinlich nur marginale Unterschiede hinsichtlich des Verständnisses, was ein Modell ausmacht. Alle Definitionen gehen davon aus, dass ein Modell ein System für ein Subjekt abbildet. Unterschiede bestehen jedoch durchaus im Detail. Wie Thomas aufzeigt, betrifft dies vor allem die Modellbildung. Um nun ein einheitliches Begriffsverständnis im Rahmen dieser Arbeit zu gewährleisten, müssen die vorgestellten Definitionen in Einklang gebracht werden.

Im Rahmen einer Analyse der vorgestellten Modellverständnisse kommt Thomas zu dem Schluss, dass die Abbildungsorientierung zu limitiert ist. Dies begründet er, wie beispielsweise auch [vB03] damit, dass der explizite Realitätsbezug zu großen Einfluss auf den Vorgang der Modellbildung nimmt. Da der zu beschreibende Realitäts-Ausschnitt vom Modellierer lediglich erfasst und wiedergegeben werden muss, beschränke sich die Modellbildung auf *„objektives Wahrnehmen und logisches Schließen“* [Tho05, S. 21].

Die allgemeine Modelltheorie nach Stachowiak vermeide dies zwar, stehe jedoch ebenso *„in der Tradition eines abbildungstheoretischen Modellverständnisses“* [Tho05, S. 24]. Dies zeigt sich besonders durch das Abbildungsmerkmal und weiterhin auch durch das Verkürzungsmerkmal deutlich. Stachowiak relativiert den Abbildungsgedanken

zwar dahingehend, dass er die Realität als konstruiert ansieht. Jedoch bestehe nach wie vor das Problem, dass Modelle als reine Abbilder der Realität verstanden werden können [Tho05].

Thomas folgt daher der Konstruktionsorientierung. Für eine allgemeingültige Begriffsdefinition ist seine Argumentation zwar schlüssig, im Rahmen dieser Arbeit ist es jedoch sinnvoll, auch die Abbildungsorientierung zu berücksichtigen. Insbesondere vor dem Hintergrund, dass realweltliche Systeme den Betrachtungsgegenstand des im Rahmen dieser Arbeit zu entwickelnden Modellierungsansatzes darstellen. Der Definition nach [Tho05] kann daher nur bedingt gefolgt werden.

Auf Grundlage von Stachowiaks allgemeiner Modelltheorie unter dem Aspekt der Konstruktionsorientierung definiert Thomas ein Modell als *„eine durch einen Konstruktionsprozess gestaltete, zweckrelevante Repräsentation eines Objekts“* [Tho05, S. 25]. Als Konstruktionsprozess versteht er dabei einen Prozess, in dessen Verlauf sich Entwürfe herausbilden. Für die Definition im Rahmen dieser Arbeit werden jedoch zusätzliche Punkte als wesentlich erachtet:

- Betrachtungsgegenstand des im Rahmen dieser Arbeit entstehenden Metamodells stellen die in Abschnitt 2.1 vorgestellten Systemlandschaften dar. Bei diesen handelt es sich um realweltliche Objekte. Somit ist für eine Begriffsdefinition von „Modell“ ein *Realitätsbezug* notwendig. Die Verwendung der Definition von Thomas ist dabei dennoch zweckmäßig, da diese die Leistung des Modellentwicklers im Rahmen des Konstruktionsprozesses hervorhebt und die Modellierung nicht als einfache Abbildung realweltlicher Strukturen abtut.

- Thomas spricht im Rahmen seiner Definition von einem „Objekt“. Er weist zwar darauf hin, dass es sich dabei um Objekte im Sinne von [Sta73] handelt. Jedoch kann diese Bezeichnung irreführend sein, da der Singular auf ein einziges zu modellierendes Objekt hindeutet. Insbesondere im Rahmen dieser Arbeit sind jedoch nicht einzelne Objekte, sondern vielmehr eine Menge von Objekten und deren Relationen relevant. Daher ist es sinnvoll, von einem modellierenden *System* denn von einem Objekt zu sprechen. Damit wird der möglichen Komplexität einer Modellierung entsprochen.

Auf Basis dieser Punkte kann somit abschließend folgende Definition festgehalten werden:

> *„Ein Modell ist eine durch einen Konstruktionsprozess gestaltete, zweckrelevante Repräsentation eines realweltlichen Systems.“*

2.2.4 Metamodelle

Im Bereich der Modellierung sind, neben Modellen, vor allem auch Metamodelle von Bedeutung. Ein Modell ist, wie zuvor gezeigt, zweckrelevante Repräsentation eines realweltlichen Systems. Es entsteht dabei durch einen Konstruktionsprozess. Um nun zu einem Modell zu gelangen, wird folglich eine Modellierungssprache benötigt. Ist jedoch nicht ein realweltliches System Gegenstand des Konstruktionsprozesses, sondern sind *„Modelle und Modellbildung selbst [...] Gegenstand der Modellierung“* [Str10], wird nicht mehr von Modellen, sondern von einem *Metamodell* gesprochen. Wie Strahringer in

[Str10] darlegt, handelt es sich bei einer Metasprache (beziehungsweise einem Metamodell) um eine Sprache, die in Bezug auf eine andere Sprache (die Objektsprache) zu deren Untersuchung genutzt wird. Eine Metasprache befindet sich folglich auf einer anderen semantischen Stufe als die Objektsprache.

Nach dem von Strahringer vorgestellten sprachbasiertem Metamodellbegriff handelt es sich bei einem Modell um ein Metamodell bezüglich eines anderen Modells, *„wenn es ein Beschreibungsmodell der Sprache, in der dieses Modell formuliert ist, darstellt“* [Str10].

2.3 Kleine und mittlere Unternehmen

Das im Rahmen dieser Arbeit zu entwickelnde Metamodell bezieht sich auf kleine und mittlere Unternehmen. Abschnitt 2.3.1 legt daher zunächst dar, was genau unter KMU zu verstehen ist. Im Anschluss wird in Abschnitt 2.3.2 deren volkswirtschaftliche Bedeutung in Deutschland aufgezeigt. Abschließend werden in Abschnitt 2.3.3 die Besonderheiten von KMU hinsichtlich des Einsatzes von IT dargelegt.

2.3.1 Definition

Zur Abgrenzung von KMU zu großen Unternehmen können *quantitative* und *qualitative* Merkmale herangezogen werden. Quantitative Merkmale beziehen sich auf messbare Größen bezüglich Unternehmen, wie beispielsweise deren Jahresumsatz. Hingegen beschreiben qualitative Merkmale die Besonderheiten von KMU hinsichtlich deren Abgrenzung zu großen Unternehmen, dies jedoch nicht im Sinne messbarer Größen. Diese qualitativen Merkmale werden als notwendig erachtet, da eine rein quantitative Erfassung die *„bestehende organisatorische und strukturelle Heterogenität, die Dynamik und die Vielschichtigkeit der Wesensmerkmale“* [Rud09, S. 64] von KMU nicht umfassend abbilden kann.

Quantitative Definitionen legen beispielsweise das Institut für Mittelstandsforschung in Bonn[1] (**IfM**) oder die Europäische Union (**EU**) vor. Die in Deutschland gebräuchlichste (quantitative) Eingrenzung stammt vom IfM [Goe08]. Diese Definition unterscheidet, wie auch die der EU, nach den Kriterien *Unternehmensgröße*, *Zahl der Beschäftigten* und *Jahresumsatz in* € und ist in Tabelle 2.1 aufgeführt. Den Zusammenschluss von kleinen und mittleren Unternehmen fasst das IfM unter dem Begriff „Mittelstand“ zusammen.

Unternehmensgröße	**Beschäftigte**	**Jahresumsatz (€)**
klein	bis 9	bis 1 Mio.
mittel	10 bis 499	1 Mio. bis 50 Mio.
groß	über 500	über 50 Mio.

Tabelle 2.1: Definition von kleinen und mittleren Unternehmen nach dem Institut für Mittelstandsforschung (Quelle: [Ins02])

Neben der Definition des IfM ist auch die Definition der EU für KMU relevant, da diese beispielsweise die Grundlage für die Verteilung von EU-Fördergeldern bildet

[1] http://www.ifm-bonn.org/

[Rud09]. Neben den bereits bei der Definition nach dem IfM erwähnten Kriterien kommt bei der Einteilung nach der EU die Jahresbilanz als weiteres Kriterium hinzu. Zudem ist es für die Einteilung nach der EU wesentlich, dass sich ein Unternehmen zu nicht mehr als 25% in Besitz eines anderen Unternehmens befindet. Tabelle 2.2 zeigt die Einteilung nach der EU auf.

Unternehmensgröße	**Beschäftigte**	**Jahresumsatz (€)**	**Jahresbilanz (€)**
kleinst	bis 9	bis 2 Mio.	bis 2 Mio.
klein	10 bis 49	bis 10 Mio.	bis 10 Mio.
mittel	50 bis 249	bis 50 Mio.	bis 43 Mio.

Tabelle 2.2: Definition von kleinen und mittleren Unternehmen nach der Europäischen Union (Quelle: [Kom06])

Daneben sieht es beispielsweise Goeke als wichtig an, dass neben den quantitativen auch qualitative Merkmale betrachtet werden, da der deutsche Mittelstand bei der alleinigen Betrachtung quantitativer Größen unzureichend charakterisiert wäre [Goe08]. Derartige Merkmale werden beispielsweise in [Goe08, Rud09] aufgezeigt. Einen sehr ausführlichen Katalog qualitativer Merkmale stellt Pfohl in [Pfo97] vor. Da eine qualitative Betrachtung im Kontext dieser Arbeit von nachrangiger Bedeutung ist, wird an dieser Stelle auf eine umfassende Vorstellung dieser Merkmale verzichtet. Tabelle 2.3 zeigt daher nur einen Ausschnitt von Pfohls Katalog und soll lediglich verdeutlichen, dass zwischen großen Unternehmen und KMU wesentliche Unterschiede bestehen, die nicht quantifiziert werden können.

Klein- und Mittelbetriebe	**Großbetriebe**
Unternehmensführung	
Eigentümer-Unternehmer	Manager
Führungspotential nicht austauschbar	Führungspotential austauschbar
Kaum Gruppenentscheidungen	Häufig Gruppenentscheidungen
Organisation	
Funktionshäufung	Arbeitsteilung
Hohe Flexibilität	Geringe Flexibilität
Kurze direkte Informationswege	Vorgeschriebene Informationswege

Tabelle 2.3: Auszug qualitativer Merkmale von kleinen und mittleren Unternehmen nach Pfohl (Quelle: [Pfo97])

Abschließende Definition

Da die Definition des IfM in Deutschland besonders weit verbreitet ist, wird auch im Rahmen dieser Arbeit dieser Definition gefolgt (siehe Tabelle 2.2). Wie zuvor gesehen, ist dabei aber stets zu berücksichtigen, dass sich KMU nicht nur hinsichtlich messbarer Größen von großen Unternehmen unterscheiden. Vielmehr sind es vor allem die qualitativen Merkmale, die eine Differenzierung erst ermöglichen (siehe dafür exemplarisch Tabelle 2.3 beziehungsweise in ausführlicher Fassung [Pfo97] oder auch [Goe08, Rud09]).

2.3.2 Volkswirtschaftliche Bedeutung

Neben der Frage, was genau KMU charakterisiert, ist vor allem auch deren volkswirtschaftliche Bedeutung von Interesse: Tatsächlich prägen KMU die deutsche Wirtschaft maßgeblich. Abbildung 2.8 verdeutlicht diesen Umstand anhand wichtiger Schlüsselzahlen des IfM.

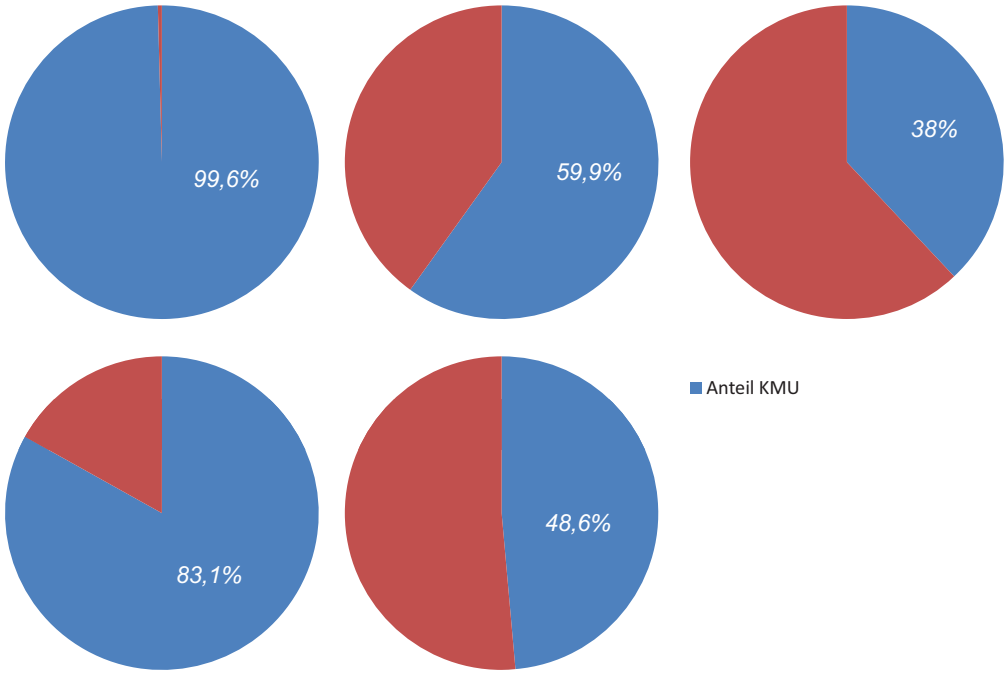

Abbildung 2.8: Schlüsselzahlen zu kleinen und mittleren Unternehmen (Quelle: [Ins10])

Im Jahr 2008 waren in Deutschland insgesamt 3.730.000 Unternehmen tätig, die je mindestens 17.500 € umgesetzt haben oder wenigstens einen sozialversicherungspflichtig Beschäftigten unterhielten. Dabei macht der Anteil kleiner und mittlerer Unternehmen 99,6% der gesamten Unternehmen in Deutschland aus. Die KMU vereinen dabei 38% der Unternehmensumsätze (insgesamt ca. 2.039,27 Mrd. €) auf sich.

Deutlich sind auch die Zahlen hinsichtlich der Beschäftigung: Nach den Statistiken des IfM sind knapp 60% der sozialversicherungspflichtig Beschäftigten in KMU angestellt. Beispielsweise geht Goeke in Anlehnung an eine IfM-Studie darauf ein, dass in den Jahren 2003 bis 2005 *„der Beschäftigungszuwachs in Deutschland ausschließlich durch die mittelständischen Unternehmen getragen war“* [Goe08, S. 14], wohingegen Großunternehmen im gleichen Zeitraum die Anzahl ihrer Beschäftigten abbauten (für genaue Angaben siehe [Ins09] oder verkürzt [Goe08]).

Eine besondere Bedeutung kommt den KMU auch hinsichtlich der Beschäftigung von Auszubildenden zu. So absolvieren mehr als 80% aller Auszubildenden ihre Ausbildung in kleinen und mittleren Unternehmen. Dieser Wert unterstreicht nochmals die hohe Bedeutung von KMU für die deutsche Volkswirtschaft.

2.3.3 Rolle und Bedeutung von IT

Informationstechnologie gewinnt in immer mehr Unternehmensbereichen an Bedeutung und ist bereits heute entscheidend für die Ausführung von Geschäftsprozessen [Rud09]. Dies gilt für KMU wie auch für Großunternehmen. Im Gegensatz zu Großunternehmen müssen sich KMU jedoch mit anderen Rahmenbedingungen auseinandersetzen (vergleiche [Mei04, Rud09]). Es zeigt sich jedoch, dass häufig eine hohe Divergenz zwischen der eigentlichen Bedeutung der IT für ein Unternehmen und dem ihr tatsächlich entgegengebrachten Stellenwert besteht [Rud09].

Untersuchungen zeigen, dass dem Einsatz von IT in KMU hinsichtlich der Erreichung der Unternehmensziele eine hohe bis sehr hohe Bedeutung zugesprochen werden kann [Rud09]. Gleichzeitig sehen sich KMU mit einem hohen Kostendruck konfrontiert, da sie in der Regel nur über ein begrenztes IT-Budget, das zudem rückläufig ist, verfügen. Das limitierte Budget kann dabei als markantes Merkmal von KMU betrachtet werden [STS04]. Neben den finanziellen Schwierigkeiten sind es vor allem auch personelle Engpässe, die KMU bei der Durchführung von IT-Projekten immer wieder zu schaffen machen [Bis09]. Die Situation von KMU kann somit als problematisch bezeichnet werden, da Investitionen nur in einem begrenzten Rahmen möglich sind.

Gleichzeitig sehen sich KMU jedoch mit ähnlichen Herausforderungen konfrontiert wie große Unternehmen, verfügen dabei allerdings, wie bereits ausgeführt, über deutlich weniger finanzielle und personelle Ressourcen [STS04]. Zudem mangelt es KMU, im Gegensatz zu großen Unternehmen, häufig an einer eigenen IT-Abteilung, sodass das Management der IT zur nebensächlichen Aufgabe der Geschäftsführung verkommt [Mei04]. Erschwerend kommt hinzu, dass Unternehmen nicht mehr als abgeschlossene Einheiten betrachtet werden, sondern die Interaktion mit Partnern und damit deren Integration eine immer wichtigere Rolle spielt [Die06].

Vor allem die in den letzten Jahren immer weiter steigende Komplexität von IT-Systemen stellt KMU vor Probleme. In engem Zusammenhang damit stehen steigende Anforderungen an eine Standardisierung der IT bei gleichzeitiger Flexibilisierung der IT-Infrastruktur [Rud09]. Der Einsatz von Standardsoftware wird dabei häufig auch durch Großunternehmen an KMU herangetragen, die als Partner auftreten [Bis09]. Insbesondere dem Management der IT-Infrastruktur wird eine besondere Bedeutung zugestanden, da *„diese häufig organisch gewachsen [ist] und sich daraus meist heterogene IT-Landschaften entwickelt haben, die schnell intransparent und mitunter ineffizient werden können“* [Rud09, S. 74]. Rudolph sieht parallel dazu weitere bedeutsame Herausforderungen für KMU, beispielsweise hinsichtlich der Entscheidungen des IT-Outsourcings und den daraus resultierenden Effekten auf das Unternehmen.

2.4 Cloud Computing

Der Begriff *Cloud Computing* ist in der IT derzeit allgegenwärtig und wird noch immer vielschichtig diskutiert. Dennoch hat sich bislang keine einheitliche Definition des Begriffs etabliert - vielmehr entstand eine Vielzahl häufig heterogener Definitionen (vergleiche exemplarisch [VRMCL08], [BLRK09]). Das gemeinsame Begriffsverständnis hat sich in jüngeren Diskussionen zwar gefestigt [BLRK09], von einer einheitlichen Definition kann aber bis heute nicht gesprochen werden.

Generell wird unter Cloud Computing eine Möglichkeit verstanden, IT-Leistungen über das Internet zu beziehen. Software und Hardware sind nicht mehr an einen einzelnen lokalen Computer gebunden, sondern werden zentral angeboten und bedarfsgerecht abgerechnet [PM10]. Die Verwaltung von Hardware und Software wird dabei vom Dienstleister übernommen. Der Nutzer kauft keine Hardware oder Anwendungen mehr, sondern bezieht lediglich die Dienstleistung über eine Art Mietmodell – ein eigenständiges Warten und Verwalten durch den Nutzer ist nicht mehr notwendig.

Eine Definition, die in der Literatur häufig als Diskussionsgrundlage dient (vergleiche [HRV11]) und daher auch im Rahmen dieser Arbeit verwendet wird, ist die Definition des National Institute of Standards and Technology (**NIST**):

> *„Cloud computing is a model for enabling convenient, on-demand network access to a shared pool of configurable computing resources (e.g., networks, servers, storage, applications, and services) that can be rapidly provisioned and released with minimal management effort or service provider interaction. This cloud model promotes availability and is composed of five essential characteristics, three service models, and four deployment models.“* [MG09, S. 6]

Weiterhin werden in [MG09] die wesentlichen Charakteristika und Ebenen des Cloud Computing sowie mögliche Bereitstellungsarten vorgestellt, was im Folgenden betrachtet wird.

Wesentliche Charakteristika

Die Definition des NIST unterscheidet fünf kennzeichnende Merkmale [MG09]:

- *On-demand Service* – Dienste werden automatisch nach Bedarf bereitgestellt. So kann beispielsweise beim Erreichen einer Kapazitätsgrenze zusätzlicher Speicherplatz einseitig durch den Nutzer alloziert werden.
- *Broad Network Access* – Dienste werden durch Standard-Mechanismen über ein Netzwerk bereitgestellt. Dadurch sind sie nicht an einen bestimmten Client gebunden und auf heterogenen Endgeräten verwendbar.
- *Resource Pooling* – Die Ressourcen des Anbieters liegen in einem Verbund vor, aus dem jeder Nutzer bedient wird („Multi-Tenant Modell“). Sie liegen in Form von physischen und virtuellen Komponenten vor, die je nach Bedarf der Nutzer zugeteilt werden.

- *Rapid Elasticity* – Die vorhandenen Dienste können schnell und elastisch zur Verfügung gestellt werden. Dies kann in einigen Fällen auch automatisch erfolgen, sodass ein Dienst selbstständig skaliert.

- *Measured Service* – Die Nutzung der Dienste und Ressourcen kann sowohl vom Nutzer als auch durch den Anbieter überwacht werden.

Als wesentlicher Unterschied zu anderen, dem Cloud Computing ähnlichen Konzepten, wie beispielsweise dem Application Service Providing (**ASP**), ist das Multi-Tenant Modell zu nennen [HRV11]. In den vielfältigen Definitionen zu Cloud Computing finden sich daneben noch weitere Merkmale, wie beispielsweise eine verbrauchsgerechte Abrechnung der bezogenen Leistungen oder das Vorhandensein einer Entwicklungsplattform (siehe [BLRK09]). Auf die Vorstellung weiterer Merkmale wird an dieser Stelle jedoch verzichtet, da weitere Merkmale keinesfalls in der Mehrzahl der Definitionen zu finden sind.

Ebenen

Cloud Computing unterscheidet generell zwischen den drei Ebenen *Infrastructure as a Service* (**IaaS**), *Platform as a Service* (**PaaS**) und *Software as a Service* (**SaaS**). Diese Ebenen werden im Folgenden vorgestellt (vergleiche nachfolgend [MG09, HWBH09]).

- *Infrastructure as a Service* bezeichnet die Bereitstellung virtueller IT-Ressourcen, wie beispielsweise Datenspeicher oder Rechenleistung. Diese Ressourcen werden einem Kunden als Dienstleistung zur Verfügung gestellt und können für beliebige Arten von Anwendungen auf Abruf bezogen werden. Ein wesentlicher Unterschied zu klassischen Hostingverfahren stellt die Skalierbarkeit entsprechend des aktuellen Bedarfs dar. Ein Beispiel für IaaS ist die *Amazon Elastic Compute Cloud*[2].

- *Platform as a Service* bezeichnet die Bereitstellung kompletter Infrastrukturen für Entwicklungs- und Betriebsumgebungen. Die Plattform bietet grundlegende Funktionalitäten in Bereichen wie Benutzerverwaltung oder Integration an. Die Bereitstellung von Basisfunktionalitäten vereinfacht die Entwicklung von Software dabei erheblich. Als Beispiel für eine PaaS-Lösung kann *Google App Engine*[3] aufgeführt werden.

- *Software as a Service* ist die Bereitstellung von Software als Dienstleistung über das Internet. Die Software wird nicht mehr durch den Kunden erworben, sondern in der Regel nutzungsabhängig über ein Mietmodell bezogen. Der Kunde muss sich dabei nicht um Installation oder Wartung der Software kümmern: genutzt wird die Software über den Browser, um andere Belange (beispielsweise Sicherheit, Updates, Verfügbarkeit) kümmert sich der SaaS-Anbieter. Ein Beispiel für eine SaaS-Lösung ist die Unternehmenssoftware *SAP Business ByDesign*®[4].

Einige Definitionen zählen noch weitere Ebenen auf, die Mehrheit bezieht sich jedoch auf die hier vorgestellten.

[2] http://aws.amazon.com/ec2/
[3] http://code.google.com/appengine/
[4] http://www.sap.com/germany/sme/solutions/businessmanagement/businessbydesign/index.epx

Bereitstellungsarten

Zur Umsetzung von Cloud Computing existieren verschiedene Ansätze. Diese Ansätze unterscheiden sich im Wesentlichen durch zwei Faktoren: *Wo* befindet sich die Cloud und *wer* übernimmt ihre Verwaltung? Im Folgenden werden vier verschiedene Ansätze kurz vorgestellt (vergleiche [MG09, HRV11]).

- In einer *Public Cloud* sind die Ressourcen der Cloud öffentlich zugänglich und werden durch einen Betreiber angeboten.
- Eine *Private Cloud* stellt keine öffentlich zugänglichen Ressourcen bereit, sondern beschränkt sich auf einzelne Unternehmen. Insbesondere bei der Arbeit mit sensiblen Daten wird häufig auf dieses Modell zurückgegriffen, da das Unternehmen seine Daten so selber verwalten kann.
- Bei einer *Community Cloud* stehen deren Ressourcen einer Gruppe von Unternehmen, die häufig ähnliche Interessen oder Ziele haben, zur Verfügung. Die Cloud wird dabei durch eines der Unternehmen oder durch einen Drittanbieter verwaltet. Die Beweggründe sind ähnlich zu denen einer Private Cloud (vor allem Datenschutz), wobei der Vorteil besteht, nicht alleine für den Betrieb der Cloud verantwortlich zu sein.
- Unter einer *Hybrid Cloud* ist ein Zusammenschluss von zwei oder mehr Arten von Clouds zu verstehen. Ein solcher Zusammenschluss kann beispielsweise sinnvoll sein, um Lastspitzen in der eigenen Private Cloud durch eine Public Cloud abzufangen.

Abbildung 2.9 veranschaulicht die zuvor erwähnten Ansätze und zeigt eine exemplarische Konstellation auf.

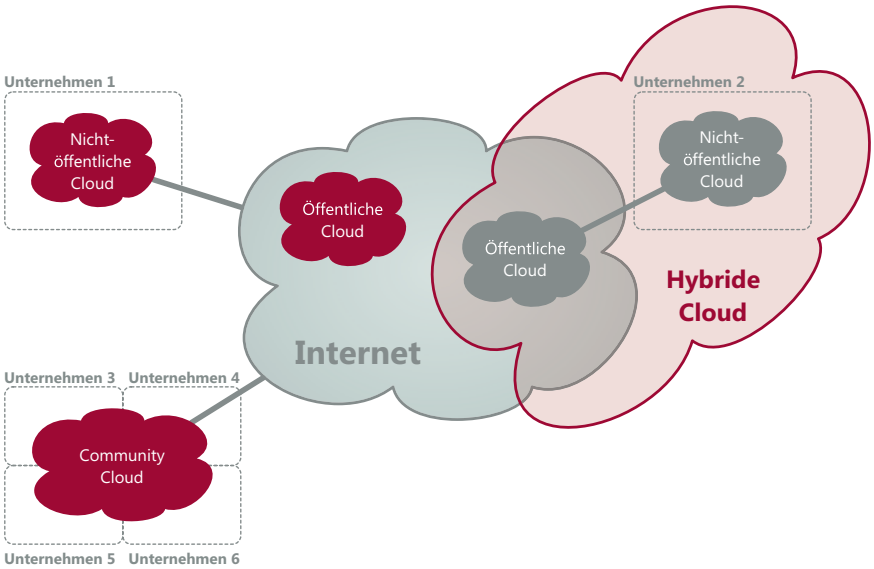

Abbildung 2.9: Betriebsmodelle für Cloud Computing (Quelle: [HRV11])

2.4.1 Serviceorientierte Architekturen

Im Kontext von Cloud Computing wird häufig auch von *Serviceorientierung* beziehungsweise von *serviceorientierten Architekturen* (**SOA**) gesprochen (vergleiche zum Beispiel [RHS05, ZE08, BKNT09, BHB10]). Baun et al. gehen dabei noch weiter und sehen eine SOA gar als *„fundamentale Voraussetzung für das Cloud Computing“* [BKNT09, S. 19] an. Dabei wird der Begriff SOA sehr unterschiedlich interpretiert, eine klare Begriffsdefinition liegt bislang nicht vor [RHS05]. Allgemein kann festgehalten werden, dass es sich bei einer SOA um einen technologieunabhängigen Architekturstil handelt, der den Entwurf von Systemlandschaften aus einzelnen Diensten beschreibt. Unter einer SOA kann demnach eine *„Methode für den Entwurf von Systemlandschaften“* [Sie07, S. 111] verstanden werden.

Ein Dienst wird in einer SOA als *Service* bezeichnet. Als ein Service wird eine fest definierte Leistung verstanden, welche *„als Element eines oder mehrerer größerer Verarbeitungsabläufe verwendet werden kann“* [RHS05, S. 413]. Nach Richter et al. stellt ein solcher Service eine fachliche Sicht auf eine Funktionalität eines Anwendungsbausteins dar, dessen Implementierung aber verborgen bleibt – dies stellt einen elementaren Grundgedanken einer SOA dar [RHS05]. Services werden dabei häufig als Geschäftsprozesse orchestriert, die wiederum als Service angeboten werden können. Auf diesem Konzept basierend zielt eine SOA demnach darauf ab, *„einen einheitlichen Ansatz für die Beschreibung und Realisierung von Geschäftsprozessen“* [BKNT09, S. 20] zu bieten. Des Weiteren sollen die Abhängigkeitsbeziehungen zwischen den einzelnen Systemen in einer SOA verringert werden. Dies wird dadurch erreicht, dass die Systeme über die Dienste nur noch lose miteinander gekoppelt sind. Dies ist insbesondere im Kontext von Systemlandschaften relevant, da dort häufig eine hohe Kopplung vorherrscht (vergleiche Abschnitt 2.1.3).

Grundsätzlich besteht eine SOA aus den Elementen *Serviceanbieter*, *Servicenutzer* und *Serviceverzeichnis*. Abbildung 2.10 illustriert deren Zusammenspiel.

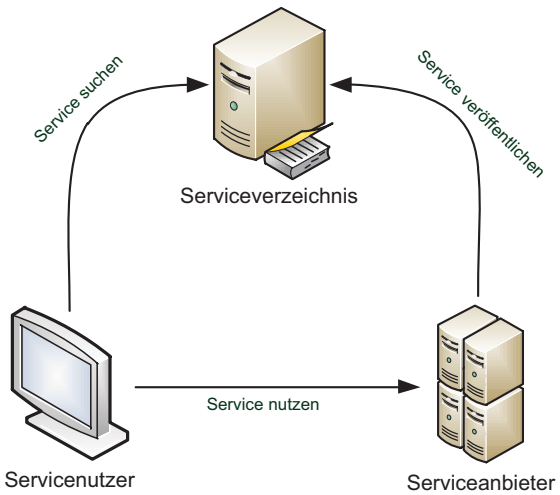

Abbildung 2.10: Elemente einer serviceorientierten Architektur (in Anlehnung an [BKNT09])

Mit Hilfe des Serviceverzeichnisses kann ein Servicenutzer den für seine Zwecke passenden Dienst auswählen. Da die Wiederverwendung von in Software implementierten Funktionen eines der zentralen Ziele einer SOA darstellt, handelt es sich bei dem Serviceverzeichnis um einen elementaren Bestandteil einer SOA. Denn um die Wiederverwendung von Services zu ermöglichen, ist es unabdingbar, überhaupt von deren Existenz zu wissen. In der Praxis wird jedoch häufig darauf verzichtet [BKNT09]. Damit ein Service im Verzeichnis auftaucht, werden alle Services eines Serviceanbieters im Serviceverzeichnis publiziert. Einen veröffentlichten Service kann ein Nutzer dann schließlich verwenden.

Weiterhin verfügt eine SOA über bestimmte Eigenschaften, mit denen zahlreiche Möglichkeiten einhergehen:

- Der Nutzer ist nur die definierte Schnittstelle des Services bekannt, die eigentliche Implementierung bleibt ihm verborgen. Dies führt zu einer Reduktion der Komplexität, was die Systemlandschaft besser beherrschbar macht [RHS05].
- Aus den verteilten Services können schnell neue Prozesse realisiert werden, ohne dafür großen Aufwand betreiben zu müssen [RHS05].
- Die Implementierung der Services kann ausgetauscht werden, ohne dass sich die Schnittstelle ändert [Sie07].
- In Abhängigkeit des tatsächlichen Grades der Wiederverwendung können hinsichtlich Wartung und Entwicklung der Systeme mittelfristig Kosten eingespart werden [RHS05].
- Durch eine SOA erwächst die Möglichkeit, gewachsene Systemlandschaften fachlich zu partitionieren und in erneuerbare Bestandteile zu zerlegen [RHS05].

Durch die Einführung einer SOA ergeben sich folglich zahlreiche Möglichkeiten. Jedoch gehen damit auch nicht zu unterschätzende Herausforderungen einher:

- Derzeit mangelt es an einer etablierten Methodik zur Modellierung, *„um zu sinnvoll definierten Services zu gelangen“* [RHS05, S. 414].
- SOA erfordert eine stark vorausschauende Vorgehensweise sowohl bei der IT-Organisation als auch in den Fachbereichen [RHS05].
- Im Rahmen einer SOA kommt es oft zu Performance-Problemen, die es zu bewältigen gilt [Sie07].

Da im Kontext dieser Arbeit die technischen Methoden zur Umsetzung einer SOA sowie deren organisatorische Aspekte nicht von Bedeutung sind, wird auf deren Erläuterung an dieser Stelle verzichtet.

Im Kontext von SOA beziehungsweise Cloud Computing spielt vor allem das *Service Level Agreement* (**SLA**) eine wichtige Rolle. Ein SLA ist eine vertragliche Vereinbarung zwischen zwei Parteien, welche *“Kosten, Preise, Ziele [...], den Servicegrad und die Qualität des laufenden Systems“* [KD11] dokumentiert und zudem Regelungen enthält, welche Ansprüche bestehen, sollte die erbrachte Leistung nicht den Vereinbarungen entsprechen.

2.4.2 Die Bedeutung von „as a Service“ in kleinen und mittleren Unternehmen

IT ist heute ein unverzichtbarer Faktor, wenn es um die Ausführung und Unterstützung von Geschäftsprozessen geht [Rud09]. Gerade in Branchen, in denen IT nicht zu den Kernkompetenzen zählt, sondern nur eine Unterstützungsfunktion bekleidet, besteht hinsichtlich der eigentlichen und der ihr tatsächlich entgegengebrachten Bedeutung häufig eine große Diskrepanz [Rud09]. Nach Rudolph trifft dies sowohl auf große wie auch auf kleine und mittlere Unternehmen zu. KMU sehen sich dabei lediglich *„mit anderen Rahmenbedingungen konfrontiert“* [Rud09, S. 73].

Die Konzepte des Cloud Computings sind dabei vor allem für kleine und mittlere Unternehmen interessant, da sie viele der Probleme, wie sie in der Praxis auftreten (vergleiche Abschnitt 2.3.3), lösen beziehungsweise abmildern können. So bieten gerade die „as a Service“-Konzepte als Mietmodelle eine hohe Investitionssicherheit für KMU [MTK10]. Daneben wird häufig dem gegenwärtigen Problem der mangelnden Fachkenntnisse entgegengewirkt, da die Verwaltung der bezogenen Lösung vollständig vom Anbieter übernommen wird und sich das Unternehmen nur noch mit der Nutzung auseinandersetzen muss.

Die Relevanz von „as a Service“-Lösungen wird auch dadurch deutlich, dass bereits zahlreiche Angebote verfügbar sind und in der Praxis auch tatsächlich genutzt werden. Im Folgenden wird daher ein exemplarischer Einblick in den „as a Service“-Markt gegeben.

Verbreitung von „as a Service“ in der Wirtschaft

Laut einer aktuellen Studie des Marktforschungsunternehmens Gartner[5] werden immer mehr „as a Service“-Angebote von Unternehmen genutzt. Dieser Trend werde auch in den kommenden Jahren anhalten und sich sogar noch verstärken [MEE+11]. SaaS-Angebote sind dabei für eine Vielzahl von Einsatzgebieten verfügbar:

- CRM-Anwendungen - SaaS-Lösungen für das Customer Relationship Management (**CRM**) gewinnen immer mehr an Bedeutung und sind mit mehr als 20% Anteil an allen verfügbaren SaaS-Lösungen die am häufigsten genutzten Anwendungen. Einer der bekanntesten Anbieter in diesem Bereich ist Salesforce.com[6].

- Kollaborations-Anwendungen - Neben CRM-Anwendungen ist auch Kollaborationssoftware als SaaS-Lösung sehr weit verbreitet. Auch hier zählt Salesforce.com zu den bekanntesten Anbietern.

- ERP-Anwendungen - Bislang kommt den ERP-Systemen, die als SaaS-Lösung zur Verfügung stehen, im Vergleich zu CRM-Lösungen eine eher geringe Bedeutung zu. Aber gerade für KMU, für die die Einführung eines ERP-Systems aufgrund von Budgetrestriktionen in der Regel nicht realisierbar ist, sind SaaS-Lösungen für den Bezug eines ERP-Systems durchaus interessant. Eine der bekanntesten Lösungen in diesem Bereich ist SAP Business ByDesign®[7].

[5] http://www.gartner.com

[6] http://www.salesforce.com/

[7] http://www.sap.com/germany/sme/solutions/businessmanagement/businessbydesign/index.epx

Derartige Lösungen sind auch für KMU von hoher Bedeutung oder gar, wie im Falle von SAP Business ByDesign®, sogar auf diese zugeschnitten.

2.5 Zusammenfassung

In diesem Kapitel wurden die Grundlagen für das Verständnis dieser Arbeit geschaffen. Zunächst erfolgte in Abschnitt 2.1 eine Definition des Terminus „Systemlandschaft“ anhand einer interdisziplinären Betrachtung der Begriffe „System“ und „Landschaft“. Daneben wurden weitere, wesentliche Charakteristika von Systemlandschaften herausgearbeitet und zudem eine Betrachtung von deren Dimensionen („Mensch“, „Aufgabe“ und „Technik“) vorgenommen.

In Abschnitt 2.2 wurde der Modellbegriff definiert. Da der Begriff in der Wissenschaft nicht einheitlich verwendet und auch in der Wirtschaftsinformatik immer wieder kontrovers diskutiert wird, wurde zunächst der allgemeine Modellbegriff vorgestellt. Im Anschluss wurde dann durch die Verknüpfung mit dem in der Wirtschaftsinformatik vorherrschenden Verständnis von Modellen eine Definition für diese Arbeit hergeleitet.

Da sich der im Rahmen dieser Arbeit zu entwickelnde Modellierungsansatz auf kleine und mittlere Unternehmen bezieht, wurde in Abschnitt 2.3 eine Definition von KMU vorgenommen. Dabei wurden zum einen die vorherrschenden quantitativen Definitionen von KMU herangezogen, zum anderen aber auch die qualitativen Merkmale, die KMU insbesondere auszeichnen, vorgestellt. Eine Definition kleiner und mittlerer Unternehmen erfolgte dann anhand quantitativer Merkmale, in Verbindung damit wurde aber zudem auf die wesentlichen qualitativen Merkmale eingegangen.

In Abschnitt 2.4 wurde dann abschließend das Cloud Computing vorgestellt. Eine einheitliche Definition des Begriffs liegt bisher noch nicht vor. Am weitesten verbreitet und akzeptiert ist bislang die Definition des NIST, auf die auch in dieser Arbeit zurückgegriffen wird. Neben dieser Definition wurden die einzelnen Ebenen des Cloud Computings und zudem dessen wesentliche Charakteristika vorgestellt und erläutert. Da im Kontext von Cloud Computing vor allem auch serviceorientierte Architekturen von hoher Bedeutung sind, wurden diese ebenfalls betrachtet. Weiterhin wurde die Bedeutung der Konzepte des Cloud Computings in KMU herausgestellt.

Abschließend kann hinsichtlich des Einsatzes von IT im kleinen und mittleren Unternehmen ein Bedarf an einer Methode zur Verbesserung des IT-Einsatzes festgestellt werden. Dieser ergibt sich jedoch nicht nur aus Sicht der einzelnen Unternehmen: Wie in Abschnitt 2.3.2 dargelegt wurde, ergibt sich ein Bedarf an einer Methodik vor allem auch hinsichtlich der volkswirtschaftlichen Bedeutung von KMU. Eine Methode zur Verbesserung kann dabei, wie in Abschnitt 2.1.2 gezeigt, die Abbildung der (System-)Landschaften in Unternehmen sein, da erst die Abbildung der Landschaft deren nachhaltige Entwicklung ermöglicht – und somit ein höherer Beitrag zur Erreichung der Unternehmensziele geleistet werden kann. Insbesondere diese Abbildung und deren Verknüpfung mit den Konzepten des Cloud Computings schaffen für KMU folglich hohe Potentiale.

Kapitel 3

Ansätze für die Modellierung von Systemlandschaften

Um Software im Allgemeinen beziehungsweise IKS im Speziellen in Unternehmen nutzbar zu machen, bedarf es einer genauen Planung. Dabei ist es nicht nur notwendig, die einzelnen Systeme zu planen, sondern vielmehr die gesamte Systemlandschaft planerisch zu entwickeln (vergleiche Abschnitt 2.1.2). Im Rahmen dieses Kapitels werden daher verschiedene Ansätze untersucht, die grundsätzlich für die Modellierung von Systemlandschaften geeignet sein sollten.

Um eine Untersuchung dieser Ansätze durchführen zu können und die Vergleichbarkeit der Untersuchungen zu gewährleisten, werden in Abschnitt 3.1 zunächst Untersuchungskriterien aufgestellt. Die Untersuchung wird dabei durch die Modellierung eines Beispielszenarios unterstützt, welches in Abschnitt 3.2 vorgestellt wird. Im Anschluss daran werden in Abschnitt 3.3 vier Ansätze für die Modellierung von Systemlandschaften untersucht. Diese wurden so ausgewählt, dass sie hinsichtlich ihrer Komplexität sowie der Investitionskosten auch für KMU interessant sein können. Neben einer allgemeinen Beschreibung der Ansätze wird dabei auch auf deren Elemente und Struktur eingegangen. Nach der erfolgreichen Durchführung der Untersuchungen wird ein abschließender Vergleich durchgeführt. Eine Zusammenfassung wird das Kapitel schließen.

3.1 Untersuchungskriterien

Die Kriterien für die Untersuchung lassen sich in zwei verschiedenen Kategorien identifizieren: Zum einen betrifft dies die Abbildung der *Komponenten* einer Systemlandschaft, zum anderen können Eigenschaften der *Sprache beziehungsweise des Werkzeugs* selber als Kategorie herangezogen werden. Zu diesen Eigenschaften zählen beispielsweise die Anzahl der Elemente der Sprache oder die Möglichkeit, statische oder dynamische Charakteristika einer Systemlandschaft abzubilden. Eine vollständige Liste der Kriterien kann Tabelle 3.1 entnommen werden (in Anlehnung an [Zwa09]).

Kategorie	Kriterium	Erläuterung
Komponenten		
	Anwendungssoftware	Software für eine (betriebliche) Aufgabe, zum Beispiel IKS.
	Hardware	Technische Systeme, beispielsweise Server.
	Systemsoftware	Grundlage für Anwendungssoftware, beispielsweise Betriebssystem.
	Geräte für Betrieb	Technische Basisausstattung, zum Beispiel Router.
	Cloud Computing	Konzepte des Cloud Computings: SaaS, IaaS, PaaS.
	Services	Fest definierte Leistung, die als Element eines oder mehrerer größerer Verarbeitungsabläufe verwendet werden kann.
	SLAs	Vertragliche Regelungen zwischen Anbieter und Nutzer von Services.
Sprache / Werkzeug		
	Anzahl Elemente	Anzahl aller Elemente der Sprache (ohne Relationen).
	Eindeutigkeit	Alle Elemente beschreiben eindeutig ihr Äquivalent im Diskurssystem.
	Konfiguration	Es besteht die Möglichkeit, Konfigurationen für Hardware und Software anzugeben.
	Statisch/Dynamisch	Statische beziehungsweise dynamische Aspekte einer Systemlandschaft können modelliert werden.

Tabelle 3.1: Kriterien für die Untersuchung

Im Rahmen der Analyse der einzelnen Ansätze wird dabei die nachfolgende Notation verwendet:

- ✓ Kriterium wird unterstützt.
- (✓) Kriterium abbildbar, aber nicht direkt unterstützt.
- – Kriterium nicht unterstützt.

3.2 Beispielszenario

Um eine umfassende Analyse der einzelnen Modellierungsansätze zu gewährleisten, ist es sinnvoll, diese hinsichtlich ihrer Eignung zur Abbildung von Systemlandschaften zu erproben. Um dabei die Vergleichbarkeit der Untersuchungen sicherzustellen, ist es weiterhin sinnvoll, die Modellierung anhand eines einheitlichen Beispielszenarios vorzunehmen. Dieses wird im Folgenden definiert.

Das Ziel des Beispielszenarios ist die Nutzung der webbasierten Projektmanagement-Software *Redmine*[1] in Verbindung mit der Entwicklungsumgebung *Eclipse*[2] und der SaaS-basierten CRM-Lösung *Service Cloud*[3] von Salesforce.com. Eine gemeinsame Nutzung dieser Anwendungen ist durchaus sinnvoll, da Eclipse mit Redmine voll integrierbar

[1] `http://www.redmine.org`
[2] `http://www.eclipse.org`
[3] `http://www.salesforce.com/de/crm/customer-service-support/`

ist und beispielsweise für das Erstellen von Tickets[4] aus der Anwendung heraus genutzt werden kann. Über Service Cloud können zudem Meldungen von Kunden direkt in Redmine berücksichtigt werden. Auf die dort vermerkten Fehler hat dann das gesamte Entwicklungsteam zugriff und kann diese beheben. Abbildung 3.1 illustriert dieses Szenario.

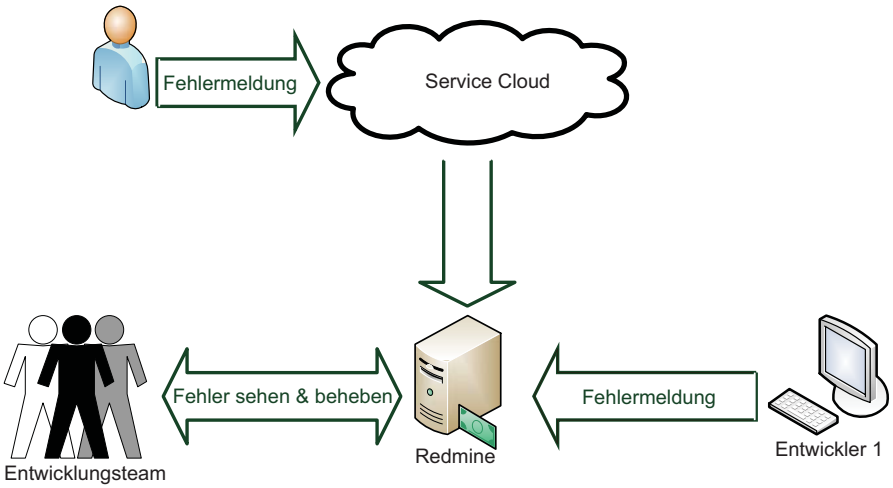

Abbildung 3.1: Konstellationen im Beispielszenario

Redmine und Eclipse können jedoch nicht komplett eigenständig betrieben werden, sondern sind von anderen Anwendungen abhängig. Daher werden nachfolgend deren Abhängigkeiten aufgezeigt, um abschließend ein konkretes Beispielszenario zu konstruieren. Auf Einbettung der Anwendungen in einen Geschäftsprozess wird dabei bewusst verzichtet, um zum einen die Komplexität des Beispielszenarios gering zu halten und zum anderen, da der Fokus dieser Arbeit auf der technischen Dimension von Systemlandschaften liegt.

Redmine

Redmine kann nicht komplett eigenständig betrieben werden, sondern weist bestimmte Abhängigkeiten auf. Zunächst benötigt Redmine das Web Application Framework *Ruby on Rails*, da es mit diesem implementiert wurde und es somit eine Grundvoraussetzung für den Betrieb darstellt. Daneben sind einige weitere, jedoch nicht feste, Abhängigkeiten vorhanden. Für den Betrieb von Redmine sind neben Ruby on Rails weiterhin ein *Rails Application Server*, ein *Datenbankmanagementsystem* (**DBMS**) und (optional) ein *Mail-Server* notwendig. Da diese schwachen Abhängigkeiten konkret umgesetzt werden müssen, ist es notwendig, konkrete Implementierungen auszuwählen. Tabelle 3.2 zeigt die zuvor geschilderten Abhängigkeiten und mögliche Implementierungen auf. Neben der notwendigen Software sind in Tabelle 3.2 auch die benötigte Hardware, Systemsoftware sowie notwendige Geräte für den Betrieb aufgeführt. Zudem sind an die Hardware beziehungsweise an die Geräte für den Betrieb bestimmte Anforderungen gebunden, die für das Beispielszenario definiert wurden.

Als Rails Application Server kommen *WEBrick*, *Mongrel* und *Phusion Passenger* in Frage. Während WEBrick und Mongrel eigenständig betrieben werden können, benötigt

[4]Elektronische Form einer Meldung, zum Beispiel Störungs- oder Fehlermeldung

Komponente	Produkt	Anforderungen	Implementierungen
Anwendungs-system	Redmine	Ruby on Rails	-
		Rails Application Server	WEBrick, Mongrel, Phusion Passenger
		DBMS	SQLite, MySQL, PostgreSQL
		Mail-Server	Postfix, Remote SMTP
Systemsoftware	Betriebssystem		Windows 2008 Server, Debian 5, Gentoo
	Ruby		Ruby 1.8, Ruby 1.9, jRuby
	Webserver		Apache, nginx
	Mail-Server		Postfix, Remote SMTP
Hardware	Anwendungsserver	1000MBit LAN	-
	Datenbankserver	10TB Speicherplatz, 12GB RAM	-
Geräte für Betrieb	Firewall	Internetverbindung	-
	Unterbrechungsfreie Stromversorgung	-	-

Tabelle 3.2: Konfiguration des Beispielszenarios

Phusion Passenger zusätzlich einen Webserver, um ausgeführt werden zu können. Hierfür kommen der *Apache Webserver* oder *nginx* in Betracht. Als Systemsoftware ist in jedem Fall ein Betriebssystem benötigt. Hierbei können alle gängigen Windows- und Linux-Varianten genutzt werden. Weiterhin wird für Ruby on Rails eine Ruby-Implementierung benötigt. Kandidaten sind hier das Standard-*Ruby* in Version 1.8 oder 1.9. Es kommen aber auch alternative Implementierungen wie *jRuby* in Frage. Wird jRuby eingesetzt, wird zusätzlich noch das *Java Development Kit* (**JDK**) benötigt. Redmine unterstützt eine Vielzahl von Datenbanksystemen, sodass eine ganze Reihe von DBMS einsetzbar ist. Geeignete Lösungen stellen *SQLite*, *MySQL* oder *PostgreSQL* dar. Ein Mail-Server wird zwar nicht zwingend benötigt, ist aber wegen des in Redmine integrierten Benachrichtigungssystems durchaus sinnvoll. Dafür kann entweder ein eigener Mail-Server betrieben (zum Beispiel mit *Postfix*) oder alternativ ein externer Server (*Remote SMTP*) genutzt werden.

In diesem Szenario wird davon ausgegangen, dass für den Betrieb von Redmine zwei Server zur Verfügung stehen. Auf dem einen Server wird der Rails Application Server mit Redmine betrieben, auf dem anderen Server steht ein DBMS für die Datenhaltung zur Verfügung. Um die ständige Verfügbarkeit beider Server zu gewährleisten, ist daneben eine unterbrechungsfreie Stromversorgung (**USV**) für jeden Server notwendig. Zur Absicherung der Systeme ist zudem eine Firewall Bestandteil des Szenarios.

Eclipse

Eclipse ist eine plattformunabhängige, integrierte Entwicklungsumgebung [Fou11]. Eclipse basiert dabei auf dem Framework Equinox[5], welches in Java implementiert ist. Folglich wird ein *Java Runtime Environment* (**JRE**) für die Nutzung von Eclipse benötigt. Des Weiteren wird mindestens ein Computer benötigt, auf dem Eclipse installiert und ausgeführt werden kann. Aufgrund der Plattformunabhängigkeit von Eclipse ist es dabei unerheblich, welches Betriebssystem dabei zum Einsatz kommt.

[5] `http://www.eclipse.org/equinox`

Service Cloud

Für den Betrieb von Service Cloud wird, abgesehen von einem PC mit Browser und Internetzugang, keine weitere Ausstattung benötigt. Jedoch ist mit der Nutzung von Service Cloud ein Service Level Agreement verbunden, welches im Beispielszenario zu berücksichtigen ist.

Konkretes Beispielszenario

Da es sich bei diesem Beispiel um ein fiktives Szenario handelt, kann eine beliebige (lauffähige) Konstellation der zuvor geschilderten Komponenten gewählt werden. Die für das Beispielszenario gewählte Software-Konstellation ist in Abbildung 3.2 zu sehen. Aus Gründen der Übersichtlichkeit wurde dabei auf die Darstellung von Hardware-Komponenten verzichtet.

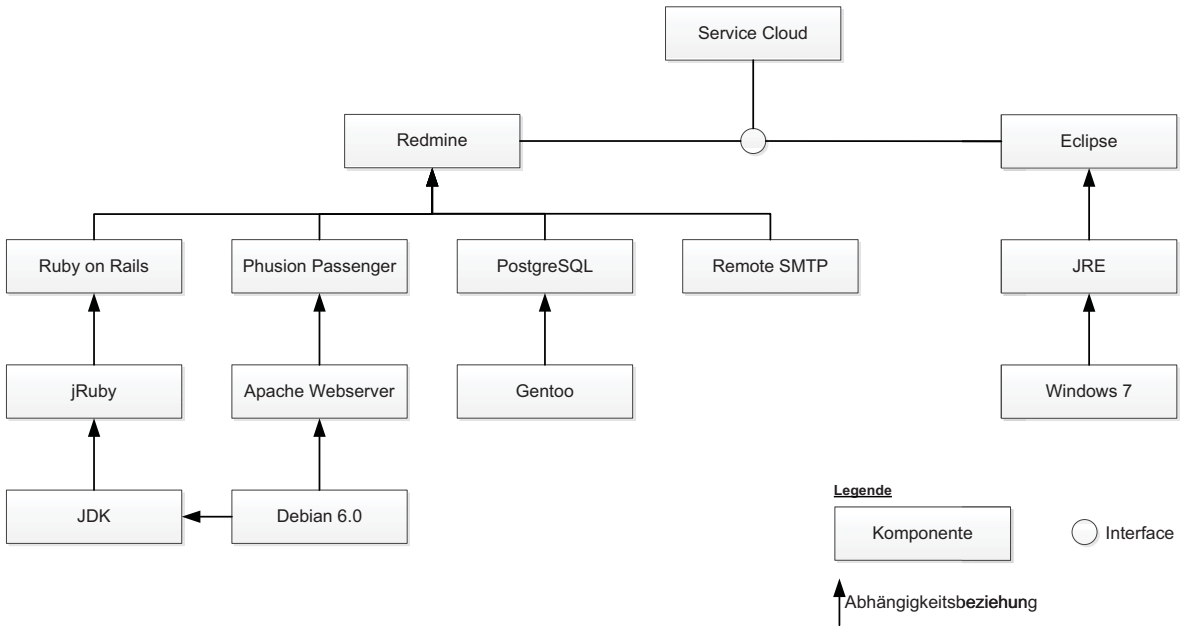

Abbildung 3.2: Beispielszenario

In diesem Szenario wird davon ausgegangen, dass Redmine zwei Web-Services zur Verfügung stellt, um erstens Fehler melden zu können und zweitens die erfolgreiche Behebung dieser Fehler anzugeben. Diese Services können von anderen Anwendungen verwendet werden. Daneben ist ein Computer („Workstation“) Teil des Beispielszenarios, auf dem Eclipse installiert und dieses mit Redmine integriert ist.

3.3 Vergleich von Modellierungsmöglichkeiten

Nachfolgend werden vier ausgewählte Ansätze für die Modellierung von Systemlandschaften vorgestellt. Jeder wird zunächst allgemein beschrieben. Im Anschluss werden dann die einzelnen Elemente des Ansatzes beziehungsweise des Werkzeugs vorgestellt. Anschließend wird das im vorherigen Abschnitt definierte Beispielszenario umgesetzt.

Die Analyse anhand der vorgestellten Untersuchungskriterien wird die Betrachtung jedes Ansatzes schließen. Abschließend erfolgt eine Gegenüberstellung .

3.3.1 Unified Modeling Language: Deployment Diagram

Die *Unified Modeling Language* (**UML**) ist eine Sprache für die Modellierung, Dokumentation, Spezifizierung und Visualisierung von komplexen Softwaresystemen [RHQ+07]. Die UML umfasst 14 Diagrammarten, von denen sieben der Modellierung der Statik des Systems dienen (*Strukturdiagramme*). Die sieben übrigen Diagramme dienen der Beschreibung der Dynamik von Systemen (*Verhaltensdiagramme*).

Zu der Klasse der Strukturdiagramme zählt das *Deployment Diagram* (im deutschsprachigen Raum auch als *Verteilungsdiagramm* bezeichnet). Das Deployment Diagram unterscheidet dabei im Wesentlichen zwischen den Diagramm-Komponenten *Artefakt* und *Knoten*. Unter einem Artefakt ist dabei die Spezifikation einer physischen Informationseinheit zu verstehen, die im Rahmen von Softwareentwicklung beziehungsweise durch Installation oder Betrieb von Software genutzt oder generiert wird [Obj10]. Ein Knoten repräsentiert in der Regel eine Hardware-Einheit [RHQ+07], kann jedoch auch Software darstellen. Das Deployment Diagramm beschreibt nun die Zuordnung von Artefakten zu Knoten [Obj10]. Dafür stellt es verschiedene Notationselemente zur Verfügung, die nachfolgend vorgestellt werden.

Notationselemente

Eine Übersicht der vom Deployment Diagram bereitgestellten Notationselemente ist Abbildung 3.3 zu entnehmen.

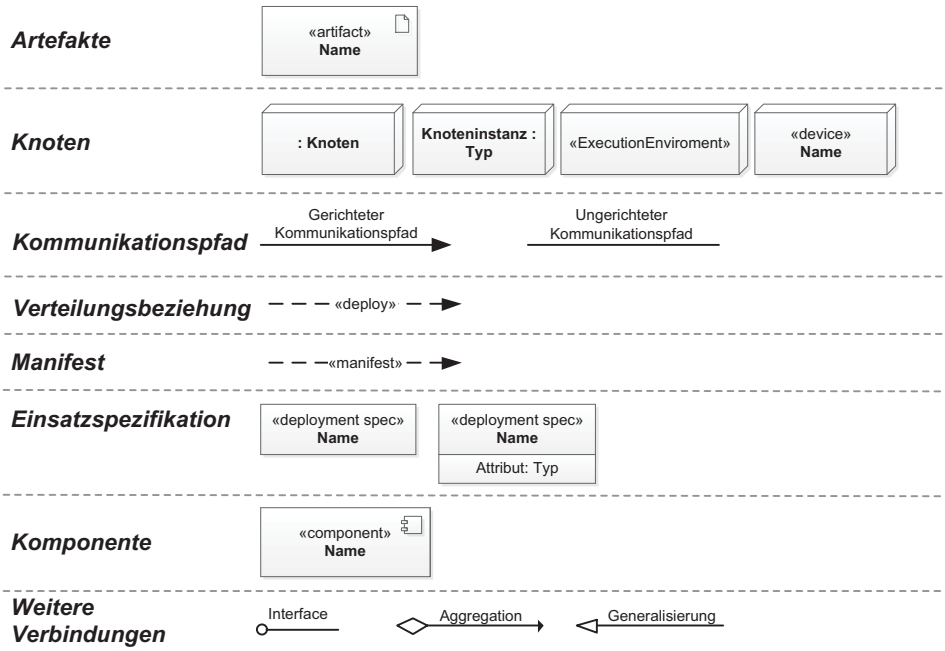

Abbildung 3.3: Notationselemente: Deployment Diagram

Das `Artefakt` wurde bereits einleitend als Spezifikation einer physischen Informationseinheit beschrieben. Ein Artefakt kann durch eine Einsatzspezifikation, ein `deployment spec`, konfiguriert werden. Dabei werden in einem `deployment spec` bestimmte Attribute angegeben, welche die Ausführungsparameter des Artefakts definieren. `Knoten` stellen eine IT-Ressource dar. Bei `Knoteninstanzen` handelt es sich um die konkrete Realisierung einer solchen Ressource. Zusätzlich zu diesen Knoten stellt das Deployment Diagram zwei Spezialknoten bereit:

- Der Knoten `ExecutionEnviroment` definiert eine Umgebung, in der bestimmte Software ausgeführt werden kann.
- Der Knoten `device` beschreibt eine Ressource, die über eine eigene Rechenkapazität verfügt.

Für die Darstellung von Kommunikationspfaden stehen zwei Möglichkeiten zur Verfügung: Der `gerichtete Kommunikationspfad` und der `ungerichtete Kommunikationspfad`. Beide dienen der Modellierung von Kommunikation zwischen Knoten beziehungsweise Knoteninstanzen und werden für die Modellierung komplexer Netzwerke verwendet.

Die Zuordnung von Artefakten zu Knoten wird durch die Beziehung `deploy` beschrieben. Die Relation `manifest` dient der Kennzeichnung, welches Artefakt durch welche Komponente (`component`) realisiert wird. Eine Komponente stellt ein Modul eines Systems dar, welches bestimmte Schnittstellen realisiert.

Neben den bereits vorgestellten Elementen existieren noch drei weitere Beziehungen, die im Deployment Diagram verwendet werden können:

- Das `Interface` definiert eine Schnittstelle für die Kommunikation.
- Die `Aggregation` drückt eine Ganzes/Teil-Beziehung aus.
- Die `Generalisierung` stellt eine Beziehung zwischen einem allgemeinen und einem speziellen Klassifizierer dar.

Beispielszenario

Das Ergebnis der Modellierung des Beispielszenarios mit Hilfe des Deployment Diagrams ist in Abbildung 3.4 zu sehen.

Sowohl die Hardware als auch die Software werden im modellierten Szenario durch Knoten repräsentiert. Hardware wird dabei durch einen Knoten vom Stereotyp `device` dargestellt. Die Betriebssysteme und zudem jRuby sind im Modell durch den speziellen Knoten `ExecutionEnviroment` eingebunden und bilden die Grundlage für die Software, die in ihnen platziert ist. Einzelne Bestandteile von Software sind als Komponenten innerhalb von Software modelliert: Der PhusionPassenger, ein Modul des Apache-Webservers, sowie die beiden Funktionalitäten `reportBug` und `fixBug` von Redmine. Da diese als Service genutzt werden sollen, ist Redmine zudem mit einem Interface verbunden, welches den Aufruf dieser Funktionen durch externe Anwendungen ermöglichen soll. Die SaaS-Lösung Service Cloud und auch der externe SMTP sind ebenfalls als Knoten modelliert, da das Deployment Diagram keine spezielle Repräsentation externer Software oder Hardware ermöglicht.

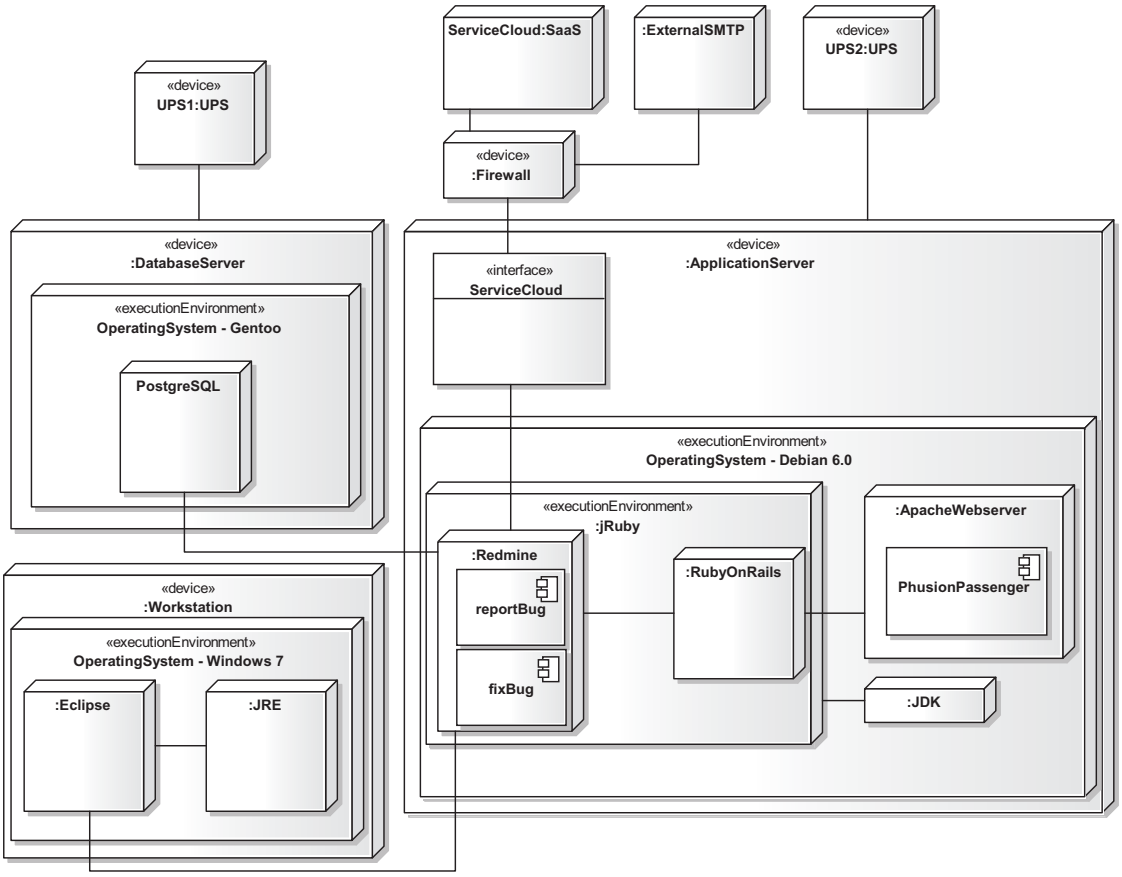

Abbildung 3.4: Beispielszenario: Deployment Diagram

Analyse

Nachdem nun die Elemente des Deployment Diagramms bekannt sind und das Beispielszenario umgesetzt wurde, kann auf dieser Basis eine Analyse anhand der in Abschnitt 3.1 vorgestellten Kriterien vorgenommen werden.

Wie bereits zuvor erläutert, werden im Beispielszenario sowohl Hardware als auch Software durch Knoten repräsentiert. Hardware kann dabei durch den Knoten-Stereotyp `device` dargestellt werden. Allerdings ist eine weitere Unterscheidung von Hardware nicht möglich. Folglich kann beispielsweise nicht zwischen einem Server und einer unterbrechungsfreien Stromversorgung unterschieden werden. Somit ist eine Abbildung jeder Art von Hardware generell denkbar, eine tatsächliche Unterscheidung verschiedener Arten ist jedoch nicht möglich. Konfigurationen, zum Beispiel für Software, können lediglich für Artefakte angegeben werden. Eine Differenzierung von Software kann durch das Nutzen von `ExecutionEnviroments` realisiert werden. So wurde beispielsweise im Beispielszenario das Betriebssystem der Server als `ExecutionEnviroment` modelliert. Prinzipiell ließe sich auf diese Weise jede Art von Systemsoftware modellieren. Jedoch würde so schnell eine hohe Verschachtelung von Elementen erreicht, was das Modell schnell sehr komplex werden ließe. Eine andere Alternative besteht jedoch nicht, da keine explizite Definition einer Ausführungsumgebung in Verbindung mit den ihr zugehörigen Komponenten möglich ist. Einzelne Module von Anwendungssoftware können, wie bereits im Beispielszenario gezeigt, durch Komponenten dargestellt werden. Sollen einzelne Kom-

ponenten über eine Schnittstelle von anderen Anwendungen genutzt werden, kann dies nicht explizit dargestellt werden, da ein `interface` nur mit einem Knoten, nicht aber mit einer Komponente verbunden werden kann. Dieses Problem könnte aber umgangen werden, wenn im `interface` die Methoden definiert werden, die es bereitstellt. Nutzt eine Anwendung jedoch nicht alle Funktionen der Schnittstelle, ist anhand der Modellierung nicht nachvollziehbar, welche Funktionen tatsächlich verwendet werden. Ebenso ist keine Abbildung von Service Level Agreements möglich.

Für die Modellierung von Konzepten des Cloud Computings stellt das Deployment Diagram keine Notationselemente bereit. Für deren Modellierung muss somit auf andere Elemente zurückgegriffen werden, wie es beispielsweise im Rahmen des Beispielszenarios erfolgt ist.

Tabelle 3.3 fasst die Ergebnisse der Analyse zusammen.

Komponenten	**Kriterium**	**Analyse**
	Anwendungssoftware	✓
	Hardware	✓
	Systemsoftware	(✓)
	Geräte für Betrieb	(✓)
	Cloud Computing	(✓)
	SaaS	–
	IaaS	–
	PaaS	–
	Services	(✓)
	SLAs	–
Sprache / Werkzeug	**Kriterium**	**Analyse**
	Anzahl Elemente	8
	Eindeutigkeit	–
	Konfiguration	(✓)
	Statisch/Dynamisch	✓ / –

Tabelle 3.3: Ergebnis der Analyse des Deployment Diagramms

3.3.2 IT Modeling Language

Die von Frank et al. entwickelte *IT Modeling Language* (**ITML**) [FHK+09] ist eine domänenspezifische Modellierungssprache für den Entwurf von IT-Infrastrukturen. Die Sprache ist ein Bestandteil von „Multi-Perspective Enterprise Modelling“ (**MEMO**) [Fra08], einer Gruppe von Modellierungssprachen für die multiperspektivische Unternehmensmodellierung. MEMO stellt eine Meta-Metasprache dar und bildet die Grundlage für alle Sprachen der Sprachfamilie [Fra08]. Unter multiperspektivische Unternehmensmodellierung ist dabei die Erstellung von Unternehmensmodellen zu verstehen. Bei einem Unternehmensmodell handelt es sich in diesem Kontext um einen konzeptionellen Entwurf des gesamten Unternehmens. Auf ein solches Unternehmensmodell gibt es verschiedene Sichten, die durch MEMO abgedeckt werden. Diese Sichten sind [Fra93]:

- Die *strategische Perspektive*, in der die Unternehmensziele festgelegt und langfristige Strategien geplant und bewertet werden,

- die *organisatorische Perspektive*, die auf die *„Gestaltung und Durchführung der kooperativen Handlungen im Unternehmen“* [Fra93, S. 7] abzielt sowie
- die *Informationssystem Perspektive*, die den Entwurf, die Umsetzung und den Betrieb von IKS beschreibt.

Die ITML ist in diesen Rahmen eingebettet. Dabei zielt sie nicht nur auf die Modellierung von Infrastrukturen ab, sondern bindet auch geschäftliche Konzepte in die Modellierung ein. Diese Konzepte sind jedoch nicht explizit Bestandteil der ITML, sondern sind in anderen Bestandteilen von MEMO ausmodelliert. Da der Fokus dieser Arbeit jedoch auf den technischen Gegebenheiten von Systemlandschaften liegt, wird lediglich die ITML und nicht die gesamte MEMO-Familie untersucht.

Notationselemente

Eine Implementierung für die Nutzung der ITML liegt zwar in Form des sich noch in der Entwicklung befindlichen Tools *MEMOCenterNG*[6] vor, war jedoch zum Zeitpunkt der Entstehung dieser Arbeit nicht verfügbar. Daher wird für die Analyse der Sprache das von Frank et al. in [FHK+09] vorgestellte Metamodell verwendet, was für die Untersuchung ausreichend ist. Das Metamodell der ITML ist in Abbildung 3.5 zu sehen.

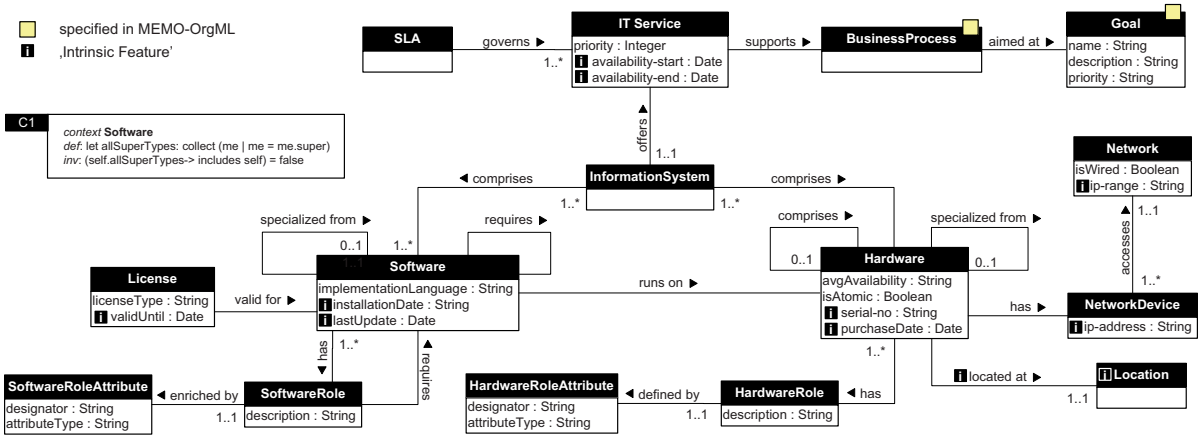

Abbildung 3.5: IT Modeling Language: Metamodell (Quelle: [FHK+09])

Als Kern des Modells kann das `InformationSystem` betrachtet werden. Dieses stellt Services (vergleiche Abschnitt 2.4.1) bereit, die durch die Entität `Service` beschrieben werden. Ein `Service` unterstützt wiederum einen `BusinessProcess`. Dieser steht zudem in Bezug zu einem Unternehmensziel (`Goal`). Ein Service Level Agreement (`SLA`) reguliert einen solchen Service. In Abbildung 3.5 ist zu sehen, dass ein `InformationSystem` aus `Hardware` und `Software` besteht. Hardware und Software können dabei eine bestimmte Funktion erfüllen, was durch `HardwareRole` beziehungsweise `SoftwareRole`

[6] `http://www.wi-inf.uni-duisburg-essen.de/FGFrank/index.php?lang=de&&groupId=1&&contentType=Project&&projId=19`

realisiert wird. Sowohl `HardwareRole` als auch `SoftwareRole` werden dabei durch Attribute, die die jeweilige Rolle beschreiben (`HardwareRoleAttribute` beziehungsweise `SoftwareRoleAttribute`), angereichert. Einer Software wird weiterhin eine Lizenz (`Licence`) zugeordnet.

Zur Hardware gehört ein sogenanntes `NetworkDevice`. Dabei handelt es sich um ein Element, welches verschiedene Hardwarekomponenten miteinander verbindet. Ein solches `NetworkDevice` greift dann auf das eigentliche Netzwerk (`Network`) zu. Wie Abbildung 3.5 weiterhin zu entnehmen ist, können sowohl Hardware als auch Software Abhängigkeiten aufweisen (ausgedrückt durch die *requires*-Beziehung). Ein Beispiel dafür ist, dass die Komponente Redmine aus dem Beispielszenario in Abschnitt 3.2 unter anderem Ruby on Rails benötigt, um lauffähig zu sein.

Beispielszenario

Abbildung 3.6 zeigt das auf Basis des ITML-Metamodells modellierte Beispielszenario. Da zum Zeitpunkt der Anfertigung dieser Arbeit kein Werkzeug zur Verfügung stand, wurde auf eine eigene Repräsentation des ITML-Metamodells zurückgegriffen. Diese nutzt verschiedene Notationselemente der UML.

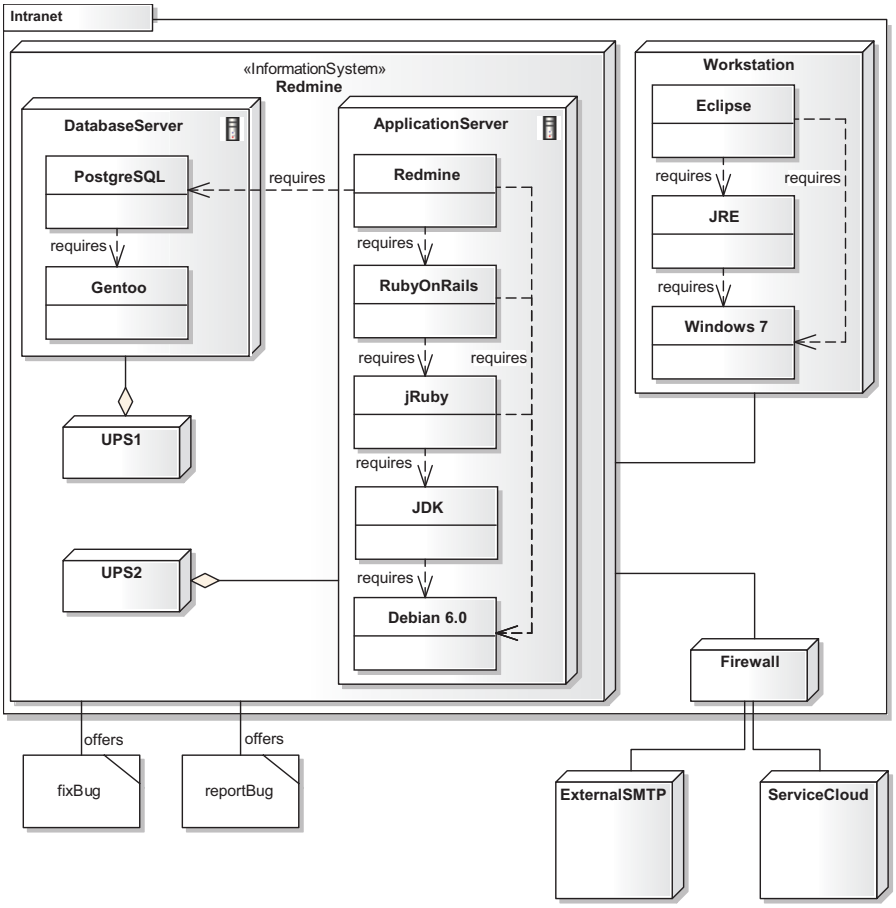

Abbildung 3.6: Beispielszenario: IT Modeling Language

Da `Hardware` (dargestellt als `Node` des Deployment Diagrams) Bestandteil eines Netzes sein kann, wurde diese in ein Netzwerk (`Intranet`) eingebettet. Die Beziehung `runs on`

aus dem Metamodell wird dadurch abgebildet, dass `Software` (Dargestellt als UML-Klasse) innerhalb eines `Node` liegt. Über eine Aggregation wurde die unterbrechungsfreie Stromversorgung als Bestandteil je eines der beiden Server modelliert. Pfeile repräsentieren die `requires`-Beziehung, welche die Abhängigkeiten von Software ausdrücken. Nach dem Metamodell der ITML bestehen Informationssysteme aus Hardware und Software und können Services bereitstellen. Daher wurden die im Beispielszenario beschriebenen Server mit der auf ihnen laufenden Software als Informationssystem modelliert, um die Bereitstellung der Services `reportBug` und `fixBug` zu ermöglichen.

Der externe SMTP wird im modellierten Szenario durch einen `Node` repräsentiert. Ebenso wird die SaaS-Lösung Service-Cloud durch einen `Node` und auf ihm installierte Software repräsentiert. Sie ist, wie auch der SMTP, über eine Firewall mit dem Netzwerk verbunden.

Die Workstation mit Eclipse wird ebenfalls durch einen Knoten mit der entsprechenden Software repräsentiert.

Analyse

Auf Basis der Vorstellung der Sprache und anhand des modellierten Beispielszenarios kann nun eine Analyse der ITML durchgeführt werden. Wie bereits zuvor erwähnt, wird Hardware im Beispielszenario als Knoten und Software als Klasse dargestellt.

Eine Differenzierung verschiedener Arten von Hardware ist im Modell der ITML nicht möglich, weshalb alle Hardwareelemente des Beispielszenarios als Knoten dargestellt sind. Eine Möglichkeit der Differenzierung besteht in der Nutzung der `HardwareRoles`. Auf diese wurde jedoch aufgrund der Übersichtlichkeit des Modells verzichtet. Ein gewisses Maß an Konfiguration von Hardware kann durch die Nutzung von `HardwareRoleAttributes` abgebildet werden.

Ebenso wie bei der Hardware ist auch eine Differenzierung von Software in dem Metamodell der ITML nicht direkt möglich, weshalb Software im Beispielszenario durch die gleichen Notationselemente repräsentiert wird. Auch hier bestünde jedoch die Möglichkeit, für eine genauere Differenzierung auf die `SoftwareRoles` zurückzugreifen. Aus Gründen der Übersichtlichkeit wurde jedoch auch hier auf diese Maßnahme verzichtet. Die Abhängigkeiten von Software lassen sich aus Basis des Metamodells nur sehr umständlich abbilden, da alle Abhängigkeiten einzeln aufgezeigt werden müssen. Dies führt dazu, dass das Modell aufgrund der vielen Relationen schnell unübersichtlich wird. Zudem kann die Kommunikation von Software untereinander nicht direkt abgebildet werden, da die ITML lediglich die Kommunikation zwischen Hardware-Systemen zulässt (vergleiche Abbildung 3.5). Wie auch schon bei der Hardware ist Software nicht direkt konfigurierbar; denkbar wäre dies jedoch über die Nutzung von `SoftwareRoleAttributes`.

Die Bereitstellung von Services wird im Rahmen der ITML lediglich durch Informationssysteme, nicht aber durch eine beliebige Art von Software, ermöglicht. Daher wurden Hardware und Software, wie in Abbildung 3.6 zu sehen, zu einem Informationssystem zusammengeführt. Das Beispielszenario macht jedoch auch deutlich, dass die ITML hinsichtlich der Nutzung von Services einige Schwachstellen aufweist. So ist es zwar möglich, dass ein Informationssystem einen Service bereitstellt und dieser auch einen Geschäftsprozess unterstützt. Jedoch kann nicht abgebildet werden, dass andere

Anwendungen diesen Service auch nutzen können. Daher greifen im modellierten Beispielszenario auch keine Anwendungen auf die Services zu.

Die verschiedenen Konzepte des Cloud Computings können durch die ITML nicht explizit abgebildet werden. Hingegen ist die Modellierung von SLAs möglich. Diese können jedoch nur mit Services assoziiert werden, nicht hingegen mit den Knoten, die im Beispielszenario die SaaS-Lösungen repräsentieren.

Tabelle 3.4 zeigt eine Zusammenfassung der Ergebnisse der vorausgegangenen Analyse.

Komponenten	**Kriterium**	**Analyse**
	Anwendungssoftware	✓
	Hardware	✓
	Systemsoftware	(✓)
	Geräte für Betrieb	(✓)
	Cloud Computing	(✓)
	SaaS	–
	IaaS	–
	PaaS	–
	Services	✓
	SLAs	(✓)
Sprache / Werkzeug	**Kriterium**	**Analyse**
	Anzahl Elemente	15
	Eindeutigkeit	–
	Konfiguration	(✓)
	Statisch/Dynamisch	✓ / –

Tabelle 3.4: Ergebnis der Analyse der IT Modeling Language

3.3.3 Modellierung von IT-Landschaften nach Kirchner

Kirchner spricht sich, wie auch andere Autoren, gegen den Einsatz einer generellen Modellierungssprache wie UML im Kontext der Modellierung von Systemlandschaften aus und stellt daher in [Kir03] einen dedizierten Ansatz vor. Dieser gliedert sich in drei semantische Sichten auf eine Systemlandschaft: *Business Process Layer*, *Software Layer* und *Hardware Layer*.

Kirchner hält das Modell dabei einfach und zielt lediglich auf *„eine grundlegende Darstellung [der] Domäne“* [Kir03, S. 77] ab. Daraus resultieren einige Schwächen im Modell, wie Kirchner selber ausführt. Im Folgenden wird auf den Aufbau des Modells eingegangen und im Anschluss daran das Beispielszenario modelliert.

Notationselemente

Wie bereits einleitend erwähnt, beschränkt sich Kirchners Modell auf die nach seiner Ansicht grundlegenden Elemente der Domäne und verzichtet daher auch auf eine Darlegung von Attributen. Das von ihm entwickelte Metamodell ist in Abbildung 3.7 zu sehen.

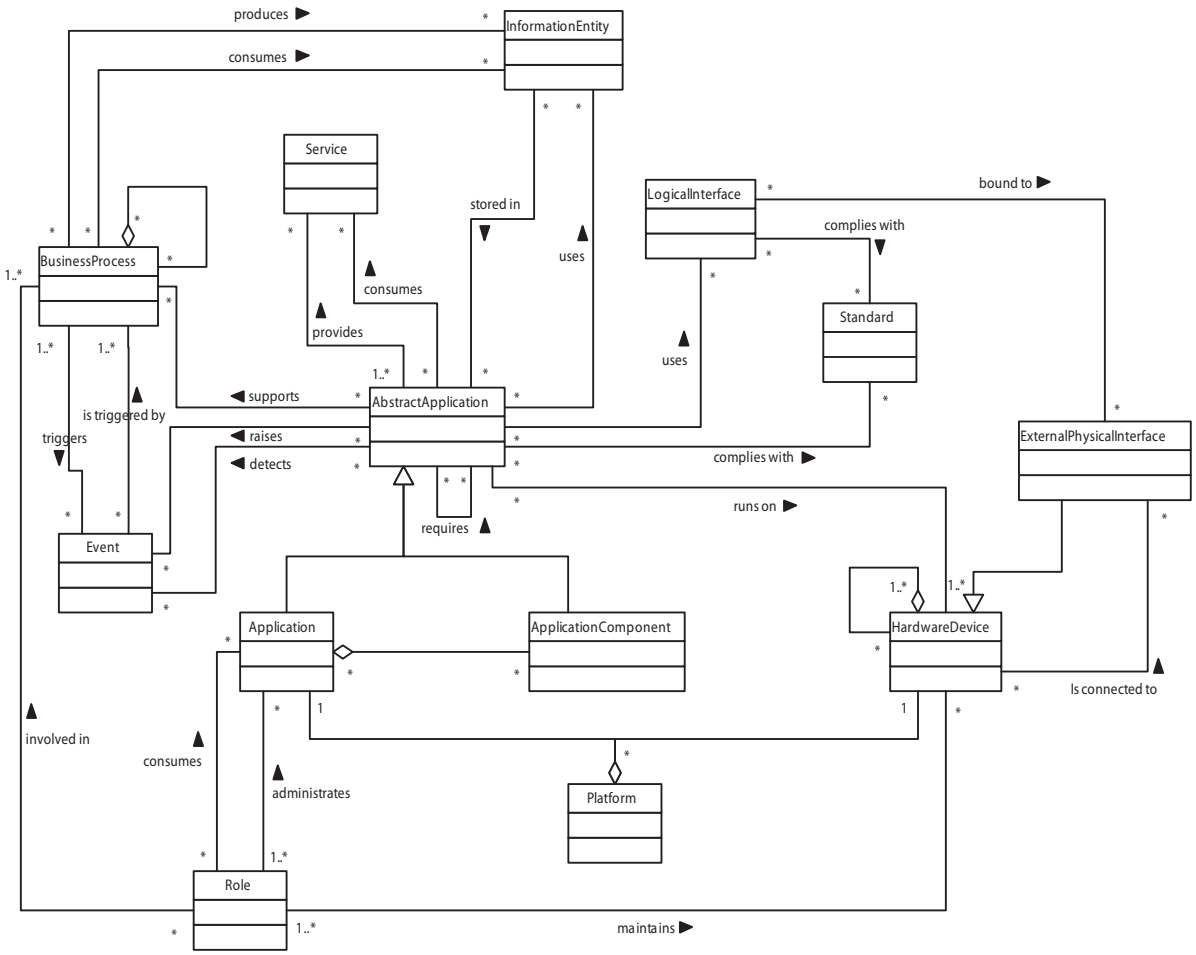

Abbildung 3.7: Metamodell: Modellierung von IT-Landschaften gemäß Kirchner (Quelle: [Kir03])

Den Kern des Modells bilden `BusinessProcess`, `AbstractApplication` und `HardwareDevice`. Bei einem `BusinessProcess` handelt es sich um einen Geschäftsprozess, der eine *„Schnittstelle zu dem Metamodell der [...] MEMO-OrgML“* [Kir03, S. 78] darstellt, weshalb diese Entität auch nicht weiter spezifiziert ist. Ein solcher Geschäftsprozess wird durch `Event` ausgelöst und kann wiederum ein anderes `Event` auslösen. Wesentlich für den Ablauf eines `BusinessProcess` sind die `InformationEntities`, welche von ihnen einerseits genutzt und andererseits auch produziert werden. Bei einer `InformationEntity` handelt es sich um eine Spezifikation der von mehreren Anwendungen gemeinsam genutzten Daten und Informationen, also beispielsweise ein Geschäftsobjekt wie „Kunde“ mit seinen standardisierten Attributen und Operationen.

Die `InformationEntities` werden in sogenannten `AbstractApplications` gespeichert und von diesen auch genutzt. Durch eine `AbstractApplication` werden *„die gemeinsamen Eigenschaften [...] und Komponenten von Applikationen“* [Kir03, S. 78] gekapselt. Die Existenz dieser abstrakten Klasse rührt von Kirchners Klassifikation von Applikationen innerhalb seines Metamodells her: Eine `Application` ist ein selbstständig lauffähiges Softwareprodukt, bei `ApplicationComponent` handelt es sich um logische Funktionskomponenten innerhalb einer Anwendung. Die Beziehung `requires` drückt

aus, dass Anwendungen von anderen Anwendungen abhängig sein können. Eine Anwendung dient gemäß diesem Modell dazu, einen Geschäftsprozess zu unterstützen. Weiterhin stellt eine `Application` beziehungsweise eine `AbstractApplication` einen oder mehrere `Services` bereit und kann solche auch nutzen. Diese weisen jedoch keinen direkten Bezug zu einem `BusinessProcess` auf.

Durch logische Schnittstellen (`LogicalInterface`) sind Applikationen in der Lage, untereinander zu kommunizieren. Die Beziehung eines `LogicalInterface` zu der Klasse `Standard` zeigt auf, ob sich eine Anwendung an Standards hinsichtlich Protokollen und Austauschformarten hält (zum Beispiel XML[7] für den Datenaustausch).

Hinsichtlich der Hardware unterscheidet Kirchner zwischen `HardwareDevice` und `ExternalPhysicalInterface`. Ein `HardwareDevice` repräsentiert eine Hardware-Komponente der Systemlandschaft und kann aus anderen Hardware-Komponenten zusammengesetzt sein. Eine solche Hardware kann über eine Schnittstelle, das `ExternalPhysicalInterface`, mit anderen Hardware-Komponenten verbunden sein. Die Beziehung dieser Schnittstelle zu einem `LogicalInterface` sagt aus, dass die Schnittstelle gegebenenfalls an ein bestimmtes Protokoll gebunden ist. Eine `Application` und `HardwareDevice` ergeben zusammen eine `Platform`.

Als wichtiges Konzept geht die `Role` in das Modell ein. Eine `Role` ist eine repräsentiert einen Nutzer innerhalb der Systemlandschaft in Form von Rollenkonzepten, die auf allen Ebenen des Modells interagieren können.

Beispielszenario

Abbildung 3.8 zeigt das Ergebnis der Modellierung des Beispielszenarios gemäß dem Metamodell von Kirchner. Da es für diesen Ansatz kein Modellierungswerkzeug gibt, wurde das Metamodell durch eine eigene Notation auf Basis von UML umgesetzt.

In dem Modell wird Hardware durch Knoten, Software durch Komponenten repräsentiert. Die Beziehung `runs on` wird durch das Platzieren von Software innerhalb der Hardware realisiert. Die Abhängigkeit `requires` wird durch Pfeile (`--→`) dargestellt. `Redmine` stellt die beiden im Beispielszenario vorgestellten Services bereit, die von `ServiceCloud` und `Eclipse` genutzt werden. Die Workstation wird ebenfalls durch einen Knoten repräsentiert, auf dem `Eclipse` installiert ist. Der `ExternalSMTP` sowie `ServiceCloud` wurden ebenfalls als Software modelliert und kommunizieren über eine Firewall mit dem firmeninternen Netz.

Analyse

Da nun die Sprachbestandteile des Ansatzes von Kirchner vorgestellt und im Rahmen eines Beispielszenarios umgesetzt wurden, kann auf dieser Basis nun eine Analyse erfolgen.

Der Ansatz von Kirchner bietet keine Möglichkeiten hinsichtlich der Unterscheidung verschiedener Arten von Hardware (vergleiche Abbildung 3.7). So ist beispielsweise eine Unterscheidung von Servern und Betriebsgeräten, wie auch im Beispielszenario zuvor gesehen, nicht möglich. Eine Eindeutigkeit der Elemente ist in diesem Modellierungsansatz folglich nicht gegeben. Möglich ist es hingegen, den Aufbau von Hardware

[7] XML=Extensible Markup Language

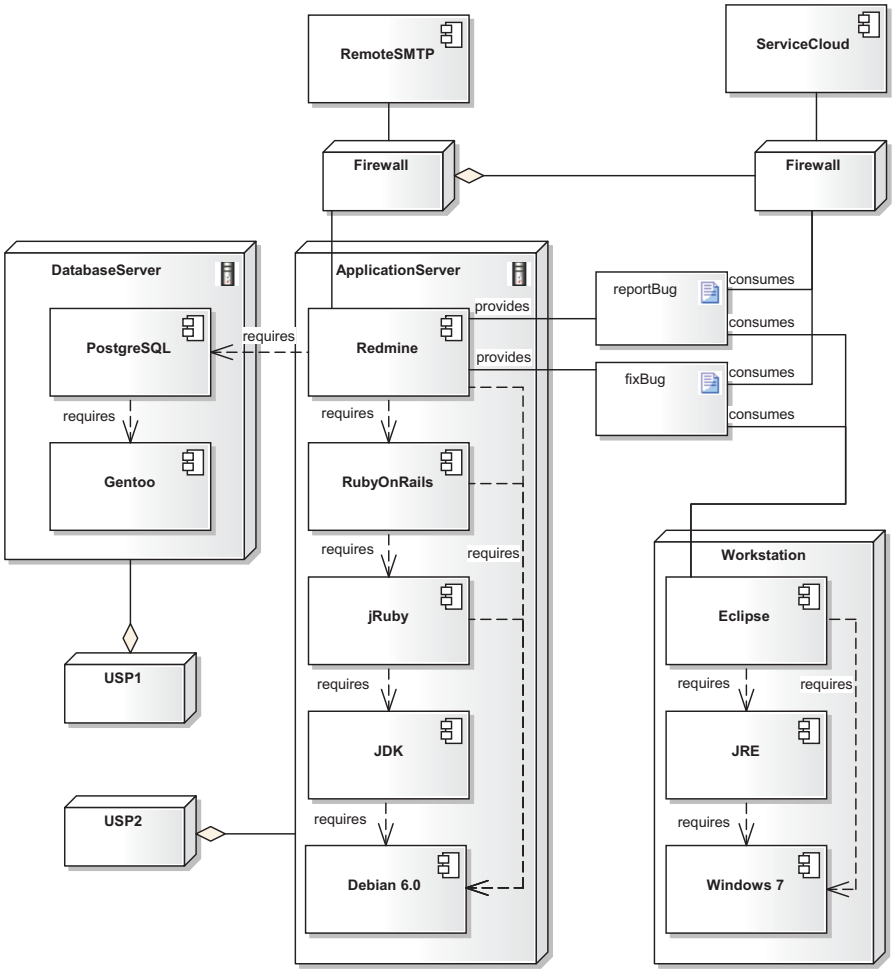

Abbildung 3.8: Beispielszenario: Modellierung von IT-Landschaften nach Kirchner

aus verschiedenen Bestandteilen über eine Aggregation abzubilden. Jedoch ist es nicht möglich, für Hardware eine Konfiguration anzugeben.

Software wird im Beispielszenario durch das UML-Notationselement für Komponenten dargestellt. Dabei zeigt sich, dass Kirchners Modellierungsansatz keine Unterscheidung von Software ermöglicht. Folglich können beispielsweise Systemsoftware und Anwendungssoftware nicht unterschieden werden. Das Beispielszenario macht auch deutlich, dass die verschiedenen Abhängigkeiten von Software stets einzeln anzugeben sind. Dies kann bei größeren Modellen schnell zu Unübersichtlichkeit führen. In Kirchners Modellierungsansatz ist es zudem möglich, dass jede Art von Software Services anbieten und nutzen kann. Services können jedoch nicht mit Service Level Agreements in Verbindung gesetzt werden, da diese in Kirchners Modell nicht berücksichtigt werden. Wie auch bei Hardware ist es zudem nicht möglich, Software zu konfigurieren.

Die Abbildung der verschiedenen Konzepte des Cloud Computings ist bei Kirchner nicht vorgesehen. Sie ist zwar durch die Nutzung anderer Konzepte prinzipiell möglich, aber spezielle Berücksichtigung findet Cloud Computing im Metamodell nicht.

In Tabelle 3.5 ist die Zusammenfassung der Ergebnisse der soeben vorgenommenen Analyse zu sehen.

Komponenten	**Kriterium**	**Analyse**
	Anwendungssoftware	✓
	Hardware	✓
	Systemsoftware	(✓)
	Geräte für Betrieb	(✓)
	Cloud Computing	(✓)
	SaaS	–
	IaaS	–
	PaaS	–
	Services	✓
	SLAs	–
Sprache / Werkzeug	**Kriterium**	**Analyse**
	Anzahl Elemente	13
	Eindeutigkeit	–
	Konfiguration	–
	Statisch/Dynamisch	✓ / –

Tabelle 3.5: Ergebnis der Analyse von Kirchners Ansatz

3.3.4 ARIS Express

ARIS Express ist ein frei verfügbares Modellierungswerkzeug der Software AG[8]. Neben der Modellierung von Organigrammen, Geschäftsprozessen und ganzer Prozesslandschaften stellt ARIS Express auch Möglichkeiten für die Modellierung der Systemlandschaft eines Unternehmens bereit. Als Möglichkeiten kommen hierbei grundsätzlich die Diagrammarten Geschäftsprozess, Systemlandschaft sowie IT-Infrastruktur in Frage. Da die Abbildung von IT-Komponenten innerhalb von Diagrammen für Geschäftsprozesse sehr beschränkt ist und der Schwerpunkt dieser Arbeit auf der Dimension „Technik“ liegt, werden im Folgenden lediglich die Diagramme „Systemlandschaft“ und „IT-Infrastruktur“ vorgestellt.

Notationselemente

Nachfolgend werden die Notationselemente der beiden Diagrammarten unterschieden, wobei eine getrennte Betrachtung erfolgt.

Systemlandschaft
Im Systemlandschafts-Diagramm werden einzelne IKS und die Domänen, denen sie zugeordnet sind, modelliert. Als Objekte stehen in diesem Diagramm das IT-System sowie Domänen zur Verfügung (vergleiche Abbildung 3.9).

Ein IT-System repräsentiert ein Anwendungssystem innerhalb der Systemlandschaft eines Unternehmens. Es wird bei diesem Diagrammtyp immer einer spezifischen Domäne zugeordnet. Eine Domäne repräsentiert eine logische Einheit eines Unternehmens, beispielsweise einen spezifischen Geschäftsbereich oder einen Prozess. So können Darstellungen von Systemlandschaften wie in Abbildung 3.9 entstehen: die Domäne „Anwen-

[8] http://www.softwareag.com

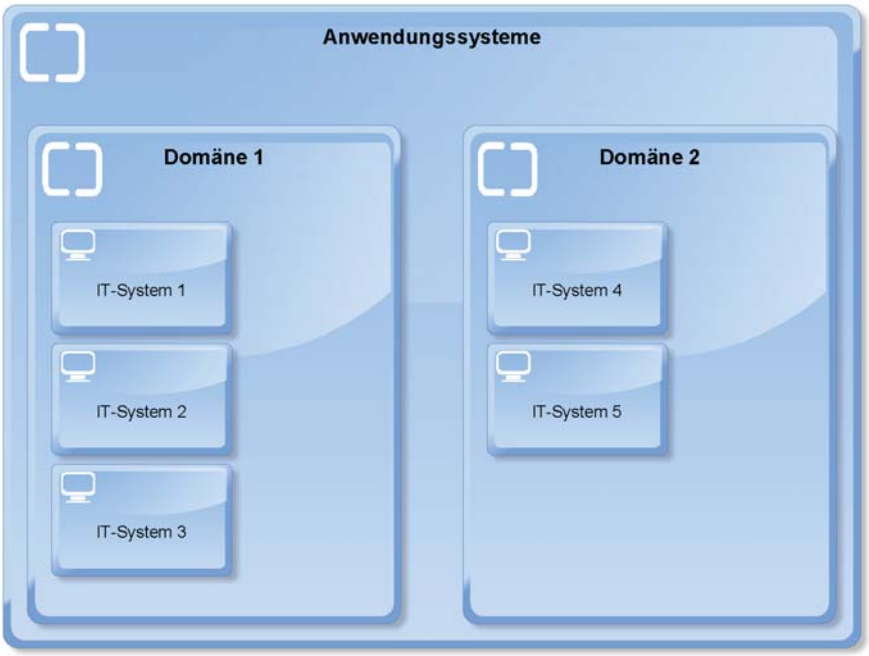

Abbildung 3.9: Notationselemente: ARIS Express, Diagramm „Systemlandschaft“

dungssysteme“ umfasst alle IT-Systeme des Unternehmens und gliedert sich in weitere Domänen, die wiederum die für die Domäne relevanten IT-Systeme umfassen.

IT-Infrastruktur
Das Diagramm IT-Infrastruktur wird für die Darstellung von Kommunikations- und Informationssytemlandschaften verwendet. Die dafür bereitgestellten Elemente sind in Abbildung 3.10 aufgeführt.

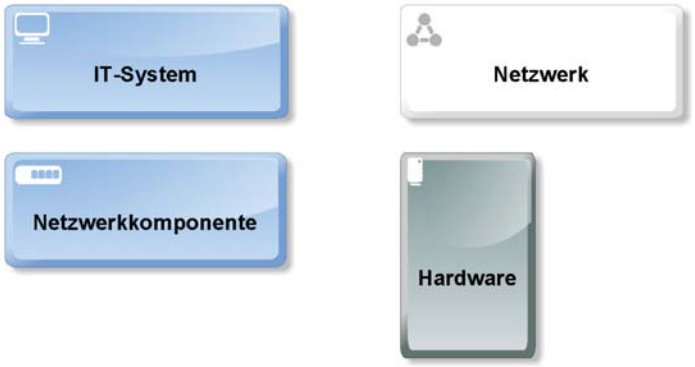

Abbildung 3.10: Notationselemente: ARIS Express, Diagramm „IT-Infrastruktur“

Innerhalb dieses Diagramms können einzelne Netze erstellt werden, die realweltliche Netzwerke repräsentieren (zum Beispiel „Internet“ und „Intranet“). Netze müssen dabei nicht isoliert sein, sondern können untereinander verbunden werden. Auch ist es möglich, Netze zu verschachteln und so Teilnetze zu bilden. Einem Netz lassen sich dabei bestimmte Netzwerkkomponenten zuordnen (zum Beispiel eine Firewall, welche die

Kommunikation zwischen zwei Netzen reguliert). Des Weiteren können IT-Systeme, die ebenso wie im Systemlandschafts-Diagramm Anwendungssysteme repräsentieren, abgebildet werden. Zudem besteht die Möglichkeit, Hardware zu modellieren (also beispielsweise Server innerhalb der Systemlandschaft).

Bei keinem der vorgestellten Objekte handelt es sich dabei um im Unternehmen eindeutig identifizierbare Objekte, sondern vielmehr um Typisierungen von Objekten, die über die gleiche technologische Basis verfügen.

Beispielszenario

Wie bereits zuvor dargelegt, ist eine Modellierung des Beispielszenarios mit Hilfe der Diagramme „Systemlandschaft“ und „IT-Infrastruktur“ möglich. Daher wird das Beispielszenario auch an dieser Stelle mittels beider Diagramme umgesetzt.

In Abbildung 3.11 ist zunächst die Modellierung des Beispielszenarios mit Hilfe des ARIS-Diagramms für Systemlandschaften zu sehen.

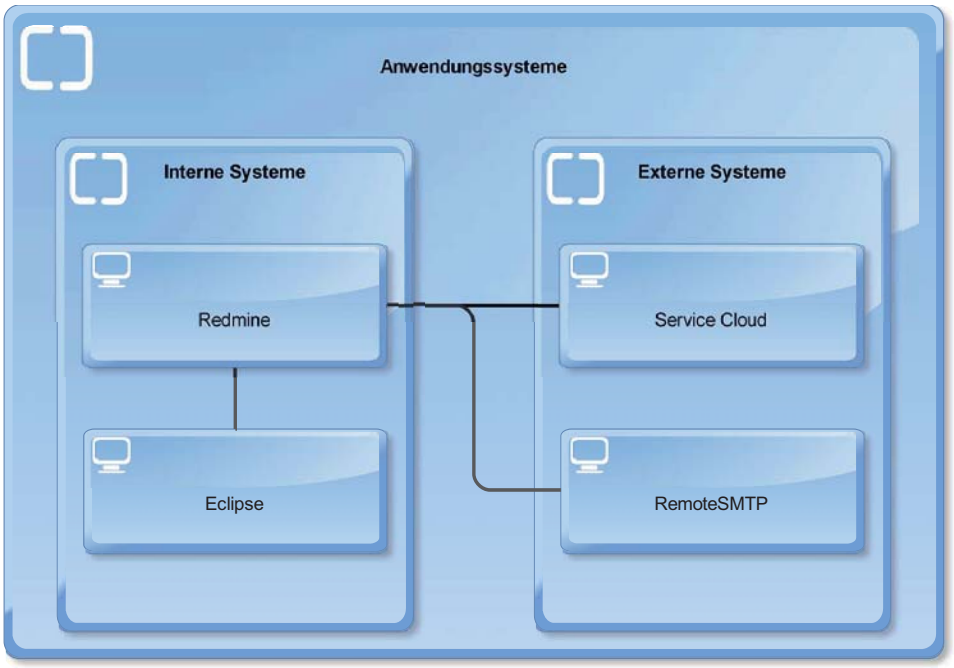

Abbildung 3.11: Beispielszenario: ARIS Express (Systemlandschaft)

Die einzelnen Softwaresysteme des Beispielszenarios sind in interne und externe IT-Systeme aufgeteilt, um das Vorhandensein verschiedener Netzwerke darzustellen. Die Relationen zwischen den einzelnen IT-Systemen zeigen deren Kommunikationsbeziehungen auf. Wie die Abbildung zeigt, ist diese Art der Darstellung wenig detailliert. Allerdings ist das Diagramm gut geeignet, um einen groben Überblick über eine Systemlandschaft zu geben.

Abbildung 3.12 zeigt die Umsetzung des Beispielszenarios mit Hilfe des Diagramms für die Darstellung von IT-Infrastrukturen, die im Vergleich zum Systemlandschafts-Diagramm wesentlich detaillierter ist.

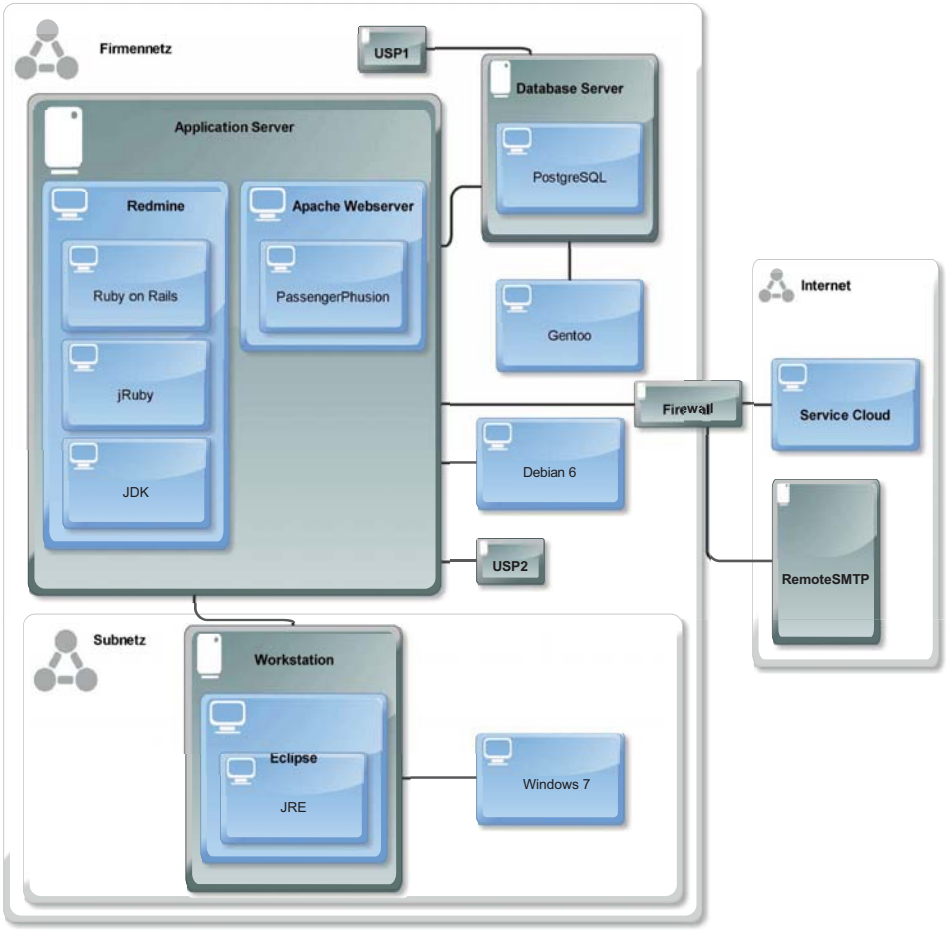

Abbildung 3.12: Beispielszenario: ARIS Express (IT-Infrastruktur)

Sowohl der Anwendungs- als auch der Datenbank-Server durch Hardware dargestellt, ebenso die ihnen zugeordnete unterbrechungsfreie Stromversorgung. Beide Server liegen zusammen mit der Workstation, die ebenfalls durch Hardware repräsentiert wird, innerhalb eines Netzwerkes, welches nur über eine Firewall mit Systemen außerhalb dieses Netzwerkes kommunizieren kann. Diese wird ebenfalls durch Hardware repräsentiert. Die einzelnen Systeme, die auf den Servern beziehungsweise auf der Workstation betrieben werden, sind in Abbildung 3.12 als IT-Systeme dargestellt. Abhängigkeiten zu anderen Systemen sind dadurch modelliert, dass die für die Ausführung eines Systems benötigten IT-Systeme in dieses System eingebettet sind. Eine Ausnahme bilden die Betriebssysteme, die über eine Relation zur Hardware dargestellt sind, um die Übersichtlichkeit zu erhöhen.

Der externe Mail-Server wurde als Hardware modelliert und liegt zusammen mit Service Cloud, welches als IT-System modelliert ist, innerhalb des externen Netzwerks „Internet“.

Analyse

Zum Abschluss der Betrachtung von ARIS Express gilt es nun, auf Basis der Vorstellung seiner Elemente und der Umsetzung des Beispielszenarios, eine Analyse durchzuführen.

Komponenten	**Kriterium**	**Analyse**
	Anwendungssoftware	✓
	Hardware	✓
	Systemsoftware	(✓)
	Geräte für Betrieb	(✓)
	Cloud Computing	(✓)
	SaaS	–
	IaaS	–
	PaaS	–
	Services	–
	SLAs	–
Sprache / Werkzeug	**Kriterium**	**Analyse**
	Anzahl Elemente	5
	Eindeutigkeit	–
	Konfiguration	–
	Statisch/Dynamisch	✓ / –

Tabelle 3.6: Ergebnis der Analyse von ARIS Express

Wie bereits zuvor dargelegt, sind im Rahmen dieser Arbeit im Wesentlichen zwei Diagrammarten von ARIS Express von Bedeutung: Das Systemlandschafts-Diagramm sowie das Infrastruktur-Diagramm. Beide nutzen dabei weitestgehend die gleiche Notation. Die Abbildung von Hardware wird durch das Notationselement `Hardware` realisiert. Eine Differenzierung von Hardware in Server und Betriebsgeräte ist jedoch nicht möglich. Gleiches gilt für Software: Auch hier müssen sowohl Anwendungen als auch Systemsoftware durch das Notationselement `IT-System` abgebildet werden, da eine granulare Differenzierung von Software nicht vorgesehen ist. Weiterhin kann weder für Hardware noch für Software eine Konfiguration angegeben werden.

Ebenso lässt ARIS Express eine Möglichkeit zur Abbildung von Konzepten des Cloud Computings vermissen. Zwar wurde versucht, dies im Beispielszenario durch die Nutzung von `Hardware` sowie `IT-Systemen` zu kompensieren. Jedoch ist eine explizite Abbildung mit dem Werkzeug nicht möglich. Weiterhin können mit ARIS Express keine Services modelliert werden. Zudem mangelt es an der Fähigkeit, SLAs in Modellen zu berücksichtigen.

Tabelle 3.6 fasst die im Rahmen der Analyse von ARIS Express gewonnenen Kenntnisse zusammen.

3.3.5 Weitere Ansätze zur Modellierung von Systemlandschaften

Neben den zuvor untersuchten Ansätzen existiert eine ganze Reihe von Sprachen und Werkzeugen, die ebenfalls das Modellieren von Systemlandschaften ermöglichen. Einige Beispiele hierfür sind:

- *Common Infrastructure Model* (**CIM**)[9], ein Standard für das Management von IT-Systemen.
- *ARIS IT Architect*[10], ein Tool der Software AG für ein Enterprise Architecture Management.
- *IBM Topology Editor*[11], ein Werkzeug zur Überführung von Software aus einer logischen Sicht in eine technische IT-Infrastruktur.
- *The Open Group Architecture Framework* (TOGAF)[12], eine Methode zur Modellierung von IT-Architekturen.
- *Architecture Analysis & Design Language* (AADL)[13], die formale Konzepte zur Beschreibung und Analyse von Systemarchitekturen bereitstellt.

Diese Ansätze sind für KMU aufgrund ihrer Komplexität oder der Höhe der Investitionskosten jedoch als ungeeignet anzusehen. Eine eingehende Untersuchung wurde daher nicht vorgenommen, jedoch können bestimmte Aspekte dieser Ansätze im weiteren Verlauf der Arbeit Berücksichtigung finden.

3.3.6 Abschließender Vergleich

Nachdem nun die einzelnen Modellierungsansätze untersucht wurden, werden diese abschließend gegenübergestellt und die jeweiligen Stärken und Schwächen der einzelnen Ansätze herausgestellt. Tabelle 3.7 zeigt eine Zusammenfassung der vorausgegangenen Analysen.

Tabelle 3.7 macht deutlich, dass keiner der vorgestellten Ansätze alle der in Abschnitt 3.1 vorgestellten Kriterien erfüllt. Besonders hinsichtlich der Konzepte des Cloud Computings bietet keiner der betrachteten Ansätze eine befriedigende Abbildungsmöglichkeit. Dies gilt auch für die Berücksichtigung von Services und, damit verbunden, von Service Level Agreements. Zwar mangelt es lediglich ARIS Express an einer Möglichkeit zur Modellierung von Services, jedoch bietet keiner der vorgestellten Ansätze ein Konzept an, das ein zufriedenstellendes Modellieren sowohl von SLAs als auch von Services gestattet und diese zudem angemessen in die Systemlandschaft zu integrieren. Service Level Agreements finden gar nur in der ITML Berücksichtigung. Jedoch zeigt sich und auch

[9] http://dmtf.org/standards/cim

[10] http://www.softwareag.com/de/products/aris_platform/aris_design/it_architect/overview/default.asp

[11] http://www-01.ibm.com/software/rational/

[12] http://www.togaf.org/

[13] http://www.aadl.info/aadl/currentsite/

		ITML	Deployment Diagram	Kirchner	ARIS Express
Komponenten	**Kriterium**				
	Anwendungssoftware	✓	✓	✓	✓
	Hardware	✓	✓	✓	✓
	Systemsoftware	(✓)	(✓)	(✓)	(✓)
	Geräte für Betrieb	(✓)	(✓)	(✓)	(✓)
	Cloud Computing	(✓)	(✓)	(✓)	(✓)
	SaaS	–	–	–	–
	IaaS	–	–	–	–
	PaaS	–	–	–	–
	Services	✓	(✓)	✓	–
	SLAs	(✓)	–	–	–
Sprache / Werkzeug	**Kriterium**				
	Anzahl Elemente	15	8	13	5
	Eindeutigkeit	–	–	–	–
	Konfiguration	(✓)	(✓)	–	–
	Statisch/Dynamisch	✓ / –	✓ / –	✓ / –	✓ / –

Tabelle 3.7: Zusammenfassung der Analyse

hier, dass dies nur in einem unzureichenden Maße geschieht (vergleiche Abschnitt 3.3.2). Tabelle 3.7 macht deutlich, dass keiner der vorgestellten Ansätze alle der in Abschnitt 3.1 vorgestellten Kriterien erfüllt. Besonders hinsichtlich der Konzepte des Cloud Computings bietet keiner der betrachteten Ansätze eine befriedigende Abbildungsmöglichkeit. Dies gilt auch für die Berücksichtigung von Services und, damit verbunden, von Service Level Agreements. Zwar mangelt es lediglich ARIS Express an einer Möglichkeit zur Modellierung von Services, jedoch bietet keiner der vorgestellten Ansätze ein Konzept an, das ein zufriedenstellendes Modellieren sowohl von SLAs als auch von Services gestattet und diese angemessen in die Systemlandschaft zu integrieren. Service Level Agreements finden gar nur in der ITML Berücksichtigung. Jedoch zeigt sich und auch hier, dass dies nur in einem unzureichenden Maße geschieht (vergleiche Abschnitt 3.3.2).

Wie sich Tabelle 3.7 weiterhin entnehmen lässt, mangelt es jedem der vorgestellten Ansätze an Eindeutigkeit hinsichtlich der Abbildung der Elemente einer Systemlandschaft. Natürlich stellt sich vor dem Hintergrund von KMU, in denen es häufig an Fachkenntnissen bezüglich der IT mangelt, die Frage, ob eine Eindeutigkeit wirklich gegeben sein muss. Schließlich würde Eindeutigkeit eine höhere Anzahl von Elementen des jeweiligen Ansatzes verlangen und folglich auch die Komplexität erhöhen. Eine grundlegende, grobe Unterscheidung von einzelnen Elementen erscheint jedoch wünschenswert, um die Aussagekraft des Modells zu erhöhen. In diesem Fall könnte es zumindest ermöglicht werden, beispielsweise zwischen einem Server und einem Gerät für den Betrieb der In-

frastruktur zu unterscheiden. Die Analyse hat weiterhin gezeigt, dass alle vorgestellten Ansätze zwar den statischen Charakter einer Systemlandschaft, nicht aber ihre Dynamik abbilden können. Wie jedoch in Abschnitt 2.1.3 gezeigt wurde, ist die Dynamik ein wesentliches Merkmal von Systemlandschaften. Natürlich ist deren Erfassung in einem statischen Modell schwierig, kann jedoch in einem gewissen Maße durch die Berücksichtigung von verschiedenen Konfigurationen von Hardware und Software sowie durch die Dokumentation dieser Änderungen zumindest ansatzweise aufgezeigt werden.

Obwohl der Fokus dieser Arbeit auf der technischen Dimension von Systemlandschaften liegt, ist dies natürlich nicht die einzige Sicht auf eine Systemlandschaft (vergleiche Abschnitt 2.1.3). Somit erscheint es sinnvoll, auch die anderen Dimensionen einer Systemlandschaft zumindest zu berücksichtigen. Die Analyse hat jedoch gezeigt, dass lediglich die ITML sowie der Ansatz von Kirchner andere Dimensionen berücksichtigen. Die anderen Ansätze beschränken sich hingegen auf technische Aspekte. Etwas anders stellt sich die Situation bei ARIS Express dar: Mit seinen weiteren Diagrammarten, beispielsweise dem für Geschäftsprozesse, können auch andere Dimensionen einer Systemlandschaft abgebildet werden. Vor allem auch, weil ARIS Express die Möglichkeit bieten, verschiedene Diagramme semantisch zu verknüpfen. Da im Kontext dieser Arbeit jedoch keine Analyse der anderen Diagrammarten erfolgt ist, kann an dieser Stelle keine verlässliche Aussage zu den Fähigkeiten von ARIS Express hinsichtlich der Abbildung anderer Dimensionen einer Systemlandschaft getroffen werden.

Die einzelnen Ansätze sind jedoch nicht nur negativ zu bewerten, sondern jeder Ansatz hat auch eigene Stärken:

- Bei der ITML sind vor allem das Vorhandensein von IT-Services sowie in Verbindung damit die SLAs als Stärken anzusehen. Dies gilt ebenso für die Möglichkeit, einzelne Netzwerke zu definieren.
- Die größte Stärke des Deployment Diagrams ist die `executionEnvironment`, da dies die Übersichtlichkeit des Modells stark erhöht. Zudem sind die `devices` als Sonderfall der Knoten positiv zu erwähnen.
- Bei Kirchners Ansatz ist der Aufbau von Software aus einzelnen Komponenten positiv zu bewerten. Zudem werden in seinem Modell Geschäftsobjekte (`InformationEntity`) berücksichtigt. Zudem ist der Ansatz von Kirchner der einzige, der auch die Dimension „Mensch“ einer Systemlandschaft durch sein Rollenkonzept berücksichtigt.
- Als Stärke von ARIS Express können die Unterteilung der Systemlandschaft in einzelne Netzwerke sowie das Bilden von Subnetzen innerhalb dieser Netzwerke hervorgehoben werden.

Diese Stärken können natürlich nicht außer Acht gelassen und müssen folglich im zu entwickelnden Metamodell berücksichtigt werden. Abschließend kann nun festgehalten werden, dass vor dem Hintergrund, dass keine der vorgestellten Sprachen den in Abschnitt 3.1 vorgestellten Kriterien genügt, ein Bedarf an einem Modellierungsansatz für Systemlandschaften in KMU besteht.

3.4 Zusammenfassung

Im Rahmen dieses Kapitels wurde eine Analyse existierender Modellierungsansätze für Systemlandschaften mit dem Fokus auf die Dimension „Technik“ vorgenommen. Zur Gewährleistung einer konsistenten Analyse wurden in Abschnitt 3.1 zunächst Untersuchungskriterien aufgestellt, auf deren Basis die Untersuchung durchgeführt wurde. Im Anschluss daran wurde in Abschnitt 3.2 ein Beispielszenario aufgestellt, welches einen Rahmen für die Analyse geschaffen hat. Darauf aufbauend wurden in Abschnitt 3.3 verschiedene Modellierungsansätze untersucht. Zunächst wurde jeder Ansatz einzeln beschrieben und anschließend das Beispielszenario aus Abschnitt 3.2 umgesetzt. Auf dieser Basis erfolgte anschließend die Analyse. In deren Rahmen wurde festgestellt, dass vor allem hinsichtlich der Berücksichtigung von Konzepten des Cloud Computings der Entwurf eines für KMU geeigneten Modellierungsansatzes alternativlos ist. Die Ergebnisse der Analyse, welche die Stärken und Schwächen der untersuchten Ansätze aufgezeigt hat, können nun in die Entwicklung des zu entwickelnden Metamodells einfließen.

Kapitel 4

Entwurf eines Metamodells

In Kapitel 3 wurden verschiedene Modelle und Werkzeuge für die Abbildung von Systemlandschaften analysiert. Dabei wurde festgestellt, dass keiner der untersuchten Ansätze den Anforderungen an die Modellierung von Systemlandschaften (vergleiche Abschnitt 3.3) in KMU genügt. Daher wird im Rahmen dieses Kapitels ein Metamodell für die Abbildung von Systemlandschaften erarbeitet, wobei der Fokus auf der Dimension „Technik“ liegt.

Abschnitt 4.1 stellt einleitend Anforderungen an das zu entwickelnde Metamodell vor. Dabei werden, neben allgemeinen Anforderungen an Modelle, vor allem auch Anforderungen in Bezug auf Systemlandschaften berücksichtigt. Zudem finden insbesondere Anforderungen bezüglich KMU Berücksichtigung.

Nachdem die Anforderungen definiert sind, erfolgt in Abschnitt 4.2 der Entwurf des Metamodells. Dazu werden zunächst die einzelnen Bestandteile des Modells vorgestellt und anschließend deren Zusammenhänge aufgezeigt.

In Abschnitt 4.3 wird eine Evaluierung des entwickelten Ansatzes einerseits auf Basis der Analysekriterien aus Abschnitt 3.1 vorgenommen. Andererseits werden die Anforderungen aus Abschnitt 4.1 für die Untersuchung herangezogen. Eine Zusammenfassung schließt die Betrachtung des entwickelten Metamodells und reflektiert das Vorgehen.

4.1 Anforderungen an Modelle & Modellierungssprachen

Bevor mit der Entwicklung des Metamodells begonnen werden kann, müssen hierfür zunächst Anforderungen definiert werden. Diese gliedern sich in drei verschiedene Kategorien. Die erste Kategorie umfasst allgemeine Anforderungen an Modellierungssprachen beziehungsweise an Modelle und damit auch an Metamodelle (siehe [FvL03]). Die zweite Kategorie betrifft die Anforderungen an eine Modellierungssprache hinsichtlich ihrer Fähigkeit zur Abbildung von Systemlandschaften, die in Anlehnung an [Kir03] aufgestellt wurden. Die dritte und letzte Kategorie befasst sich mit allen KMU-spezifischen Anforderungen (vergleiche Abschnitt 2.3.3). Eine Zusammenstellung aller Anforderungen mitsamt einer kurzen Erläuterung ist Tabelle 4.1 zu entnehmen.

Allgemeine Anforderungen	
Anwenderbezogene Anforderungen	
Einfachheit	Nach Frank und van Laak definiert sich Einfachheit einer Modellierungssprache über die Zahl ihrer Konzepte und zudem über die sie darstellenden Symbole [FvL03].
Verständlichkeit und Anschaulichkeit	Die Einfachheit einer Sprache in Verbindung mit der Ähnlichkeit der Sprache zu denjenigen Sprachen, die dem Anwender bereits bekannt sind.
Anwendungsbezogene Anforderungen	
Mächtigkeit und Angemessenheit	Alle relevanten Belange der Domäne sollten modelliert werden können.
Operationalisierbarkeit	Sprache dient nicht nur der Visualisierung, sondern auch bietet auch eine Nutzbarkeit hinsichtlich mathematischer Operationen (zum Beispiel Analyse bestimmter Sachverhalte).
Anforderungen hinsichtlich Systemlandschaften	
Flächendeckende Abbildung	Abbildung von Hard- und Software, Berücksichtigung von Geschäftsprozessen, Berücksichtigung von Verbindungen von Geschäftsprozessen mit Hard- und Software, Berücksichtigung von Schnittstellen zu einer Sprache für Geschäftsprozessmodellierung, Wahl geeigneter Attribute. Die Basis dafür bildet die Definition von Systemlandschaften.
Möglichkeiten zur Integration berücksichtigen	Anhand des Modells kann Integrationsbedarf erkannt werden und Möglichkeiten zur Abbildung von Integration werden geboten.
Anforderungen hinsichtlich KMU	
Einfachheit	Äquivalent zur „Einfachheit“ der anwenderbezogene Anforderungen.
Cloud Computing	Da Integrationsbedarf auch über die Unternehmensgrenzen hinweg immer weiter steigt und IT-Budgets sinken, sind Methoden der Integration und preisgünstigere Modelle für KMU (beispielsweise Software as a Service) in dem Metamodell zu berücksichtigen.

Tabelle 4.1: Anforderungen an das zu entwickelnde Metamodell

Tabelle 4.1 enthält zwar Überschneidungen von Anforderungen in verschiedenen Kategorien (beispielsweise „Einfachheit“ sowohl bei anwenderbezogenen Anforderungen als auch bei KMU-Anforderungen), was jedoch beabsichtigt ist: Sie sind nicht als Redundanz zu verstehen, sondern zeigen vielmehr die Wichtigkeit der Anforderungen bezüglich verschiedener Aspekte auf, was insbesondere die Einfachheit des Modells betrifft. Die vorgestellten Anforderungen bilden daher eine wesentliche Grundlage für die Gestaltung des Metamodells.

In Tabelle 4.1 fehlen die *formalen Anforderungen*, wie sie in [FvL03] aufgestellt wurden. Deren Berücksichtigung ist im Kontext dieser Arbeit nicht sinnvoll, da diese nach Frank und van Laak vor allem dann von Bedeutung sind, wenn *„eine (maschinelle)*

Überprüfung der Integrität von Modellen, die Transformation von Modellen oder die Berechnung von Modelleigenschaften erwünscht sind“ [FvL03, S. 25]. Dies ist im Kontext dieser Arbeit jedoch nicht der Fall, weshalb eine Betrachtung der formalen Anforderungen nicht erfolgt.

4.2 Vorstellung des entwickelten Metamodells

Nachdem alle Anforderungen an eine Abbildung von Systemlandschaften definiert wurden, kann nun das entwickelte Metamodell vorgestellt werden.

Um ein vollständiges Verständnis des Modells zu gewährleisten, wird dieses zunächst in Form einzelner Klassen vorgestellt. Diese Klassen bilden bestimmte semantische Gruppen von Elementen des Metamodells ab. Zunächst wird die Klasse *Software* vorgestellt, die alle Entitäten beinhaltet, die sich auf verschiedene Arten von Software beziehungsweise damit in Verbindung stehenden Konzepten bezieht. Darauf aufbauend werden die Hardware-Elemente des Modells (Klasse *Hardware*) vorgestellt. Anschließend werden die Teile des Modells eingeführt, die sich auf die Konzepte des Cloud Computings (Klasse *Cloud Computing*) beziehen. Zudem werden die Entitäten vorgestellt, die in keine der drei Klassen eingeordnet werden können. Dabei handelt es sich um rudimentär berücksichtigte Entitäten bezüglich der Dimensionen „Mensch“ und „Aufgabe“ sowie um Konzepte zur Berücksichtigung der dynamischen Eigenschaften einer Systemlandschaft. Bei jeder der Klassen werden zunächst die Entitäten dieser Klasse vorgestellt und anschließend deren Verbindungen aufgezeigt. Neben der Vorstellung des Metamodells wird für jede Entität zudem eine exemplarische Attributierung auf Basis des Common Information Model [Tas11] vorgeschlagen. Diese beschränkt sich jedoch auf grundlegende Charakteristika der einzelnen Entitäten, um die Komplexität des Modells möglichst gering zu halten.

In Abbildung 4.1 ist eine schematische Darstellung aufgeführt, die einen groben Überblick über das Metamodell geben soll.

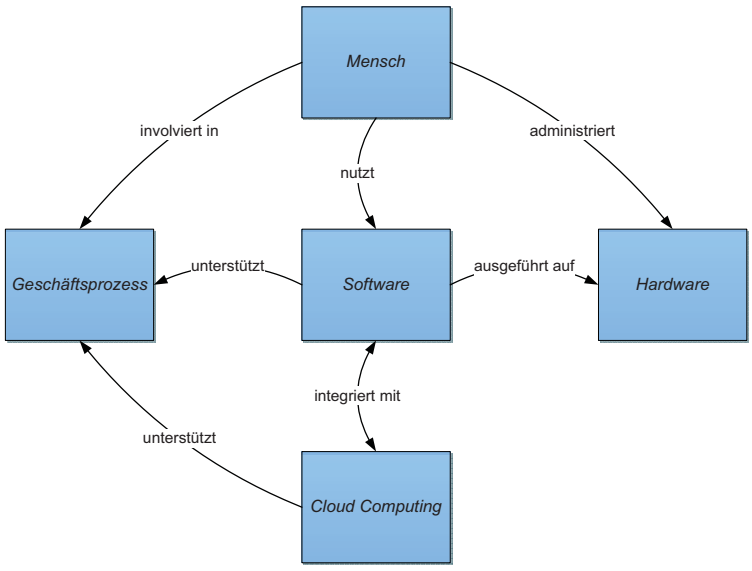

Abbildung 4.1: Schematische Darstellung des Metamodells

Da eine Vorstellung des gesamten Metamodells an dieser Stelle, wie bereits zuvor erwähnt, zu umfangreich wäre, werden im Rahmen dieses Kapitels lediglich die einzelnen Klassen sowie die Relationen zwischen den Entitäten der Klassen aufgezeigt. Für eine vollständige Darstellung des Metamodells wird auf Abbildung A.1 verwiesen.

4.2.1 Software

Wie im Rahmen der Analyse in Kapitel 3 gezeigt wurde, konnte keiner der untersuchten Ansätze den Anforderungen an die Abbildung von Softwarekomponenten einer Systemlandschaft vollständig genügen. Schwächen zeigten sich dabei insbesondere hinsichtlich der Unterscheidung verschiedener Arten von Software sowie bezüglich der Abbildung der Abhängigkeiten von Software untereinander. Bei der Entwicklung des Metamodells wurde das Augenmerk daher insbesondere auf die Beseitigung dieser Schwächen gelegt.

Entitäten

Die Klasse *Software* unterscheidet verschiedene Arten von Konzepten, die mit Software in Verbindung stehen. Jede Entität bezieht sich dabei auf eine bestimmte Art von Software oder dient der Vereinfachung bestimmter Gegebenheiten hinsichtlich der Modellierung von Software. In Tabelle 4.2 ist eine Auflistung der einzelnen Entitäten zu finden. Zudem ist der Tabelle eine kurze Beschreibung aller Entitäten zu entnehmen.

Entität	**Erläuterung**
Software	Abstrakte Klasse, welche die generischen Eigenschaften von Software kapselt.
SoftwareComponent	Stellt eine Komponente / ein Modul von Software dar.
ApplicationSoftware	Eine explizite Anwendung der abstrakten Klasse *Software*, die Anwendungssoftware abbildet.
SystemSoftware	Eine explizite Anwendung der abstrakten Klasse *Software*, die Systemsoftware abbildet.
SoftwareEnvironment	Gruppierung von Systemsoftware, um die Abbildung von Abhängigkeiten zwischen Anwendungssoftware und Systemsoftware zu vereinfachen.
Service	Fest definierte Leistung, die als Element eines oder mehrerer größerer Verarbeitungsabläufe verwendet werden kann und von *ApplicationSoftware* bereitgestellt oder genutzt wird.
SLA	Service Level Agreement, das die Rahmenbedingungen für einen Service festlegt.

Tabelle 4.2: Entitäten der Klasse „Software“

Eine genaue Übersicht der Entitäten samt deren Relationen kann Abbildung 4.2 entnommen werden.

`Software` ist, wie bereits in der Erläuterung in Tabelle 4.2 dargelegt, eine abstrakte Klasse, welche generische Eigenschaften aller Arten von Software kapselt. So beschreibt diese Entität beispielsweise, dass Software aus mehreren Komponenten beziehungsweise

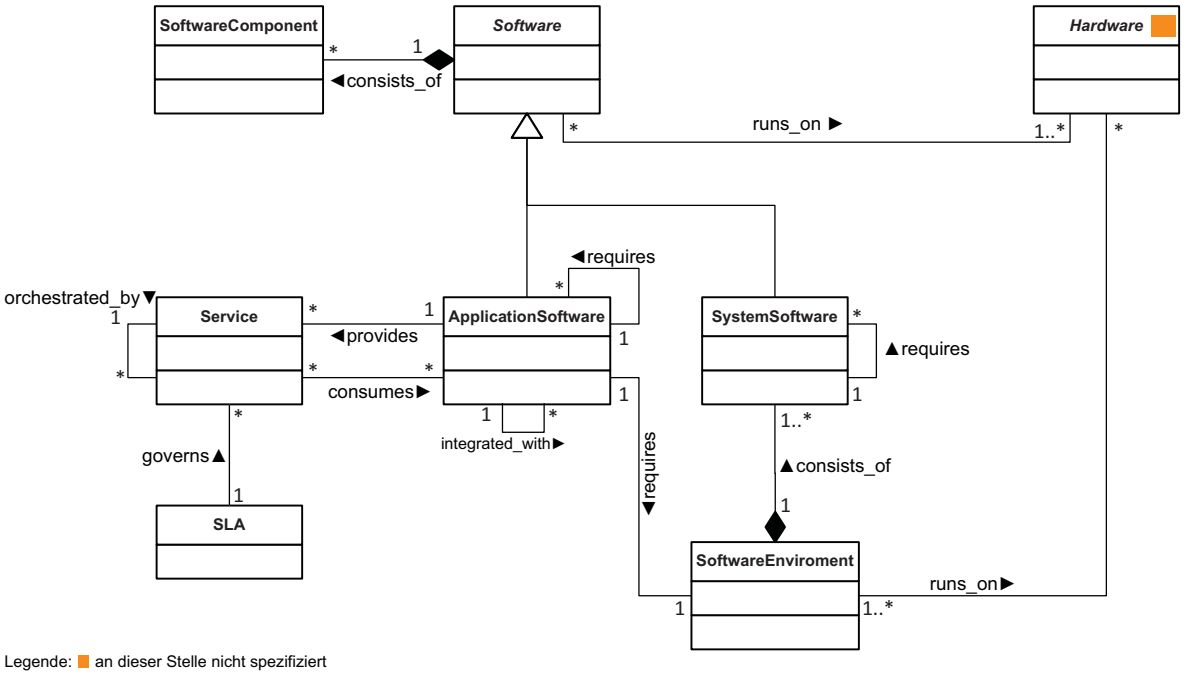

Abbildung 4.2: Entitäten der Klasse „Software“ und Relationen

Modulen (`SoftwareComponent`) aufgebaut sein kann (ausgedrückt durch die Beziehung `consists_of`) und das Software auf Hardware ausgeführt wird. Zudem umfasst diese abstrakte Klasse einige Attribute, die Software im Allgemeinen charakterisieren. Derartige (denkbare) Attribute werden im nachfolgenden Abschnitt vorgestellt.

Konkrete Realisierungen von `Software` stellen Anwendungssoftware (`ApplicationSoftware`) und Systemsoftware (`SystemSoftware`) dar. Bei Anwendungssoftware handelt es sich um Software, die zur Erfüllung einer bestimmten (betrieblichen) Aufgabe eingesetzt wird. Unter Systemsoftware ist Software zu verstehen, die für den operativen Betrieb benötigt wird (zum Beispiel ein Betriebssystem). Es kann sowohl `ApplicationSoftware` von anderer `ApplicationSoftware` abhängig sein, als auch `SystemSoftware` von anderer `SystemSoftware`. Anwendungssoftware ist dabei in der Lage, die von ihr gebotenen Funktionalitäten in Form von `Services`, wie sie in Abschnitt 2.4.1 charakterisiert wurden, bereitzustellen (Beziehung `provides`). Anwendungssoftware kann zudem auf definierte Services zugreifen und deren Funktionalität nutzen (Beziehung `consumes`). Reguliert wird ein Service durch ein Service Level Agreement (`SLA`). Ein Service muss jedoch nicht zwangsläufig durch Anwendungssoftware bereitgestellt werden. Ein neuer `Service` kann auch durch Orchestrierung mehrerer bereits vorhandener Services geschaffen werden (ausgedrückt durch die Beziehung `orchestrated_by`). Orchestrierung bezeichnet dabei eine Möglichkeit zur Kompositionen von Services.

Die Aufnahme von `ApplicationSoftware`, `SystemSoftware`, `Service` und `SLA` in das Metamodell liegt in dem Bestreben, Mehrdeutigkeiten bei der Abbildung der Domäne zu vermeiden, begründet. Die Entität `SoftwareEnvironment` repräsentiert dagegen kein Objekt, dass sich tatsächlich in einer Systemlandschaft wiederfindet, sondern ist ein Konzept zur Vereinfachung der Abbildung von Beziehungen zwischen Entitäten. Konkret dient es dazu, die Abhängigkeiten von `ApplicationSoftware` zu `SystemSoftware`

dahingehend zu vereinfachen, dass `ApplicationSoftware` nur noch von einer Menge von `SystemSoftware` abhängig ist und die einzelnen Abhängigkeiten nicht mehr aufgezeigt werden müssen. Wie im Rahmen der Analyse in Kapitel 3 gesehen, ist das Abbilden von Abhängigkeiten hinsichtlich der Übersichtlichkeit ein Problem fast aller untersuchten Ansätze. Daher wurde `SoftwareEnvironment` als Sammlung von Abhängigkeiten in das Metamodell aufgenommen. Folglich besteht auch keine direkte Abhängigkeitsbeziehung zwischen Anwendungssoftware und Systemsoftware. Stattdessen wird dies durch die Beziehung `requires` zwischen `ApplicationSoftware` und `SoftwareEnvironment` realisiert. Eine weitere wichtige Relation von `ApplicationSoftware` ist `integrated_with`: diese Beziehung dient dazu, die Kommunikation von zwei Systemen abzubilden, die nicht mittels Services kommunizieren, sondern dies auf direkte Weise tun. Diese „hohe Kopplung“ stellt zwar ein häufiges Problem in Systemlandschaften dar (vergleiche Abschnitt 2.1.3), wurde aber dennoch in das Metamodell aufgenommen, da die Nutzung von Services häufig nicht vollständig etabliert ist.

Von wesentlicher Bedeutung für die Entitäten der Klasse Software ist `Hardware`. Zwar ist Hardware kein Bestandteil der Klasse *Software* und wird daher an dieser Stelle auch nicht erläutert (siehe dafür Abschnitt 4.2.2), spielt jedoch für den Betrieb von Software eine entscheidende Rolle, da Software auf Hardware ausgeführt wird (Beziehung `runs_on`). Dies gilt ebenso für die Entität `SoftwareEnvironment`.

Attributierung

Tabelle 4.3 zeigt eine mögliche Attributierung der einzelnen Entitäten der Klasse *Software*. Die Attribute basieren dabei auf CIM, dem jedoch nur solche Attribute entnommen wurden, die für die einzelnen Entitäten des Metamodells maßgeblich sind. Die Auflistung erhebt folglich keinen Anspruch auf Vollständigkeit, sondern ist vielmehr als grundsätzlicher Vorschlag zu verstehen.

Entität	**Attribut**	**Beschreibung**
Software	`Name`	Bezeichnung der Software.
	`Version`	Aktuelle Versionsnummer der Software
	`Manufacturer`	Hersteller der Software.
	`Licence_Date`	Datum, an dem die Lizenz abläuft.
	`Serial`	Seriennummer der Software.
SoftwareComponent	`Name`	Bezeichnung der Komponente.
	`Version`	Aktuelle Versionsnummer der Software.
	`Manufacturer`	Hersteller der Komponente.
ApplicationSoftware	–	–
SystemSoftware	–	–
SoftwareEnvironment	–	–
Service	`Operations`	Liste von Funktionen, die der Service anbietet.
SLA	–	–

Tabelle 4.3: Mögliche Attributierung der Entitäten der Klasse „Software“

Die Attribute von `Software` sowie von `SoftwareComponent` sind CIM entnommen. Da für Anwendungs- und Systemsoftware durch die Vererbungsbeziehung zu `Software`

die charakteristischen Attribute bereits definiert sind und die Komplexität des Modells nicht zu sehr erhöht werden soll, ist eine weitere Ausgestaltung der Attribute an dieser Stelle nicht sinnvoll.

Die Ausprägung eines `Service` ist stark vom tatsächlichen Kontext anhängig. Somit kommt als Attribut lediglich eine Liste der von ihnen angebotenen Operationen in Frage. Es sind zwar auch weitere Attribute denkbar, wie beispielsweise ein Bezeichner. Jedoch werden weitere Attribute an dieser Stelle nicht benötigt, weshalb auf eine weitere Detaillierung verzichtet wird.

Ähnlich stellt es sich bei der Entität `SLA` dar: Hier sind zahlreiche Attribute denkbar. Jedoch sind diese stark vom tatsächlichen Kontext abhängig und werden sehr schnell sehr umfangreich (vergleiche exemplarisch [Ber05, S. 99]). Daher erscheint die Vergabe von Attributen bei SLAs hinsichtlich deren heterogener Ausprägung sowie der gewollt geringen Komplexität des Metamodells nicht zielführend.

Tabelle 4.3 ist weiterhin zu entnehmen, dass auf die Festlegung von Attributen für die Entität `SoftwareEnvironment` verzichtet wurde. Diese Entscheidung rührt von der Tatsache her, dass es sich bei dieser Entität lediglich um ein Konzept zur Vereinfachung der Modellierung handelt und Attribute im Kontext der vorliegenden Arbeit folglich nicht benötigt werden.

4.2.2 Hardware

Neben der Klasse *Software* ist natürlich auch die Klasse *Hardware* von großer Bedeutung. Im Rahmen der Analyse in Kapitel 3 konnte gezeigt werden, dass die untersuchten Ansätze hinsichtlich einer eindeutigen Abbildung von Hardware Probleme haben, wenn die Unterscheidung von Hardware anhand der Definition einer IT-Infrastruktur [PHZ09] beziehungsweise anhand der Kriterien aus [Zwa09], die auch die Grundlage für die Untersuchungskriterien in Abschnitt 3.1 bildet, erfolgt. Im Kontext der Erstellung des Metamodells wurde daher insbesondere darauf geachtet, eine Lösung für diese Probleme zu finden.

Entitäten

In der Klasse *Hardware* werden verschiedene Ausprägungen von Hardware innerhalb der Systemlandschaft unterschieden. Jede Entität definiert dabei eine Art von Hardware, die sich hinsichtlich ihres Verwendungszweckes von anderen Entitäten unterscheiden.. Eine Übersicht der im Metamodell berücksichtigten Entitäten sowie eine Beschreibung dieser ist Tabelle 4.4 zu entnehmen. Deren Zusammenspiel ist in Abbildung 4.3 veranschaulicht.

Entität	**Erläuterung**
Hardware	Abstrakte Klasse, welche die generischen Eigenschaften von Hardware kapselt.
HardwareComponent	Beschreibt einen Bestandteil eines Hardwareverbundes.
Location	Ist ein physischer Ort, an dem Hardware lagert.
Server	Explizite Ausprägung der abstrakten Klasse *Hardware*, die Server repräsentiert.
VirtualMachine	Software, die Hardware emuliert und auf einem `Server` ausgeführt wird.
Workstation	Explizite Ausprägung der abstrakten Klasse *Hardware*, die einen Personal Computer repräsentiert.
Device	Explizite Ausprägung der abstrakten Klasse *Hardware*, die ein Gerät innerhalb der Systemlandschaft repräsentiert, das weder `Workstation` noch `Server` ist.
Network	Ein Netzwerk, dass die Entitäten von *Hardware* untereinander verbindet.

Tabelle 4.4: Entitäten der Klasse „Hardware“

Wie bereits der Erläuterung in Tabelle 4.4 entnommen werden kann, handelt es sich bei `Hardware` um eine abstrakte Entität. Sie stellt das Äquivalent zu `Software` dar und dient somit der Kapselung von generischen Relationen und Eigenschaften von Hardware. So beschreibt die Relation `consists_of` zwischen `Hardware` und `HardwareComponent` beschreibt den Aufbau von Hardware aus einzelnen Komponenten. Das Metamodell gibt dabei nicht ab, wie granular die einzelnen Bestandteile von Hardware beschrieben werden. Dies wäre, je nach Intention des Modellierers, durch diesen selber festzulegen. Hardware kann durch die Beziehung `has_site` eine `Location` zugeordnet werden. Dabei handelt es sich um einen physischen Standort im Unternehmen.

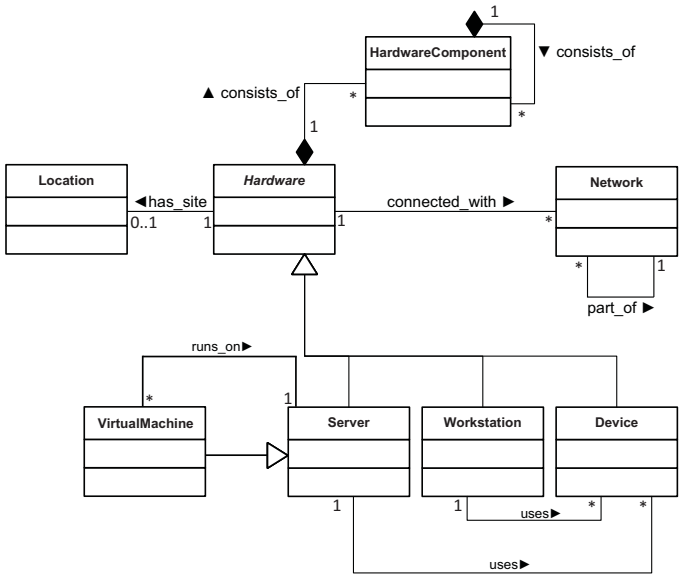

Abbildung 4.3: Entitäten der Klasse „Hardware“ und Relationen

Server, Workstation und Device stellen konkrete Realisierungen von Hardware dar. Ein Server stellt die benötigte Infrastruktur für das Ausführen von Software bereit, auf die dann beispielsweise eine Workstation zugreifen kann. Eine Workstation ist ein Computer, der von einem Mitarbeiter des Unternehmens verwendet wird. Ein Device ist ein beliebiges Gerät, bei dem es jedoch weder um einen Server noch um eine Workstation handelt, beispielsweise ein Drucker. Die Beziehung uses zwischen Server und Device beziehungsweise zwischen Workstation und Device zeigt auf, dass ein Gerät von beiden Entitäten genutzt werden kann.

Eine weitere Vererbungsbeziehung besteht zwischen Server und VirtualMachine. Virtualisierung ist heute ein weit verbreitetes Konzept und bezeichnet Techniken, die eine Abstraktionsschicht zwischen einer Applikation und physischen Ressourcen einziehen [HWBH09]. Eine virtuelle Maschine ist in diesem Kontext als eine Software zu verstehen, welche eine Hardwareumgebung emuliert. Somit müsste sie im Metamodell eigentlich als Software dargestellt werden. Da sie jedoch durch Eigenschaften von Hardware charakterisiert wird, sind virtuelle Maschinen im Metamodell als Hardware modelliert. Die Relation runs_on zwischen VirtualMachine und Server drückt dabei aus, dass eine virtuelle Maschine auf einem Server ausgeführt wird.

Ein weiteres wesentliches Element von *Hardware* sind Netzwerke, die durch die Entität Network repräsentiert werden. Sowohl Server als auch Workstation und Device können mit einem oder mehreren Netzwerken verbunden sein. Die Relation part_of drückt dabei aus, dass Netzwerke in verschiedene Subnetze gegliedert sein können.

Attributierung

In Tabelle 4.5 ist eine Auflistung möglicher Attribute der Entitäten der Klasse *Hardware* zu sehen. Diese sind, wie auch schon bei der Klasse *Software*, dem Common Information Model entnommen. Für die Bestimmung der Attribute wurde dabei zum einen auf die Attribute semantisch identischer Entitäten von CIM, zum anderen auf die Entitäten selber zurückgegriffen. Diese wurden im vorliegenden Metamodell jedoch als Attribute übernommen, um die Komplexität möglichst gering zu halten.

Hardware umfasst generalisierte Eigenschaften, die sich auch in den von Hardware abgeleiteten Entitäten Server, Workstation, Device sowie VirtualMachine finden lassen beziehungsweise finden lassen könnten. Diese müssen zwar, zumindest bei einem Device, nicht zwingend vorhanden sein beziehungsweise angegeben werden. Jedoch erscheint eine Berücksichtigung vor dem Hintergrund der bestehenden Möglichkeit sinnvoll. Daher wurden die Attribute in der abstrakten Entität Hardware berücksichtigt.

Da HardwareComponent eine sehr generische Entität darstellt, werden hier, bis auf die Angabe des Herstellers, keine weiteren Attribute vorgeschlagen, da zu viele Ausprägungen dieser Entität vorstellbar sind und diese in ihren Attributen sehr stark variieren können – und somit die Komplexität des Modells steigern würde, was jedoch vor dem Hintergrund von KMU zu vermeiden ist. Hingegen umfasst die Location alle Attribute einer Location, wie sie CIM beschreibt, da sie eine sinnvolle Anreicherung darstellen und zudem nicht zu umfangreich sind.

Netzwerke werden durch einen Bezeichner (Name) sowie eine Start- (StartAddress) beziehungsweise Endadresse (EndAddress) für die Vergabe von IP-Adressen charakterisiert. Zudem findet sich bei Network das Attribut Subnetmask, welches immer in Kombi-

Entität	Attribut	Beschreibung
Hardware	`Name`	Bezeichner der Hardware.
	`Manufacturer`	Name des Herstellers.
	`CPU-Cores`	Anzahl der Kerne einer CPU.
	`CPU-Speed`	Geschwindigkeit eines Kerns in GHz.
	`Memory`	Verfügbarer physischer Arbeitsspeicher.
	`Storage`	Kapazität der Festplatten in GB.
	`IP`	IP der Hardware.
HardwareComponent	`Name`	Bezeichnung der Komponente.
	`Manufacturer`	
Location	`Name`	Bezeichner des Standortes.
	`PhysicalLocation`	Name des Herstellers.
	`Address`	Seriennummer der Software.
Server	–	–
VirtualMachine	–	–
Workstation	–	–
Device	–	–
Network	`Name`	Eindeutiger Bezeichner des Netzwerks.
	`StartAddress`	IP-Adresse, ab der Adressen vergeben werden.
	`EndAddress`	IP-Adresse, bis zu der Adressen vergeben werden.
	`Subnetmask`	Gibt die Präfixlänge einer IP-Adresse an.

Tabelle 4.5: Mögliche Attributierung der Entitäten der Klasse „Hardware“

nation mit IP-Adressen verwendet wird. Dabei dient die Subnetzmaske dazu, den Präfix für eine IP-Adresse festzulegen.

4.2.3 Cloud Computing

Wie die Analyse in Kapitel 3 ergeben hat, kann keiner der betrachteten Ansätze die Konzepte des Cloud Computings zufriedenstellend abbilden. Eine Modellierung ist zwar indirekt durch die Nutzung anderer Notationselemente der untersuchten Ansätze möglich, was jedoch insgesamt nicht zielführend ist. Da im Kontext dieser Arbeit vor allem die Konzepte des Cloud Computings in der Modellierung von Systemlandschaften berücksichtigt werden sollen, müssen diese explizit eingebunden werden. Nachfolgend wird daher auf die Umsetzung dieser Konzepte im Rahmen des vorliegenden Metamodells eingegangen. Dabei werden zunächst die Entitäten und deren Relationen vorgestellt sowie im Anschluss eine mögliche Attributierung der Entitäten vorgeschlagen.

Entitäten

Die Klasse *Cloud Computing* fasst alle relevanten Entitäten in Bezug auf die Konzepte des Cloud Computings zusammen. Sie verfügt im Vergleich zu den bis hier vorgestellten Klassen *Software* und *Hardware* über weniger Entitäten, was auch die Auflistung der Entitäten in Tabelle 4.6 zeigt.

Entität	Erläuterung
CloudComputing	Abstrakte Klasse, welche die generischen Eigenschaften von Software kapselt.
SaaS	Realisierung von *CloudComputing*, welche die Cloud Computing Ebene „SaaS“ darstellt.
IaaS	Realisierung von *CloudComputing*, welche die Cloud Computing Ebene „IaaS“ darstellt.
PaaS	Realisierung von *CloudComputing*, welche die Cloud Computing Ebene „PaaS“ darstellt.

Tabelle 4.6: Entitäten der Klasse „Cloud Computing“

Zwar ist die Anzahl der Entitäten der Klassen gering, jedoch weisen sie einen hohen Grad an Integration mit den Entitäten anderer Klassen auf. Daher bedarf es auch an dieser Stelle einer Erläuterung der Entitäten von *Cloud Computing* sowie deren Beziehungen zu anderen Elementen des Metamodells. Ein Überblick ist Abbildung 4.4 zu entnehmen.

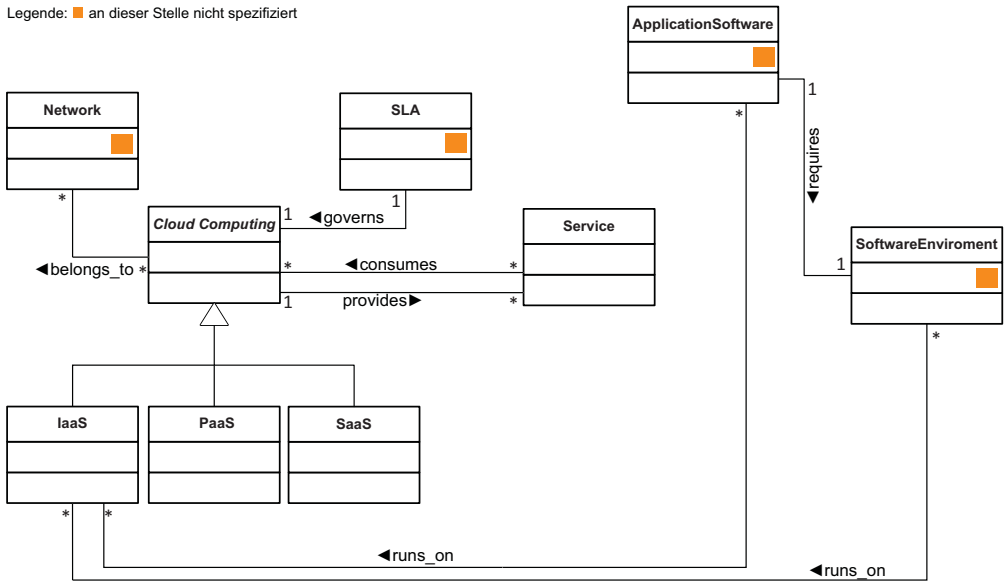

Abbildung 4.4: Entitäten der Klasse „Cloud Computing“ und Relationen

Den Kern der Klasse bildet die abstrakte Entität `CloudComputing`. Sie fasst alle grundlegenden Eigenschaften und Relationen der verschiedenen Ebenen des Cloud Computings zusammen. Über `CloudComputing` werden auch die für die expliziten Realisierungen `SaaS`, `IaaS` und `PaaS` wichtigen Relationen abgebildet. Dies ist zunächst die Zugehörigkeit aller Entitäten der Klasse zu einem Netzwerk, was durch die Beziehung `belongs_to` zwischen `CloudComputing` und `Network` abgebildet wird. Zudem sind im Kontext von Cloud Computing vor allem auch Service Level Agreements von Bedeutung. Diese werden durch die Beziehung `governs` zwischen `CloudComputing` und `SLA` der Entität `Cloud Computing` beziehungsweise den konkreten Realisierungen dieser abstrakten Entität zugeordnet. Die Relationen `consumes` und `provides` zwischen `CloudComputing` und `Service` drücken die Integration von Systemen des Unternehmens und Cloud-Lösun-

gen aus. So kann es beispielsweise einer SaaS-Anwendung ermöglicht werden, über einen Service auf Daten zuzugreifen, die auf einem Server innerhalb des Unternehmens und nicht innerhalb der Cloud liegen. Gleichzeitig wird es internen Anwendungssystemen ermöglicht, auf durch Cloud-Anwendungen offerierte Services zuzugreifen.

Neben diesen grundlegenden, mittels `CloudComputing` abgebildeten Beziehungen, verfügt lediglich `IaaS` über eigene Relationen. IaaS kann genutzt werden, um die Infrastruktur des Unternehmens On-Demand zu erweitern. Da bei IaaS also Bereitstellung von Rechenkapazitäten über Netzwerke erfolgt, muss `IaaS` mit Systemsoftware und Anwendungssoftware integriert werden, um die allozierten Kapazitäten nutzen zu können. Dies geschieht durch die beiden Relationen `runs_on` zwischen `IaaS` und `ApplicationSoftware` beziehungsweise `IaaS` und `SoftwareEnvironment`.

Attributierung

Wie bei den bisher vorgestellten Klassen ist eine Attributierung der Entitäten der Klasse *Cloud Computing* wünschenswert. Anders als bei den bisherigen Klassen sind die Attribute jedoch nicht CIM entnommen, da auch hier die einzelnen Ebenen des Cloud Computings nicht explizit berücksichtigt. Die aufgezeigten Attribute der einzelnen Entitäten in Tabelle 4.7 stellen somit einen Vorschlag des Autors dar.

Entität	**Attribut**	**Beschreibung**
CloudComputing	`Name`	Bezeichner des Dienstes.
	`URL`	URL des Dienstes.
	`IP`	IP des Dienstes.
	`Provider`	Anbieter des Dienstes.
	`DeploymentModel`	Definiert die Bereitstellungsart, z. B. „Public“ (vergleiche Abschnitt 2.4).
SaaS	–	–
IaaS	`Type`	Kategorie des Dienstes.
	`Size`	Unterkategorie von `Type`, welche die Ressourcen genauer spezifiziert (zum Beispiel Größe des Arbeitsspeichers).
	`AutoScaling`	Kapazität der Festplatten in GB.
PaaS	`Features`	Liste der bereitgestellten Funktionalitäten.

Tabelle 4.7: Mögliche Attributierung der Entitäten der Klasse „Cloud Computing“

Die vorgeschlagenen Attribute von `CloudComputing` repräsentieren allgemeine Informationen dar, die für alle drei Ebenen des Cloud Computings relevant sind. Das Angebot an SaaS-Lösungen ist sehr heterogen. Eine allgemeine Attributierung ist daher an dieser Stelle schwierig, weshalb auf die Ausarbeitung weiterer Attribute verzichtet wird. Anders zeigt es sich bei IaaS-Lösungen: Zwar ist auch hier eine Vielzahl von Angeboten verfügbar, diese sind jedoch zumindest dahingehend homogen, dass Rechenkapazitäten über das Internet bezogen werden. Die Rechenkapazitäten werden dabei von der Mehrzahl der Anbieter in Kategorien aufgeteilt. Die für `IaaS` vorgeschlagenen Attribute basieren daher auf der Typisierung verschiedener Kategorien von IaaS-Angeboten, die in

der Form vom Marktführer für IaaS-Dienste Amazon[1] angeboten werden. Um aber die Attributierung nicht auf einen einzelnen Anbieter zuzuschneiden, wurden nicht genau dessen Attribute eingeführt. Stattdessen wurde die Idee von verschiedenen Typen von IaaS-Diensten (durch das Attribut `Type` repräsentiert) aufgegriffen, die sich wiederum in verschiedene Leistungsklassen untergliedern (ausgedrückt durch das Attribut `Size`). Zudem wurde das Attribut `AutoScaling` in die Auflistung aufgenommen. Diese gibt an, ob je nach Grad der Ressourcen-Auslastung automatisch zwischen verschiedenen Ausprägungen von `Size` gewechselt wird.

Die Identifizierung von charakteristischen Attributen gestaltet sich bei `PaaS`, wie es auch schon bei `SaaS` der Fall war, schwierig. Im Kontext von PaaS geht es, wie in Abschnitt 2.4 gesehen, um die Bereitstellung kompletter Infrastrukturen für Entwicklungs- und Betriebsumgebungen. Da diese sehr heterogen ausgeprägt sein können, beschränkt sich die Festlegung von Attributen der Entität `PaaS` auf das Attribut `Features`. Dieses stellt eine Liste der durch die Plattform offerierten Funktionalitäten dar.

4.2.4 Weitere Elemente des Metamodells

Neben den bereits vorgestellten Klassen von Entitäten umfasst das Metamodell noch weitere Objekte, die bislang nicht vorgestellt wurden. Dies betrifft zwei Aspekte:

1. Die Einbeziehung der Dimensionen „Mensch“ und „Aufgabe“. Bislang wurde lediglich die Dimension „Technik“ einer Systemlandschaft betrachtet. Zwar liegt genau hier der Schwerpunkt dieser Arbeit, dennoch ist zumindest eine Berücksichtigung der Aspekte dieser Dimensionen wünschenswert.

2. Die Berücksichtigung der Dynamik einer Systemlandschaft. Wie bereits in Kapitel 2 ausgeführt, zeichnen sich Systemlandschaften insbesondere durch ihren dynamischen Charakter aus. Daneben wurde in Kapitel 3 festgestellt, dass zumindest die untersuchten Ansätze diese nicht berücksichtigen. Zwar gestaltet sich die Berücksichtigung dynamischer Aspekte in einem statischen Modell schwierig, dennoch soll das entwickelte Metamodell einen Ansatz aufzeigen, wie diese Dynamik berücksichtigt werden kann.

Eine Attributierung wird in beiden Fällen jedoch nicht vorgenommen, da der Fokus des Metamodells auf der Abbildung der technischen Dimension einer Systemlandschaft liegt.

Einbeziehung der Dimensionen „Mensch“ und „Aufgabe“

Wie bereits einleitend erwähnt, ist eine Einbeziehung aller drei Dimensionen in das Metamodell wünschenswert. Da der Schwerpunkt dieser Arbeit auf der Dimension „Technik“ liegt, wurden andere Dimensionen nur rudimentär abgebildet. Sie zeigen jedoch eine Möglichkeit zur Berücksichtigung derartiger Konzepte in der Modellierung auf. Abbildung 4.5 zeigt die Entitäten hinsichtlich der Systemlandschafts-Dimension „Aufgabe“ sowie deren Relationen im Metamodell.

[1] Für die genaue Unterscheidung siehe `http://aws.amazon.com/ec2/`

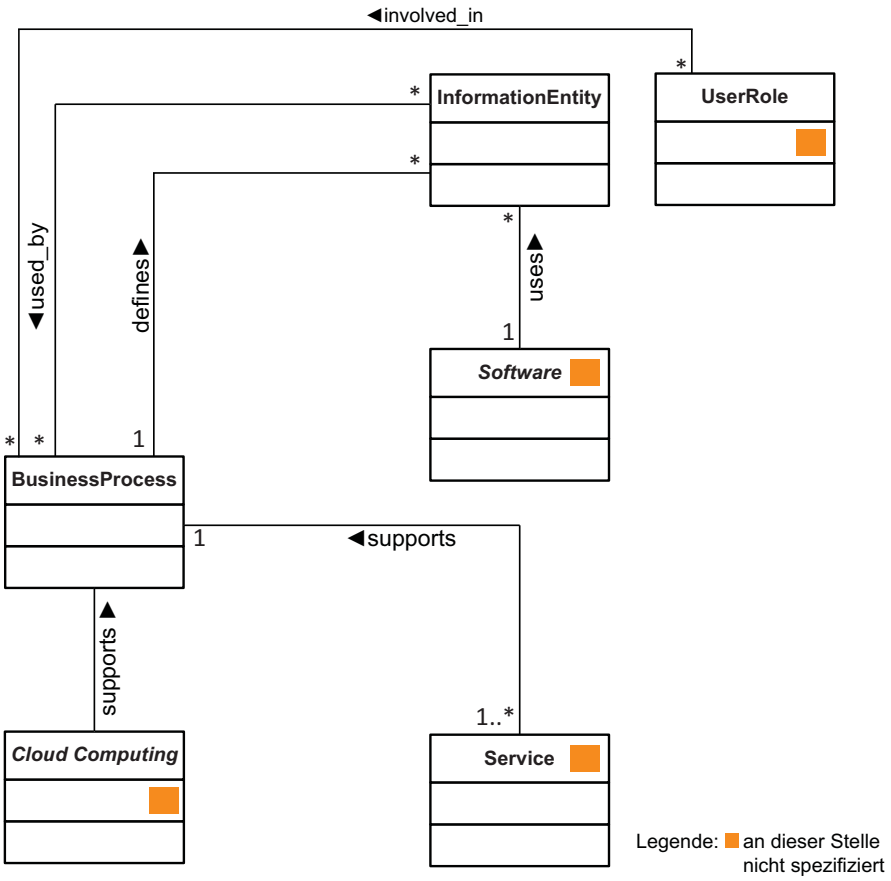

Abbildung 4.5: Entitäten der Dimension „Aufgabe“ und Relationen

Die „Aufgabe“ wird im vorliegenden Metamodell durch die Entität `BusinessProcess` berücksichtigt. Dabei handelt es sich um die Repräsentation eines Geschäftsprozesses. Ein solcher Geschäftsprozess wird durch die abstrakte Entität `CloudComputing` beziehungsweise deren Realisierungen und `Services` unterstützt. `BusinessProcess` dient, wie bereits zuvor erwähnt, jedoch lediglich dazu, die Rolle eines Geschäftsprozesses innerhalb einer Systemlandschaft aufzuzeigen. Im Rahmen eines Geschäftsprozesses sind zudem Menschen involviert, was durch die Beziehung `involved_in` zwischen `BusinessProcess` und `UserRole` ausgedrückt wird. Natürlich ist das Konzept hinter einem Geschäftsprozess weitaus komplexer als es Abbildung 4.5 suggeriert. Auf detaillierte Modellierung der Entität wurde aber im Hinblick auf den Schwerpunkt dieser Arbeit bewusst verzichtet. Eine Detaillierung, wie sich beispielsweise innerhalb der MEMO-OrgML zu finden ist, kann aber durch die Berücksichtigung im Metamodell auch im Rahmen weiterer Arbeiten vorgenommen werden.

Neben dem Geschäftsprozess findet auch die Entität `InformationEntity` Verwendung. Dabei handelt es sich um ein Geschäftsobjekt, also ein Objekt, das für alle Anwendungssysteme einer Systemlandschaft die gleichen Attribute und Operationen aufweist. Die Entität ist zwar nicht direkt der Dimension „Aufgabe“ zuzuordnen, allerdings ist deren Berücksichtigung in Verbindung mit einem Geschäftsprozess nur folgerichtig. Zum einen liegt dies darin begründet, dass Geschäftsobjekte durch Geschäftsprozesse definiert und folglich von Anwendungen gemäß der Definition des Prozesses genutzt werden. Dies impliziert auch den zweiten Punkt, der eine Berücksichtigung von `InformationEntity`

rechtfertigt: In Unternehmen werden einheitliche Standards in Bezug auf Prozesse (und damit auch Anwendungssysteme) immer wichtiger [Mei04]. Daher ist eine Standardisierung der Objekte, mit denen Anwendungssysteme arbeiten, konsequent. Zudem ist ein Einbeziehen von `InformationEntity` sinnvoll, da es bereits im Rahmen der Analyse in Kapitel 3 als besondere Stärke des Modellierungsansatzes von Kirchner ausgemacht wurde.

Neben der Dimension „Technik“ spielt vor allem auch der Faktor Mensch bei Systemlandschaften eine maßgebliche Rolle. Wie bereits in Abschnitt 2.1.3 aufgezeigt wurde, ist der Mensch eine elementare, aber häufig vernachlässigte Komponente von Systemlandschaften. Die Wichtigkeit liegt darin begründet, dass die Existenz von einzelnen Systemen und folglich auch Systemlandschaften *„letztendlich auf die Absichten des Menschen zurückzuführen“* [HHR04, S. 14] ist. Dieser Faktor kann natürlich nur schwierig in ein statisches Modell einbezogen werden. Daneben interagiert der Mensch aber auch mit der Systemlandschaft und ist zudem in Geschäftsprozesse eingebunden. Die Berücksichtigung dieser Funktionen ist in Abbildung 4.6 dargestellt.

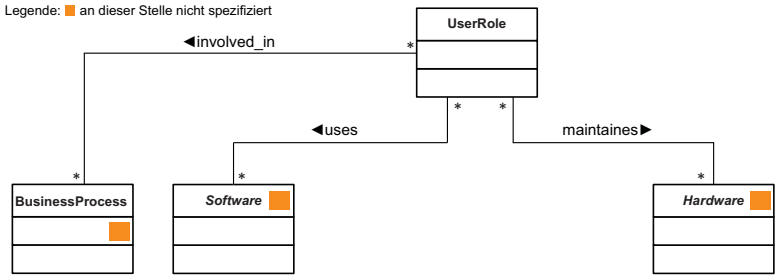

Abbildung 4.6: Entitäten der Dimension „Mensch“ und Relationen

Abbildung 4.6 zeigt, dass die Einbeziehung der Dimension „Mensch“ in das Metamodell durch die Berücksichtigung von Nutzern beziehungsweise Nutzerrollen in Form der Entität `UserRole` realisiert wird. Wie auch bei der Dimension Aufgabe wurde an dieser Stelle aber bewusst auf eine detaillierte Modellierung dieses Konzeptes verzichtet, da der Schwerpunkt dieser Arbeit nicht auf der Betrachtung des Faktors Mensch liegt. Jedoch werden weitere Detaillierungen der Dimension durch die grundlegende Einbeziehung von menschlichen Faktoren ermöglicht.

Einbeziehung der Dynamik

Eines der prägenden Merkmale von Systemlandschaften ist ihre Dynamik (vergleiche Abschnitt 2.1.3). Die Systemlandschaft entsteht nicht plötzlich und verharrt dann in ihrer Struktur, sondern zeichnet sich vor allem dadurch aus, dass ihre gesamte Struktur einem stetigen Wandel unterliegt. Natürlich impliziert das vorliegende Metamodell, wie auch die in Kapitel 3 untersuchten Ansätze, dahingehend ein gewisses Maß an Dynamik, da das Modell jederzeit erweitert werden kann. Allerdings handelt es sich hierbei nicht um eine ausreichende Berücksichtigung dieses besonderen Merkmales einer Systemlandschaft. In Abbildung 4.7 wird daher ein Ansatz gezeigt, der die dynamischen Eigenschaften deutlicher einbezieht.

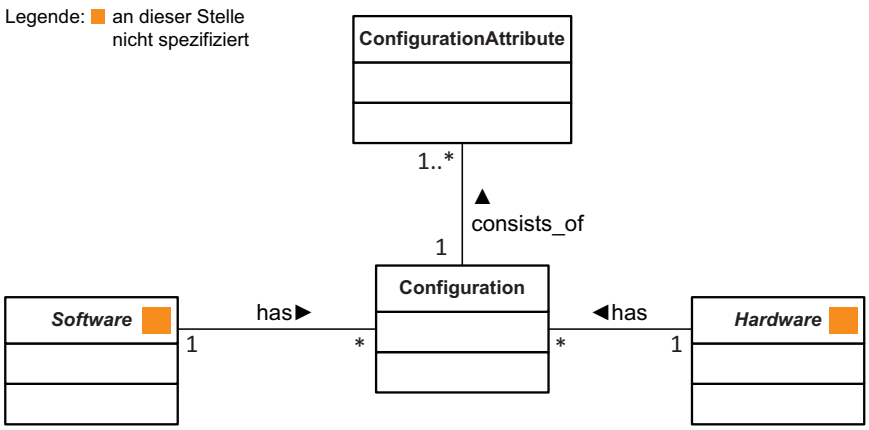

Abbildung 4.7: Entitäten der Dynamik und Relationen

Die Basis für die Abbildung der dynamischen Aspekte in der Form, wie sie im Metamodell umgesetzt wurde, ist das Einbeziehen von Konfigurationen (`Configuration`). Die Entität `Configuration` stellt im Wesentlichen eine Sammlung von `ConfigurationAttributes` dar, welche die Ausprägung der Parameter einer Konfiguration definiert. Eine Konfiguration beschreibt die Parametrisierung von Hardware oder Software, definiert also einen bestimmten Zustand eines Systems. Zwar bildet dieses Konzept nur einen Teil der vielfältigen Änderungen einer Systemlandschaft ab. Die Dokumentation dieser Konfigurationen zeigt aber zumindest einen Teil der Dynamik auf. Im Rahmen des Metamodells wird daher das Attribut `Version` für die Entität `Configuration` aufgezeigt.

4.3 Evaluierung

Nachdem der Aufbau des entwickelten Metamodells erläutert wurde, kann dieses nun evaluiert werden. Wie bereits zuvor dargelegt, wurde anhand der in Abschnitt 4.1 definierten Anforderungen sowie den Ergebnissen der Analyse vorhandener Modellierungsansätze aus Kapitel 3 ein Metamodell aufgestellt, dass zum einen die Schwächen der vorhandenen Ansätze beheben und zum anderen den definierten Anforderungen genügen soll. Die Evaluierung erfolgt dabei anhand verschiedener Schritte:

1. Bewertung des Metamodells anhand der in Abschnitt 4.1 aufgestellten Anforderungen.
2. Modellierung des Beispielszenarios aus Abschnitt 3.2 anhand des entwickelten Metamodells.
3. Analyse des Metamodells anhand der Untersuchungskriterien aus Abschnitt 3.1.

Die Bewertung wird nun im Folgenden auf Basis dieser Punkte vorgenommen.

4.3.1 Analyse der der allgemeinen Anforderungen

Hinsichtlich der allgemeinen Anforderungen wurde zwischen anwenderbezogenen und anwendungsbezogenen Anforderungen unterschieden. Diese werden nachfolgend betrachtet.

Anwenderbezogene Anforderungen

Anwenderbezogene Anforderungen nehmen, insbesondere vor dem Hintergrund kleiner und mittlerer Unternehmen, eine wesentliche Rolle ein. Unterschieden wird dabei zwischen den Kriterien *Einfachheit* sowie *Verständlichkeit und Anschaulichkeit*.

Einfachheit ist dabei ein nur schwer definierbares Kriterium. Frank und van Laak verstehen unter Einfachheit die *„Zahl [der] Konzepte sowie der [...] darstellenden Symbole“* [FvL03, S. 28]. Eine Sprache ist demnach umso einfacher, je geringer die Zahl der Konzepte und darstellenden Symbole ist. Dabei besteht das Problem, dass es sich hier nicht um ein objektives Kriterium handelt, sodass für diesen Fall keine fundierte Aussage möglich ist. Einfachheit lässt sich vielmehr als Vergleichskriterium verstehen. Demnach ist die Frage, wie einfach das entwickelte Metamodell im Vergleich zu anderen Ansätzen ist.

Bilden die in Kapitel 3 untersuchten Sprachen den Bezugspunkt für die Beantwortung der Frage, ob das vorliegende Metamodell einfach gestaltet ist, muss diese verneint werden. Dies liegt darin begründet, dass der Umfang des entwickelten Metamodells größer ist als der Umfang der untersuchten Sprachen. Ist jedoch beispielsweise CIM Ausgangspunkt der Betrachtung, kann festgehalten werden, dass das Metamodell dem Kriterium der Einfachheit genügt – schließlich umfasst CIM mehr als 400 Entitäten.

Eine Bewertung von „Einfachheit“ ist, wie soeben dargelegt, stark von der Betrachtung abhängig. Insgesamt kann also keine verlässliche Aussage hinsichtlich der Einfachheit des Metamodells getroffen werden. Wohl aber lässt sich eine vergleichsweise geringe Komplexität im Hinblick auf die durchschnittliche Anzahl von Elementen feststellen, wenn sowohl einfache als auch komplexe Ansätze als Ausgangspunkt dienen. In diesem Fall könnte die vergleichsweise geringe Komplexität als eine Art von Einfachheit im Sinne von Frank und van Laak interpretiert werden.

Das Kriterium Verständlichkeit/Anschaulichkeit ist zum einen von der Einfachheit abhängig, zum anderen hängt die Verständlichkeit aber auch davon ab, wie stark der vorliegende Ansatz den dem Anwender bereits bekannten Modellierungssprachen ähnelt. Da das vorliegende Metamodell mithilfe von Notationselementen der UML erstellt wurde, kann aufgrund der weiten Verbreitung von UML davon ausgegangen werden, dass das entwickelte Metamodell verständlich ist. Dies wird auch dadurch unterstützt, dass im Metamodell lediglich Entitäten der Domäne berücksichtigt wurden, was einen wichtigen Punkt darstellt [FvL03]. Wird nun davon ausgegangen, dass „Einfachheit“ erfüllt ist, kann dem Metamodell auch Verständlichkeit zugesprochen werden. Eine allgemeingültige Aussage gestaltet sich jedoch auch hinsichtlich dieses Kriteriums schwierig.

Anwendungsbezogene Anforderungen

Eine weitere Kategorie der allgemeinen Anforderungen ist die Gruppe der anwendungsbezogenen Anforderungen. Diese umfasst die Kriterien *Mächtigkeit und Angemessenheit.* sowie *Operationalisierbarkeit.*

Als Mächtigkeit und Angemessenheit definieren Frank und van Laak die Möglichkeit für den Modellierer, alle für ihn relevanten Sachverhalte in der gewünschten Detailtiefe beschreiben zu können [FvL03]. Dies ist jedoch kein ausreichendes Maß, da der gewünschte Detaillierungsgrad eines Modells sehr subjektiv ist. Als Basis für dieses Kriterium sind daher die Gegebenheiten in KMU als Zielgruppe dieser Arbeit sowie die Eigenschaften von Systemlandschaften als abzubildende Domäne zu verstehen: Ein häufiges Problem in KMU ist, dass sie über keine IT-Abteilung verfügen und ihre Expertise im Bereich IT folglich eher gering ausgeprägt ist. Damit das Metamodell also verwendet werden kann, muss die Domäne hinreichend einfach abgebildet werden, was den möglichen Detaillierungsgrad einschränkt. Vor dem Hintergrund des Aufbaus der technischen Dimension von Systemlandschaften (siehe Abschnitt 2.1.3) ist der Detaillierungsgrad jedoch ohnehin eingeschränkt. Die im Metamodell vorhandenen Konzepte, beispielsweise den Aufbau von Hardware aus einzelnen Komponenten betreffend, lassen daneben jedoch einen sehr hohen Detaillierungsgrad (zumindest bei bestimmten Entitäten) zu. Somit ist festzuhalten, dass das Kriterium der Mächtigkeit und Angemessenheit unter den Voraussetzungen dieser Arbeit erfüllt ist, wenngleich sich dieser Punkt *„nicht durch präzise [...] Kriterien beschreiben [lässt]“* [FvL03, S. 33].

Eine weitere anwendungsbezogene Anforderung ist die Operationalisierbarkeit. Darunter ist die Eignung des Modells für Analysen und Transformationen zu verstehen. Diese kann nur bedingt als erfüllt betrachtet werden. Zwar ist durch die Abbildung der Entitäten der technischen Infrastruktur und deren grundlegende Attributierung eine Analyse durchaus denkbar. Jedoch beschränkt sich die Analyse auf diese Dimension, da die anderen Dimensionen zwar berücksichtigt, aber kaum detailliert wurden. Da der Fokus dieser Arbeit auf der Abbildung der Dimension „Technik“ liegt, kann die Operationalisierbarkeit dahingehend als erfüllt angesehen werden – zumindest in dem Sinne, dass der Grundstein dafür gelegt wurde.

4.3.2 Analyse der Anforderungen hinsichtlich Systemlandschaften

Hinsichtlich der Abbildung von Systemlandschaften wurden in Abschnitt 4.1 die Anforderungen *Flächendeckende Abbildung* und *Möglichkeiten zur Integration* unterschieden.

Die Anforderung nach einer flächendeckenden Abbildung der Systemlandschaft umfasst eine Reihe von Kriterien, die durch das Metamodell abgedeckt werden müssen (vergleiche [Kir03]):

1. Bewertung des Bestands an Soft- und Hardware,
2. Orientierung an Geschäftsprozessen,
3. Berücksichtigung der Verbindung von Geschäftsprozessen mit Soft- und Hardware,

4. Berücksichtigung von Schnittstellen und Erweiterungen zu einer Geschäftsprozessmodellierungssprache und

5. Einführung geeigneter Attribute.

Der erste Punkt ist in Verbindung mit der Dimension „Technik“ einer Systemlandschaft zu sehen. Diese Dimension beschreibt, wie in Abschnitt 2.1.3 gezeigt, das Zusammenspiel von Hard- und Software. Demnach muss das Metamodell die Erfassung von Anwendungssoftware in Verbindung mit der IT-Infrastruktur (vergleiche Abschnitt 2.1.3) ermöglichen. Durch die Entitäten `ApplicationSoftware`, `SystemSoftware`, `VirtualMachine` sowie die Elemente der Klasse *Hardware* ist folglich eine „Bewertung des Bestands an Soft- und Hardware“ möglich.

Der zweite Punkt fordert, dass eine Orientierung an Geschäftsprozessen erfolgt. Zwar liegt der Schwerpunkt dieser Arbeit auf der technischen Dimension einer Systemlandschaft, dennoch wurden auch die anderen Dimensionen im Metamodell berücksichtigt. Dabei repräsentiert die Entität `BusinessProcess` einen Geschäftsprozess, die zudem mit den anderen Elementen des Metamodells in Relation gesetzt wurde. Demzufolge ist auch die „Berücksichtigung der Verbindung von Geschäftsprozessen mit Soft- und Hardware“ (dritter Punkt) gegeben und Somit ist diese Anforderung erfüllt. Dies gilt ebenso für den vierten Punkt, der die Berücksichtigung von Schnittstellen zu einer Modellierungssprache für Geschäftsprozesse fordert.

Der fünfte und letzte Punkt fordert die Einführung geeigneter Attribute. Zwar stellt Kirchner in [Kir03] diese Anforderung auf, definiert dabei jedoch nicht, wann genau ein Attribut als geeignet anzusehen ist. Unter Eignung könnte beispielsweise die Fähigkeit der Attribute verstanden werden, eine Entität des Modells zu charakterisieren. Da die Attributierung des entwickelten Metamodells auf Basis des Common Information Model erfolgte und die daraus abgeleiteten Attribute somit auf einer gründlichen Analyse beruhen, kann angenommen werden, dass die im Metamodell berücksichtigten Attribute für eine Beschreibung der Entitäten geeignet sind. Somit wäre diese Anforderung erfüllt.

Damit ist hinsichtlich der Forderung nach einer „flächendeckenden Abbildung“ einer Systemlandschaft festzuhalten, dass diese erfüllt wird.

Daneben wurde in Abschnitt 4.1 der Bedarf nach einer Möglichkeiten zur Abbildung von Integrationskonzepten festgelegt. Im Metamodell werden dafür verschiedene Konzepte bereitgestellt: Zum einen geschieht dies durch die Einbeziehung der Entität `Service` in das Metamodell, zum anderen durch die Relation `integrated_with` zwischen zwei Ausprägungen von `ApplicationSoftware`. Die Forderung nach einer Abbildung von Integrationskonzepten ist somit erfüllt.

Zusammenfassend ist demnach festzuhalten, dass alle Anforderungen hinsichtlich der Abbildung einer Systemlandschaft durch das Metamodell abgedeckt werden.

4.3.3 Analyse der Anforderungen hinsichtlich kleiner und mittlerer Unternehmen

Neben den allgemeinen Anforderungen und den Anforderungen bezüglich Systemlandschaften wurden in Abschnitt 4.1 zudem Anforderungen bezüglich KMU aufgestellt. Diese betreffen zum einen die *Einfachheit*, zum anderen die Abbildung von Konzepten des *Cloud Computings*.

Die Forderung nach „Einfachheit des“ Metamodells ergibt sich aus den Besonderheiten in KMU hinsichtlich der dort vorherrschenden IT-Kompetenz. Diese „Einfachheit“, wie sie die Menge der Anforderungen hinsichtlich KMU fordert, ist mit der Einfachheit im Sinne der allgemeinen Anforderungen gleichzusetzen. Im Rahmen der Analyse von „Einfachheit“ (vergleiche Abschnitt 4.3.1) wurde festgestellt, dass es sich dabei nicht um ein objektives Kriterium handelt. Vielmehr stellt „Einfachheit“ ein Vergleichskriterium dar, sodass eine Aussage bezüglich dieses Kriteriums nur im Vergleich mit anderen Ansätzen vorgenommen werden kann und folglich keine allgemeingültige Aussage möglich ist. In Bezug auf die in Kapitel 3 betrachteten Ansätze und unter Berücksichtigung von komplexeren Modellen wie CIM konnte jedoch festgestellt werden, dass das entwickelte Metamodell vergleichsweise einfach ist.

Die Forderung nach der Berücksichtigung von Cloud Computing ergibt sich aus dem Kontext der Arbeit und sieht die Berücksichtigung der Konzepte des Cloud Computings vor. Bei diesen Konzepten handelt es sich um die Ebenen SaaS, PaaS und IaaS (vergleiche hierfür Abschnitt 2.4). Diese Ebenen finden ich im Metamodell in Form einzelner Entitäten wieder: `SaaS`, `PaaS` und `IaaS`. Damit ist die Forderung nach der Berücksichtigung von Cloud Computing durch das Metamodell abgedeckt.

Somit sind alle Anforderungen an einen Modellierungsansatz für Systemlandschaften hinsichtlich der Anforderungen von KMU durch das entwickelte Metamodell abgedeckt.

4.3.4 Beispielszenario

Nachdem eine Analyse auf Basis der zu Beginn des Kapitels definierten Anforderungen erfolgt ist, wird nun eine Analyse im Sinne der Untersuchungen, wie sie in Abschnitt 3.3 vorgenommen wurden, durchgeführt. Bevor jedoch die Bewertung des entwickelten Metamodells vorgenommen werden kann, wird das Beispielszenario aus Abschnitt 3.2 umgesetzt. Auf Basis der Struktur des Metamodells sowie des Ergebnisses des Modellierungsprozesses kann eine Bewertung vorgenommen werden. Abbildung 4.8 zeigt das Ergebnis der Modellierung des Beispielszenarios anhand des entwickelten Metamodells.

Für eine geeignete Repräsentation der Entitäten des Metamodells wurde eine Darstellung auf Basis der UML für die Umsetzung gewählt. Wie Abbildung 4.8 zeigt, wird dabei jede Entität durch ein eigenes Notationselement dargestellt. Jedes Notationselement wird dabei zusätzlich durch seinen jeweiligen Entitäts-Typ (in Form von «`Typ`») gekennzeichnet, sodass die Umsetzung des Beispielszenarios selbsterklärend ist. Lediglich die Repräsentation der Entität `Network` ist nicht unmittelbar offensichtlich: Die Zugehörigkeit von Entitäten zu einem Netzwerk wird durch eine graue Hinterlegung symbolisiert. Zusätzlich ist das Netzwerk mit einem Bezeichner versehen, wie Abbildung 4.8 zeigt.

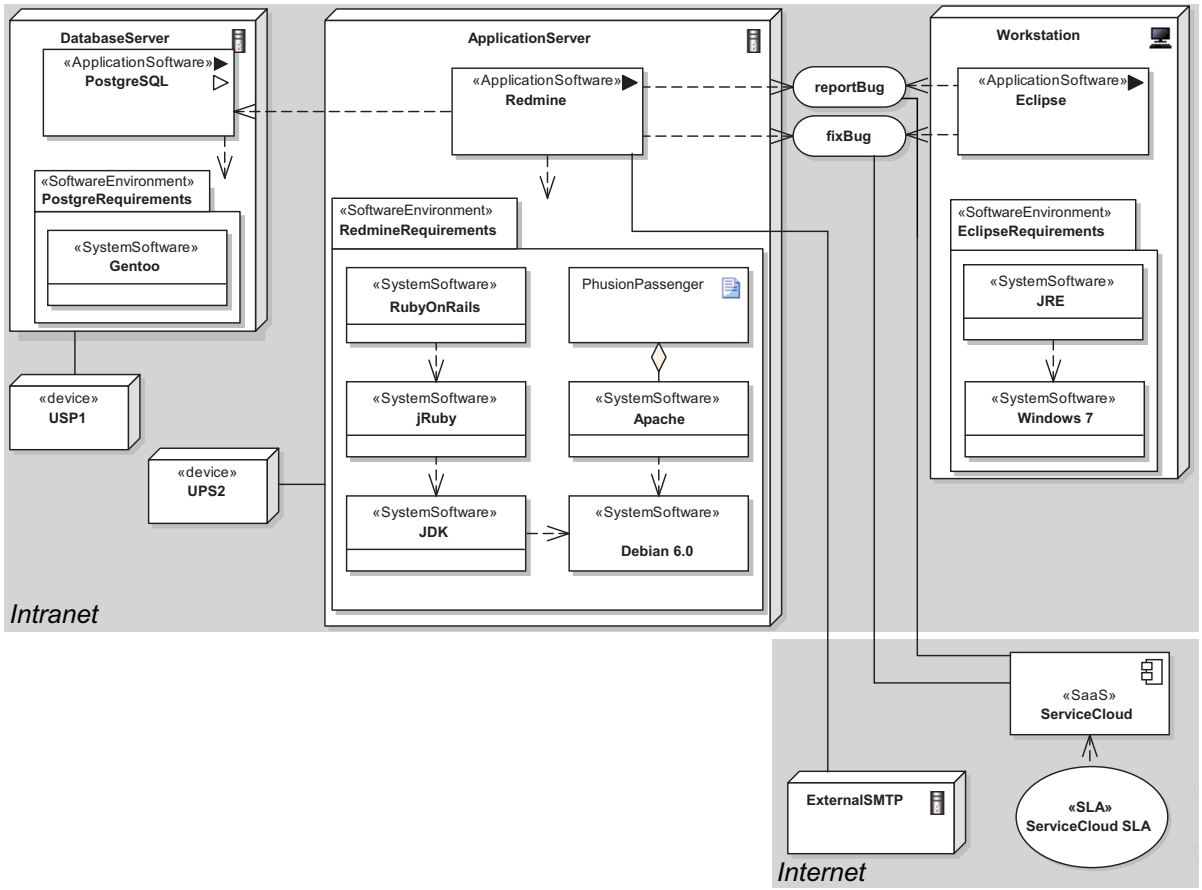

Abbildung 4.8: Beispielszenario: Entwickeltes Metamodell

4.3.5 Analyse anhand der Untersuchungskriterien

Auf Basis des Aufbaus des Metamodells sowie der im vorherigen Abschnitt vorgenommenen Modellierung des Beispielszenarios kann nun eine Analyse durchgeführt werden.

Eine Differenzierung der verschiedenen Arten von Hardware ist, ebenso wie bei Software, möglich. So ist der Typ der Hardware beziehungsweise der Software eindeutig bestimmbar. Mangelnde Eindeutigkeit kann dem Metamodell lediglich in der Hinsicht unterstellt werden, dass es keine Unterscheidungen dahingehend ermöglicht, wann genau es sich bei Software um Anwendungssoftware oder Systemsoftware handelt. Darin besteht jedoch auch nicht die Aufgabe des Metamodells, weshalb dieser Punkt zu vernachlässigen ist.

Im Gegensatz zu den in Abschnitt 3.3 analysierten Modellierungsansätzen konnten mithilfe des entwickelten Metamodells alle Konzepte des Cloud Computings abgebildet werden. Gleiches gilt für die Berücksichtigung von Services und Service Level Agreements. Ebenso ermöglicht das Metamodell die Einbeziehung von Konfigurationen in die Systemlandschaft, was im Rahmen des Beispielszenarios jedoch aus Komplexitätsgründen nicht enthalten ist. Dies ermöglicht zudem nicht nur die Modellierung einer statischen Systemlandschaft, sondern führt dazu, dass auch deren dynamische Charakteristika berücksichtigt werden.

Tabelle 4.8 zeigt eine Zusammenfassung der Ergebnisse der vorausgegangenen Analyse.

Komponenten	**Kriterium**	**Analyse**
	Anwendungssoftware	✓
	Hardware	✓
	Systemsoftware	✓
	Geräte für Betrieb	✓
	Cloud Computing	✓
	SaaS	✓
	IaaS	✓
	PaaS	✓
	Services	✓
	SLAs	✓
Sprache / Werkzeug	**Kriterium**	**Analyse**
	Anzahl Elemente	23
	Eindeutigkeit	(✓)
	Konfiguration	✓
	Struktur/Verhalten	✓/ ✓

Tabelle 4.8: Ergebnis der Analyse des entwickelten Metamodells

Die Stärken des entwickelten Metamodells sind bereits durch Tabelle 4.8 ersichtlich: Im Gegensatz zu den anderen analysierten Ansätzen (vergleiche Abschnitt 3.3.6) vermag das erstellte Metamodell, die Aspekte der technischen Dimension einer Systemlandschaft eindeutig abzubilden. Lediglich die Tatsache, dass das Metamodell keine eindeutige Trennung von Anwendungs- und Systemsoftware ermöglicht, kann hinsichtlich der Eindeutigkeit negative Folgen haben. Das Untersuchungskriterium „Eindeutigkeit“ ist daher in Tabelle 4.8 als „bedingt erreicht“ gekennzeichnet, da die Unterscheidung zwischen Anwendungssoftware und Systemsoftware dem Modellierer obliegt.

Der hier präsentierte Modellierungsansatz ist jedoch, wie auch bereits bei einer genauen Betrachtung des Beispielszenarios deutlich wird, nicht frei von Schwächen: Eine Firewall, wie sie die Spezifikation des Beispielszenarios fordert, ist nicht Bestandteil der Umsetzung. Dies liegt darin begründet, dass im Rahmen der Erstellung des Metamodells eine Kommunikation von Software oder Hardware, die zu jeweils verschiedenen Netzwerken gehören, nicht über ein `Device` möglich ist. Diese Entscheidung wurde jedoch bewusst getroffen, um die Übersichtlichkeit des Modells zu gewährleisten. So wird die eindeutige Zuordnung zwischen zwei Kommunikationspartnern garantiert, was die in Abschnitt 3.3 vorgestellten Ansätze nicht immer vermochten (vergleiche hierfür Abbildung 4.8 und Abbildung 3.6). Dennoch ist dies als Schwachpunkt im Metamodell auszumachen. Zudem kann die Art der Berücksichtigung von virtuellen Maschinen im Metamodell als Schwachpunkt angesehen werden: bei diesen handelt es sich um Software, die Hardware emulieren. Im Metamodell werden sie jedoch aufgrund ihrer Eigenschaften als Hardware aufgeführt. Hinsichtlich ihrer Funktion innerhalb der Systemlandschaft ist diese Abbildung zwar korrekt, jedoch müssten sie mit Blick auf die formale Korrektheit des Metamodells als Software einbezogen werden. Weiterhin kann als Schwachpunkt aufgeführt werden, dass auf einem Device jede Art von Software laufen kann. Dies kann zwar der Fall sein, beispielsweise wenn ein Computer die Funktion einer Firewall übernimmt,

jedoch ist es nicht immer zutreffend. Um das Modell jedoch einfach zu halten, wurde diese Schwachstelle in Kauf genommen.

4.3.6 Fazit

Auf Basis der vorausgegangenen Analyse kann nun ein abschließendes Fazit hinsichtlich der Evaluierung des Metamodells getroffen werden. Die Analyse basiert zum einen auf der Untersuchung des Modells hinsichtlich der in Abschnitt 4.1 definierten Anforderungen, zum anderen auf der Analyse des Metamodells unter der Zuhilfenahme eines Beispielszenarios.

In Bezug auf die Anforderungen an ein Metamodell für die Abbildung von Systemlandschaften kann festgehalten werden, dass diese weitestgehend erfüllt sind. Zwar konnten teilweise keine abschließenden Aussagen zur Erfüllung der Anforderungen durch das Metamodell erfolgen. Dies lag jedoch zumeist darin begründet, wie es beispielsweise bei „Einfachheit“ oder „Verständlichkeit“ der Fall war, dass die Kriterien sehr weich definiert oder generell kaum messbar sind. Durch Einschränkungen der Anforderungen hinsichtlich des Rahmens dieser Arbeit sind die Kriterien dennoch als erfüllt anzusehen.

In Bezug auf die Analyse des Metamodells, wie sie auch im Rahmen von Kapitel 3 bei der Untersuchung existierender Modellierungsansätze erfolgte, ist festzustellen, dass es den definierten Kriterien genügt. Zwar können einige Punkte, wie bereits in Abschnitt 4.3.5 dargelegt, als für die Eignung des Modells kritisch angesehen werden. Jedoch wurden die Entscheidungen, welche die Schwächen des Modells bedingen, bewusst getroffen – vor allem, um die Komplexität des Metamodells möglichst gering zu halten und es somit für KMU nutzbar zu machen.

Abschließend wird festgestellt, dass das Metamodell die Evaluation durchlaufen hat und somit im Rahmen dieser Arbeit erfolgreich umgesetzt wurde.

4.4 Zusammenfassung

Ziel dieses Kapitels war die Vorstellung des entwickelten Metamodells. In Abschnitt 4.1 wurden dafür zunächst Anforderungen an ein Metamodell für Systemlandschaften in KMU vor dem Hintergrund des Cloud Computings definiert. Diese gliedern sich in allgemeine Anforderungen, die an jedes Modell gestellt werden, in Anforderungen an Systemlandschaften und schließlich in Anforderungen an KMU. Im Anschluss daran wurde in Abschnitt 4.2 das Metamodell eingeführt. Dieses wurde sukzessiv in Form einzelner semantischer Gruppen von Entitäten (als „Klassen“ bezeichnet) vorgestellt. Zunächst erfolgte die Beschreibung der jeweiligen Entitäten einer Klasse, bevor deren Relationen untereinander erläutert wurden. Zudem wurde für die Entitäten eine mögliche Attributierung vorgeschlagen. Nach der Vorstellung des Metamodells musste dieses evaluiert werden. Daher wurde in Abschnitt 4.3 eine Analyse auf Basis von Anforderungen an das Metamodell und durch eine Untersuchung anhand der Struktur des Metamodells sowie der Umsetzung eines Beispielszenarios vorgenommen. Die Evaluierung kam dabei zu dem Schluss, dass die Umsetzung des Metamodells erfolgreich war.

Kapitel 5

Fazit und Ausblick

In diesem Kapitel werden die wichtigsten Punkte der Arbeit zusammengefasst und ihre Ergebnisse bewertet. Zudem wird ein Ausblick auf Möglichkeiten zur Fortführung dieser Arbeit gegeben. Zum einen bezieht sich der Ausblick auf Aspekte, die in dieser Arbeit aus unterschiedlichen Gründen nicht oder nur kaum betrachtet wurden. Zum anderen werden Ansätze für weitere Forschungsfragen vorgestellt.

5.1 Zusammenfassung

Aufgrund der in KMU vorherrschenden Rahmenbedingungen, beispielsweise hinsichtlich des verfügbaren IT-Budgets oder der vorhandenen IT-Expertise, werden häufig Potentiale in Bezug auf die vorhandene Informationstechnologie verschwendet oder bleiben ungenutzt. Mit Blick auf den globalen Wettbewerb bietet ein zielgerichteter Einsatz der IT für KMU jedoch die Möglichkeit, sich am Markt zu behaupten. Dieser Trend hält bereits seit einigen Jahren an und wird sich in den kommenden Jahren voraussichtlich noch verstärken.

Damit in kleinen und mittleren Unternehmen, insbesondere vor dem Hintergrund der häufig fehlenden IT-Kenntnisse, überhaupt ein zielgerichteter IT-Einsatz erfolgen kann, müssen Möglichkeiten geschaffen werden, die dieses Vorhaben unterstützen. Diese Möglichkeiten sollten dabei insbesondere aktuelle Trends der IT, wie etwa das Cloud Computing, berücksichtigen, da hier vor allem für KMU große Potentiale liegen. Ein großes Problem hinsichtlich des Einsatz von IT in Unternehmen und insbesondere auch in KMU ist, dass diese kaum Kenntnisse über die von ihnen verwendeten Systeme haben. Dieses Problem überträgt sich dabei auch auf die Systemlandschaft eines Unternehmens. Ein wesentlicher Schritt zur Vermeidung dieses Effektes ist die gezielte Planung und Gestaltung der (System-)Landschaft und daraus folgend die Unterstützung bei der Umsetzung der Unternehmensziele zu gewährleisten. Dafür ist es unabdingbar, den Ist-Zustand einer Systemlandschaft zu erfassen.

Das Ziel dieser Arbeit war daher die Entwicklung eines Metamodells zur Abbildung von Systemlandschaften in kleinen und mittleren Unternehmen. Um dieses Metamodell entwickeln zu können, war es jedoch zuvor notwendig, einige grundlegende Definitionen zu erarbeiten. Insbesondere dem Terminus „Systemlandschaft“ mangelte es bislang an einer klaren Definition. Zwar wird im Kontext der wissenschaftlichen Literatur häufig auf Eigenschaften von Systemlandschaften eingegangen, jedoch blieb die Frage, was genau

unter einer Systemlandschaft zu verstehen ist, unbeantwortet. Um ein Metamodell für die Abbildung von Systemlandschaften entwickeln zu können, bedurfte es jedoch einer genauen Definition des Begriffes. Anhand einer interdisziplinären Betrachtung auf Basis der Begriffe „System“ und „Landschaft“ sowie des Kompositums „Systemlandschaft“ und dessen Einordnung in einen informationstechnischen Kontext konnte im Rahmen der Erarbeitung der Grundlagen eine Arbeitsdefinition aufgestellt werden, die als Verständnis- und Bewertungsgrundlage diente. Neben dem Begriff der Systemlandschaft wurden im Rahmen von Kapitel 2 weitere, für das Verständnis der vorliegenden Arbeit wesentliche Begriffe definiert. Zu diesen zählten „Modell“, „KMU“ und „Cloud Computing“.

Zur Schaffung einer geeigneten Basis für das Metamodell war es notwendig, neben den wesentlichen Begriffsdefinitionen die Domäne der Systemlandschafts-Modellierung zu analysieren. Zu diesem Zweck wurden Sprachen beziehungsweise Werkzeuge untersucht, die für die Modellierung von Systemlandschaften geeignet sein können. Bei der Auswahl dieser Sprachen und Werkzeuge wurde darauf geachtet, dass diese sowohl hinsichtlich ihres Umfangs als auch bezüglich der Höhe der Investitionskosten für KMU geeignet sind. Die Grundlagen für die Analyse der einzelnen Ansätze bildeten zum einen die Betrachtung des jeweiligen Aufbaus, zum anderen die Umsetzung eines zuvor definierten Beispielszenarios mithilfe des jeweiligen Ansatzes. Im Anschluss daran wurde die Analyse anhand eines Kriterienkatalogs vorgenommen. Diese lieferte die Erkenntnis, dass jeder der untersuchten Ansätze Schwächen aufweist. Dies galt insbesondere für die Einbeziehung der verschiedenen Konzepte des Cloud Computings, vor allem aber auch bezüglich einer hinreichend genauen Abbildung der Domäne. Daneben konnten jedoch auch einige Stärken ausgemacht werden, die folglich im anschließend entwickelten Metamodell berücksichtigt werden konnten.

Das zentrale Element dieser Arbeit, das Metamodell, wurde in Kapitel 4 vorgestellt. Als Basis für die Erstellung des Modells dienten, neben der Definition einer Systemlandschaft, zum einen die Erkenntnisse, die im Rahmen der Analyse existierender Ansätze in Kapitel 3 gewonnen wurden. Zum anderen wurde ein Katalog vorgestellt, der Anforderungen an das Metamodell hinsichtlich der Kategorien *allgemeine Anforderungen*, *Anforderungen hinsichtlich Systemlandschaften* sowie *Anforderungen hinsichtlich KMU* umfasst. Das Metamodell wurde dann im weiteren Verlauf schrittweise vorgestellt und anschließend einer Evaluierung unterzogen. Diese erfolgte zunächst auf Basis des zu Beginn von Kapitel 4 vorgestellten Anforderungskatalogs. Weiterhin wurde für die Evaluierung das Beispielszenario herangezogen, welches bereits für die Analyse in Kapitel 3 genutzt wurde. Die so gewonnenen Kenntnisse konnten anschließend für eine Beurteilung des entwickelten Ansatzes dienen. Dabei wurde festgestellt, dass das Metamodell die Evaluation erfolgreich durchlaufen hat, weshalb ein positives Fazit hinsichtlich der Eignung des entwickelten Modells gezogen wurde.

Bewertung der wissenschaftlichen Zielsetzung

Zu Beginn dieser Arbeit wurde eine wissenschaftliche Zielstellung formuliert, die verschiedene Punkte umfasst. Um eine abschließende Beurteilung der vorliegenden Arbeit sowie des in ihrem Kontext entwickelten Metamodells vornehmen zu können, ist es notwendig, die gesteckten Ziele hinsichtlich ihrer Erfüllung zu analysieren.

Als essentiell für die Entwicklung des Metamodells wurde die Definition des Begriffs

„Systemlandschaft“ gesehen. Dieser Zielstellung wurde mit der Erarbeitung einer Definition in Abschnitt 2.1.3 nachgekommen. Das Ziel wurde somit erreicht.

Weiterhin wurde als Ziel festgelegt, dass der Fokus des Metamodells die Dimension „Technik“ sein sollte. Dieser Anforderung konnte ebenfalls nachgekommen werden. Zudem fanden auch die Dimensionen „Aufgabe“ und „Mensch“ Berücksichtigung, was ebenfalls durch die Zielstellung gefordert wurde. Zwar wurden diese nicht detailliert in das entwickelte Metamodell integriert, es konnte aber zumindest eine mögliche Einbeziehung aufgezeigt werden.

Da das Metamodell auf die Abbildung von Systemlandschaften ausgerichtet ist, wurde im Rahmen der wissenschaftlichen Zielstellung die Forderung aufgestellt, dass das Modell alle relevanten Bestandteile einer Systemlandschaft einbeziehen soll. Eine Bewertung dieser Forderung kann an dieser Stelle auf Basis der in Abschnitt 2.1.3 hergeleiteten Definition von „Systemlandschaft“ vorgenommen werden. Dient diese Definition als Grundlage für die relevanten Bestandteile einer Systemlandschaft, ist dieses Ziel erreicht worden.

Zudem wurde gefordert, dass das Metamodell einfach zu halten ist. Eine Bewertung dieses Punktes gestaltet sich schwierig, da die Einfachheit eines Modells stark vom Blickwinkel des Betrachters abhängig ist. Im Rahmen der Analyse des Metamodells in Abschnitt 4.3 wurde das Kriterium „Einfachheit“ bereits eingehend untersucht. Dabei wurde festgestellt, dass eine verlässliche Aussage hinsichtlich dessen Einfachheit nicht möglich ist. Wird das Metamodell beispielsweise aus Sicht einer der in Kapitel 3 analysierten Ansätze betrachtet und Einfachheit gleichzeitig als *„Zahl [der] Konzepte sowie der [...] darstellenden Symbole“* [FvL03, S. 28] definiert, kann festgestellt werden, dass entwickelte Modell nur vergleichsweise einfach ist. Dies liegt darin begründet, dass der Umfang des entwickelten Metamodells den Umfang der untersuchten Ansätze übersteigt. Hingegen kann von Einfachheit ausgegangen werden, wenn beispielsweise das *Common Information Model* den Ausgangspunkt der Betrachtung bildet: Im Vergleich zu diesem ist das Metamodell deutlich weniger umfangreich und folglich ist Einfachheit gegeben. Aus Sicht beider Fälle, also sowohl sehr komplexer als auch sehr einfacher Ansätze, kann dem Metamodell jedoch ein gewisses Maß an Einfachheit zugesprochen werden.

Weiterhin wurde das Ziel formuliert, dass das Metamodell fachliche Funktionalitäten einer Systemlandschaft (im Sinne von *Services*) abbilden muss. Dieses Ziel steht in engem Zusammenhang mit dem letzten Ziel: Das Metamodell soll die Konzepte des Cloud Computings abbilden können. Dabei wurden beide Ziele erfüllt.

Abschließendes Fazit

Eine Beurteilung des Metamodells wurde bereits im Rahmen der Evaluierung im Abschnitt 4.3 vorgenommen. Auf Basis der dort durchgeführten Untersuchungen anhand der Anforderungen an das Metamodell sowie der Analyse des Beispielszenarios wurde festgehalten, dass das Metamodell die Evaluation erfolgreich durchlaufen hat. Folglich ist es geeignet, Systemlandschaften, wie sie in Kapitel 2 definiert wurden, erfolgreich abzubilden.

Hinsichtlich einer abschließenden Beurteilung der gesamten Arbeit stellt die erfolgreiche Umsetzung des Metamodells jedoch nur einen Aspekt dar. Zusätzlich dazu ist das Ergebnis der Beurteilung der wissenschaftlichen Zielsetzung heranzuziehen. Diese

ist, wie im vorherigen Abschnitt gesehen, als erreicht anzusehen, da die gesteckten Ziele erfüllt werden konnten. Lediglich hinsichtlich der Einfachheit des Metamodells konnte keine allgemeingültige Aussage getroffen werden. Dies liegt jedoch in der Natur dieses Kriteriums begründet, da es nur schwer definierbar ist.

Abschließend kann festgehalten werden, dass das entwickelte Metamodell den im Rahmen der Arbeit definierten Anforderungen und Kriterien genügt und daher erfolgreich umgesetzt wurde.

5.2 Ausblick

Die vorliegende Arbeit stell lediglich einen ersten Schritt bezüglich der Modellierung von Systemlandschaften in KMU dar. Wie bereits der wissenschaftlichen Zielstellung zu Beginn dieser Arbeit zu entnehmen ist, liegt der Fokus des entwickelten Metamodells auf den technischen Gegebenheiten einer Systemlandschaft. Wie jedoch die Definition von Systemlandschaft zeigt, wird diese in den Dimensionen *Mensch*, *Aufgabe* und *Technik* beschrieben. Zwar wurde versucht, die Dimensionen Mensch und Aufgabe in das Metamodell einzubeziehen. Um jedoch eine vollständige Abbildung von Systemlandschaften gemäß deren Definition zu gewährleisten, sind weitere Arbeiten am Modell notwendig. Möglichkeiten zur Erweiterung stellen beispielsweise eine Detaillierung der Geschäftsprozesse, wie in der MEMO-OrgML, oder die Berücksichtigung von Konzepten wie Lebenslagen (siehe [Krü07]) dar. Weiterer Forschungsbedarf besteht vor allem hinsichtlich der Attributierung der einzelnen Entitäten des Metamodells. Zwar wurde im Rahmen dieser Arbeit Vorschläge für eine mögliche Attributierung gegeben, jedoch können diese keineswegs als vollständig bezeichnet werden. Insbesondere hinsichtlich der Tatsache, dass das Metamodell trotz einer weiterführenden Attributierung der Entitäten nicht zu komplex werden sollte, besteht noch Forschungsbedarf.

Eine weiterführende Attributierung erscheint auch hinsichtlich der Operationalisierung des Metamodells sinnvoll. Wie bereits einleitend in Kapitel 1 erwähnt, ist eine Abbildung von Systemlandschaften nicht nur im Hinblick auf deren Erfassung und Planung interessant: Vor allem ist auch deren Steuerung von Bedeutung. Dafür ist es wünschenswert, das entwickelte Metamodell mit Konzepten wie etwa in [GHS10] zu verknüpfen und so beispielsweise ein automatisches Konfigurationsmanagement in Verbindung mit einem automatischen Software Deployment umzusetzen. Insbesondere vor dem Hintergrund mangelnder IT-Kenntnisse in KMU erscheint eine Verknüpfung des Metamodells mit derartigen Ansätzen sinnvoll. Erste Vorbereitungen hierfür wurden bereits durch die Berücksichtigung von Entitäten, welche die Konfiguration von Hard- und Software beschreiben, getroffen.

Erweiterungsmöglichkeiten bestehen auch hinsichtlich der Einbringung von Standardisierungsmaßnahmen in das Metamodell, da KMU auch hier mit immer weiter steigenden Anforderungen konfrontiert sind [Rud09]. Erste Ansätze wurden diesbezüglich zwar durch die Berücksichtigung von Geschäftsobjekten im Metamodell eingebracht, allerdings stellt dies nur einen ersten Schritt dar. Zudem gilt es, die in Abschnitt 4.3.6 bereits aufgezeigten Schwachstellen zu beheben.

Literaturverzeichnis

[AS08] Jan Stefan Addicks and Ulrike Steffens. Supporting Landscape Dependent Evaluation of Enterprise Applications. *Multikonferenz Wirtschaftsinformatik (MKWI)*, pages 1815–1825, 2008.

[BB11] Martin Bichler and Kamal Bhattacharya. IT Service Management and IT Automation - Methods and Models for Efficient IT Operations. *Business & Information Systems Engineering*, 3(1):1–2, 2011.

[Ben03] Oliver Bender. Kulturlandschaft und Ländlicher Raum: Struktur und Dynamik der Kulturlandschaft. Diskusion (neuer) Methoden und Anwendungen einer diachronischen Landschaftsanalyse. *Mitteilungen der Österreichischen Geographischen Gesellschaft*, 145:119–146, 2003.

[Ber05] Thomas Götz Berger. *Konzeption und Management von Service-Level-Agreements für IT-Dienstleistungen*. PhD thesis, TU Darmstadt, June 2005.

[BHB10] Alexander Benlian, Thomas Hess, and Peter Buxmann. *Software-as-a-Service : Anbieterstrategien, Kundenbedürfnisse und Wertschöpfungsstrukturen*. Gabler, April 2010.

[Bis09] Thomas Biskup. *Agile fachmodellgetriebene Softwareentwicklung für mittelständische IT-Projekte*. PhD thesis, Fakultät II - Informatik, Wirtschafts- und Rechtswissenschaften, Department für Wirtschaftsinformatik, 2009.

[BKNT09] Christian Baun, Marcel Kunze, Jens Nimis, and Stefan Tai. *Cloud Computing: Web-basierte dynamische IT-Services*. Informatik Im Fokus. Springer, 2009.

[BLRK09] Markus Böhm, Stefanie Leimeistet, Christoph Riedl, and Helmut Krcmar. Cloud Computing: Outsourcing 2.0 oder ein neues Geschäftsmodell zur Bereitstellung von IT-Ressourcen? *Information Management & Consulting*, 24:6–14, 2009.

[Bos92] Hartmut Bossel. Modellbildung und Simulation: Konzepte, Verfahren und Modelle zum Verhalten dynamischer Systeme. *Vieweg Verlag*, 1, 1992.

[Buc05] Annemarie Bucher. Landschaften in Bewegung. In Sandra Kalcher, Thies Schröder, Irene Burkhardt, Axel Lohrer, Andreas Müller, Jutta Sankowski,

Teja Trüper, and Christian Welzbacher, editors, *Spielräume*, pages 132–148. Birkhäuser Basel, 2005.

[Die06] Norbert Diernhofer. *IT-Dienstmodellierung - Modellierung von Beziehungen und Abhängigkeiten zwischen Geschäftsprozessmodellen, Softwarekomponenten und der IT-Infrastruktur*. PhD thesis, Technische Universität München, Fakultät für Informatik, 2006.

[FHK+09] Ulrich Frank, David Heise, Heiko Kattenstroth, Donald Ferguson, Ethan Hadar, and Marvin Waschke. ITML: A Domain-Specific Modeling Language for Supporting Business Driven IT Management. In Juha-Pekka Tolvanen, Matti Rossi, J. Gray, and J. Sprinkle, editors, *Proceedings of the 9th OOPSLA workshop on domain-specific modeling (DSM)*, pages 28–35, Helsinki, 2009. Helsinki Business School.

[Fou11] Eclipse Foundation. Eclipse IDE, 2011. `http://www.eclipse.org`, letzter Zugriff: 18.08.2011.

[Frö09] Jane Fröming. *Ein Konzept zur Simulation wissensintensiver Aktivitäten in Geschäftsprozessen*. GITO, 2009.

[Fra93] Ulrich Frank. Multiperspektivische Unternehmensmodellierung als Basis und Gegenstand integrierter CSCW-Systeme. In Ulrich Hasenkamp, S. Kirn, and Michael Syring, editors, *CSCW-Systeme*, pages 179–198. Addison-Wesley, 1993.

[Fra08] Ulrich Frank. The MEMO Meta Modelling Language (MML) and Language Architecture. Technical report, Universität Duisburg-Essen, 2008.

[FvL03] Ulrich Frank and Bodo van Laak. Anforderungen an Sprachen zur Modellierung von Geschäftsprozessen. Technical Report 34, Arbeitsberichte des Instituts für Wirtschaftsinformatik, Universität Koblenz-Landau, 2003.

[Gab08] Roland Gabriel. Informationssystem. In *Enzyklopädie der Wirtschaftsinformatik*. Karl Kurbel and Jörg Becker and Norbert Gronau and Elmar J. Sinz and Leena Suhl, 2008. `http://www.enzyklopaedie-der-wirtschaftsinformatik.de/wi-enzyklopaedie/lexikon/uebergreifendes/Kontext-und-Grundlagen/Informationssystem`.

[Gem07] Gemeinsame Landesplanungsabteilung der Länder Berlin und Brandenburg. Kulturlandschaften. Chancen für die regionale Entwicklung in Berlin und Brandenburg. Technical report, Ministerium für Infrastruktur und Raumordnung und Senatsverwaltung für Stadtentwicklung, 2007.

[GGH+07] Bastian Grabski, Sebastian Günther, Sebastian Herden, Lars Krüger, Claus Rautenstrauch, and André Zwanziger. Very Large Business Applications. *Informatik-Spektrum*, 30:259–263, 2007.

[GHS10] Sebastian Günther, Maximilian Haupt, and Matthias Splieth. Utilizing Internal Domain-Specific Languages for Deployment and Maintenance of IT Infrastructures. Technical report, Very Large Business Applications Lab Magdeburg, Fakultät für Informatik, Otto-von-Guericke-Universität Magdeburg, 2010.

[GK08] Bastian Grabski and Lars Krüger. System Landscape Methodology: Forschungsbedarf für VLBAs. *Multikonferenz Wirtschaftsinformatik*, 1:1877–1888, 2008.

[Goe08] Manfred Goeke. Der deutsche Mittelstand - Herzstück der deutschen Wirtschaft. In Manfred Goeke, editor, *Praxishandbuch Mittelstandsfinanzierung*, pages 9–22. Gabler, 2008.

[HF56] A.D. Hall and R.E. Fagen. Definition of system. *General Systems*, 1(1956):18–28, 1956.

[HHR04] Lutz J. Heinrich, Armin Heinzl, and Friedrich Roithmayr. *Wirtschaftsinformatik-Lexikon (7. Aufl.)*. Oldenbourg, 2004.

[Hok09a] Dorothea Hokema. Die Landschaft der Regionalentwicklung: Wie flexibel ist der Landschaftsbegriff? *Raumforschung und Raumordnung*, 67(3):239–249, 2009.

[Hok09b] Dorothea Hokema. Landschaft im Wandel? Zeitgenössische Landschaftsbegriffe in Wissenschaft, Planungspraxis und Alltag, 2009.

[HRV11] Till Haselmann, Christian Röpke, and Gottfried Vossen. Empirische Bestandsaufnahme des Software-as-a-Service-Einsatzes in kleinen und mittleren Unternehmen. Technical Report 131, Westfälische Wilhelms-Universität Münster, Institut für Wirtschaftsinformatik, 2011.

[HS09] Wolfgang Haber and Thies Schröder. Lebende Einheiten. In Sandra Kalcher, Thies Schröder, Ina Bimberg, Martin Janotta, Claus Käpplinger, Axel Klapka, Axel Lohrer, Andreas Müller, and Juttab Sankowski, editors, *System Landschaft*, pages 8–25. Birkhäuser Basel, 2009.

[HVH06] Bernhard Humm, Markus Voß, and Andreas Hes. Regeln für serviceorientierte Architekturen hoher Qualität. *Informatik-Spektrum*, 29(6):395–411, 2006.

[HWBH09] Daniel Hilkert, Christian M. Wolf, Alexander Benlian, and Thomas Hess. Das „as-a-Service“-Paradigma: Treiber von Veränderungen in der Software-Industrie. In Alexander Benalin, Thomas Hess, and Peter Buxmann, editors, *Software-as-a-Service : Anbieterstrategien, Kundenbedürfnisse und Wertschöpfungsstrukturen*, pages 57–74, 2009.

[Ins02] Institut für Mittelstandsforschung Bonn. KMU-Definition, 2002. `http://www.ifm-bonn.org/index.php?id=89`, letzter Zugriff: 18.05.2011.

[Ins09] Institut für Mittelstandsforschung Bonn. Arbeitsplatzdynamik und nachhaltige Beschäftigungswirkungen in kleinen und mittleren Unternehmen, 2009. http://www.ifm-bonn.org/assets/documents/Working-Paper-06-09.pdf, letzter Zugriff: 19.05.2011.

[Ins10] Institut für Mittelstandsforschung Bonn. Kennzahlen zum Mittelstand 2009/2010 in Deutschland. Online, 2010. http://www.ifm-bonn.org/index.php?id=99, letzter Zugriff: 19.05.2011.

[Ips06] D. Ipsen. *Ort und Landschaft*. VS Verlag für Sozialwissenschaften, 2006.

[ISO00] ISO/IEC. ISO/IEC 15288 Systems Engineering – Systems Life Cycle Processes, 2000.

[KD11] Karl Kurbel and Rastsislau Datsenka. Outsourcing des Systembetriebs, 2011. http://www.enzyklopaedie-der-wirtschaftsinformatik.de/wi-enzyklopaedie/lexikon/uebergreifendes/Globalisierung/Outsourcing/outsourcing-des-systembetriebs, letzter Zugriff: 27.09.2011.

[Küh08] Olaf Kühne. *Distinktion, Macht, Landschaft: zur sozialen Definition von Landschaft*. VS Verlag für Sozialwissenschaften, 2008.

[Kir03] Lutz Kirchner. Eine Sprache für die Modellierung von IT-Landschaften: Anforderungen, Potenziale, zentrale Konzepte. In *MobIS*, pages 69–86, 2003.

[Kle07] Ralf Klein. Analyse des Service Engineering Systems. In *Modellgestütztes Service Systems Engineering*, pages 59–120. DUV, 2007.

[KN00] Georg Kneer and Armin Nassehi. *Niklas Luhmanns Theorie sozialer Systeme: eine Einführung*. Uni-Taschenbücher S. Fink Wilhelm GmbH + Co.KG, 2000.

[Kom06] Europäische Kommission. Die neue KMU-Definition. Benutzerhandbuch und Mustererklärung, 2006. http://ec.europa.eu/enterprise/enterprise_policy/sme_definition/sme_user_guide_de.pdf, letzter Zugriff: 18.05.2011.

[KR72] F.E. Kast and J.E. Rosenzweig. General systems theory: Applications for organization and management. *The Academy of Management Journal*, 15(4):447–465, 1972.

[Krü07] Lars Krüger. Lebenslagen in Very Large Business Applications. Technical report, Very Large Business Applications Lab Magdeburg, Fakultät für Informatik, Otto-von-Guericke-Universität Magdeburg, 2007. http://www.vlba-lab.de/de/Home/Research/Topics/Circumstances/, letzter Zugriff: 22.09.2011.

[LSK04] Gerd Lutze, Alfred Schultz, and Joachim Kiesel. Landschaftsstruktur im Kontext von naturräumlicher Vorprägung und Nutzung – ein systemanalytischer Ansatz. *Landschaftsstruktur im Kontext von naturräumlicher Vorprägung und Nutzung–Datengrundlagen, Methoden und Anwendungen. IÖR-Schriftenreihe, Dresden, Bd*, 43:1–12, 2004.

[Luh84] Niklas Luhmann. *Soziale Systeme: Grundriss einer allgemeinen Theorie.* Suhrkamp, 1984.

[Mat08] Florian Matthes. Softwarekartographie. *Informatik-Spektrum*, 31:527–536, 2008.

[MEE+11] Sharon A. Mertz, Chad Eschinger, Tom Eid, Chris Pang, and Laurie F. Wurster. Forecast: Software as a Service, Worldwide, 2010-2015, 1H11 Update, Juni 2011. `http://www.gartner.com/it/page.jsp?id=1739214`, letzter Zugriff: 04.08.2011.

[Mei04] Stefan Meinhardt. *IT im Mittelstand.* dpunkt-Verlag, 2004.

[MG09] Peter Mell and Timothy Grance. The NIST definition of cloud computing. Technical Report 6, National Institute of Standards and Technology, 2009.

[MTK10] Jörn-Axel Meyer, Alexander Tirpitz, and Christian Koepe. *IT-Verhalten und -Defizite in KMU.* Josef Eul Verlag GmbH, 2010.

[MV80] Humberto R. Maturana and Francisco J. Varela. *Autopoiesis and cognition: the realization of the living.* Boston studies in the philosophy of science. D. Reidel Pub. Co., 1980.

[Obj10] Object Management Group. Unified Modeling Language: Superstructure, Version 2.4, 2010. `http://www.omg.org/spec/UML/2.4/Superstructure/Beta2/PDF`, letzter Zugriff: 10.08.2011.

[Pfo97] Hans-Christian Pfohl. Volkswirtschaftliche Bedeutung von Klein- und Mittelbetrieben. In Hans-Christian Pfohl, editor, *Betriebswirtschaftslehre der Mittel- und Kleinbetriebe*, volume 3, pages 27–51. Erich Schmidt Verlag, 1997.

[PHZ09] Susanne Patig, Sebastian Herden, and André Zwanziger. IT-Infrastruktur, 2009. `http://www.enzyklopaedie-der-wirtschaftsinformatik.de/wi-enzyklopaedie/lexikon/daten-wissen/Informationsmanagement/IT-Infrastruktur`, letzter Zugriff: 01.08.2011.

[PM10] Ferdinand Pavel and Anselm Mattes. Cloud-Computing: großes Wachstumspotenzial. *Wochenbericht des DIW Berlin*, 77(48):10–16, 2010.

[Pre07] Jacqueline Preußer. *Management der IT-Ressourcen: Aktuelle Entscheiderbefragung im Mittelstand und in großen Unternehmen.* Best of IT-Solutions. FAZ-Inst. für Manangement, Markt- und Medieninformationen, 2007.

[RHQ+07] Chris Rupp, Jürgen Hahn, Stefan Queins, Mario Jeckle, and Barbara Zengler. *UML 2 glasklar: Praxiswissen für die UML-Modellierung und -Zertifizierung*. Carl Hanser Verlag, 2007.

[RHS05] Jan-Peter Richter, Harald Haller, and Peter Schrey. Serviceorientierte Architektur. *Informatik-Spektrum*, 28(5):413–416, 2005.

[Roh08] Michael Rohloff. Ein Ansatz zur Beschreibung und zum Management von Unternehmensarchitekturen. *Multikonferenz Wirtschaftsinformatik (MKWI)*, pages 639–650, 2008.

[RS02] Claus Rautenstrauch and Thomas Schulze. *Informatik für Wirtschaftswissenschaftler und Wirtschaftsinformatiker*. Springer-Lehrbuch. Springer, 2002.

[Rud09] Simone Rudolph. Anwendungskontext mittelständische Unternehmen. In *Servicebasierte Planung und Steuerung der IT-Infrastruktur im Mittelstand*, pages 63–78. Gabler, 2009.

[Sch98] Reinhard Schütte. *Grundsätze ordnungsmäßiger Referenzmodellierung: Konstruktion konfigurations- und anpassungsorientierter Modelle.* nbf neue betriebswirtschaftliche Forschung. Gabler, 1998.

[Sie07] Johannes Siedersleben. SOA revisited: Komponentenorientierung bei Systemlandschaften. *Wirtschaftsinformatik*, 49(1):110–117, 2007.

[Sta73] Herbert Stachowiak. *Allgemeine Modelltheorie*. Springer-Verlag, Wien, 1973.

[Str10] Susanne Strahringer. Metamodell, 2010. `http://www.enzyklopaedie-der-wirtschaftsinformatik.de/wi-enzyklopaedie/lexikon/is-management/Systementwicklung/Hauptaktivitaten-der-Systementwicklung/Problemanalyse-/konzeptuelle-modellierung-von-is/metamodell`, letzter Zugriff: 04.08.2011.

[STS04] Michael Schmitt, Michael Trefz, and Steffen Schollh. Einführung der Unternehmenssoftware SAP Business One beim Logistikdienstleister Trefz. In *IT im Mittelstand* [Mei04], pages 67–75.

[Tar77] Alfred Tarski. *Einführung in die mathematische Logik*. Moderne Mathematik in elementarer Darstellung. Vandenhoeck & Ruprecht, 1977.

[Tas11] Distributed Management Taskforce. Common Information Model, 2011. `http://dmtf.org/standards/cim`, letzer Zugriff am 26.09.2011.

[Tho05] Oliver Thomas. *Das Modellverständnis in der Wirtschaftsinformatik: Historie, Literaturanalyse und Begriffsexplikation*, volume 184c. Institut für Wirtschaftsinformatik im Deutschen Forschungszentrum für Künstliche Intelligenz, 2005.

[Tur99] Klaus Turowski. Architekturkonzept zur Realisierung flexibel erweiterbarer Fachkomponenten. *Modellierung betrieblicher Informationssysteme. PROCEEDINGS der MobIS-Fachtagung*, pages 132–152, 1999.

[vB68] Ludwig von Bertalanffy. General system theory. *New York: George Braziller*, 1968.

[vB03] Jan vom Brocke. *Referenzmodellierung: Gestaltung und Verteilung von Konstruktionsprozessen*. Advances in information systems and management science. Logos-Verl., 2003.

[VRMCL08] Luis M. Vaquero, Luis Rodero-Merino, Juan Caceres, and Maik Lindner. A break in the clouds: towards a cloud definition. *SIGCOMM Comput. Commun. Rev.*, 39:50–55, December 2008.

[WKW94] WKWI. Profil der Wirtschaftsinformatik. *Wirtschaftsinformatik*, 38:80, 1994.

[ZE08] Peter Zencke and Rüdiger Eichin. SAP Business ByDesign - Die neue Mittelstandslösung der SAP. *Wirtschaftsinformatik*, 50(1):47–51, 2008.

[Zsc95] Dietrich Zschocke. *Modellbildung der Ökonomie.: Modell, Information, Sprache.* Vahlen Handbücher der Wirtschafts- u. Sozialwissenschaften. Vahlen, 1995.

[Zwa09] André Zwanziger. Vergleich bestehender Modellierungssprachen für IT-Infrastrukturen. In *Very Large Business Applications (VLBA): Systemlandschaften der Zukunft*, pages 112–123. Shaker Verlag, Aachen, Deutschland, 2009.

Anhang A

Metamodell

Entitäten des Metamodells

In Tabelle A.1 ist eine Auflistung aller Entitäten des Metamodells sowie eine kurze Beschreibung dieser Entitäten zu sehen.

Entität	**Erläuterung**
Software	Abstrakte Klasse, welche die generischen Eigenschaften von Software kapselt.
SoftwareComponent	Stellt eine Komponente / ein Modul von Software dar.
ApplicationSoftware	Eine explizite Anwendung der abstrakten Klasse *Software*, die Anwendungssoftware abbildet.
SystemSoftware	Eine explizite Anwendung der abstrakten Klasse *Software*, die Systemsoftware abbildet.
SoftwareEnvironment	Gruppierung von Systemsoftware, um die Abbildung von Abhängigkeiten zwischen Anwendungssoftware und Systemsoftware zu vereinfachen.
Service	Fest definierte Leistung, die als Element eines oder mehrerer größerer Verarbeitungsabläufe verwendet werden kann und von *ApplicationSoftware* bereitgestellt oder genutzt wird.
SLA	Service Level Agreement, dass die Rahmenbedingungen für einen Service festlegt.
Hardware	Abstrakte Klasse, welche die generischen Eigenschaften von Hardware kapselt.
HardwareComponent	Beschreibt ein Bauteil / einen Bestandteil eines Hardwareverbundes.
Location	Ist ein physischer Ort, an dem Hardware lagert.
Server	Eine explizite Ausprägung der abstrakten Klasse *Hardware*, die Server repräsentiert.
VirtualMachine	Software, die Hardware emuliert und auf einem `Server` ausgeführt wird.
Workstation	Eine explizite Ausprägung der abstrakten Klasse *Hardware*, die einen Personal Computer repräsentiert.

Entität	Erläuterung
Device	Eine explizite Ausprägung der abstrakten Klasse *Hardware*, die ein Gerät innerhalb der Systemlandschaft repräsentiert, das weder `Workstation` noch `Server` ist.
Network	Ein Netzwerk, dass die Entitäten von *Hardware* untereinander verbindet.
CloudComputing	Abstrakte Klasse, welche die generischen Eigenschaften von Software kapselt.
SaaS	Realisierung von `CloudComputing`, welche die Cloud Computing Ebene „SaaS“ darstellt.
IaaS	Realisierung von `CloudComputing`, welche die Cloud Computing Ebene „IaaS“ darstellt.
PaaS	Realisierung von `CloudComputing`, welche die Cloud Computing Ebene „PaaS“ darstellt.
UserRole	Repräsentiert einen Nutzer innerhalb der Systemlandschaft.
BusinessProcess	Repräsentiert einen Geschäftsprozess in einem Unternehmen.
InformationEntity	Repräsentiert ein Geschäftsobjekt, das durch einen `BusinessProcess` definiert wird.
Configuration	Eine Konfigurationsbeschreibung von Hardware und Software.
ConfigurationAttribute	Ein Parameter innerhalb einer `Configuration`

Tabelle A.1: Auflistung und Erläuterung aller Entitäten des Metamodells

Zusammenstellung der vorgeschlagenen Attributierung

Tabelle A.2 zeigt eine Zusammenstellung aller Attribute, die im Rahmen von Kapitel 4 vorgeschlagen wurden.

Entität	Attribut	Beschreibung
Software	`Name`	Bezeichnung der Software.
	`Version`	Aktuelle Versionsnummer der Software
	`Manufacturer`	Hersteller der Software.
	`Licence_Date`	Datum, an dem die Lizenz abläuft.
	`Serial`	Seriennummer der Software.
SoftwareComponent	`Name`	Bezeichnung der Komponente.
	`Version`	Aktuelle Versionsnummer der Software.
	`Manufacturer`	Hersteller der Komponente.
ApplicationSoftware	–	–
SystemSoftware	–	–
SoftwareEnvironment	–	–

Service	`Operations`	Liste von Funktionen, die der Service anbietet.
SLA	–	–

Hardware	`Name`
	`Manufacturer`
	`CPU-Cores`
	`CPU-Speed`
	`Memory`
	`Storage`
	`IP`
		Bezeichner der Hardware.
		Name des Herstellers.
		Anzahl der Kerne einer CPU.
		Geschwindigkeit eines Kerns in GHz.
		Verfügbarer physischer Arbeitsspeicher.
		Kapazität der Festplatten in GB.
		IP der Hardware.

HardwareComponent	`Name` `Manufacturer`	Bezeichnung der Komponente.

Location	`Name`
	`PhysicalLocation`
	`Address`
		Bezeichner des Standortes.
		Name des Herstellers.
		Seriennummer der Software.

Server	–	–
VirtualMachine	–	–
Workstation	–	–
Device	–	–

Network	`Name`
	`StartAddress`
	
	`EndAddress`
	`Subnetmask`
		Eindeutiger Bezeichner des Netzwerks.
		IP-Adresse, ab der Adressen vergeben werden.
		IP-Adresse, bis zu der vergeben werden.
		Gibt die Präfixlänge einer IP-Adresse an.

CloudComputing	`Name`
	`URL`
	`IP`
	`Provider`
	`DeploymentModel`
		Bezeichner des Dienstes
		URL des Dienstes.
		IP des Dienstes.
		Anbieter des Dienstes.
		Definiert die Bereitstellungsart, z. B. „Public“ (vergleiche Abschnitt 2.4).

SaaS	–	–

IaaS	`Type`
	`Size`
	
	`AutoScaling`
		Kategorie des Dienstes.
		Unterkategorie von `Type`, welche die Ressourcen genauer spezifiziert (z. B. Grösse des Arbeitsspeichers).
		Kapazität der Festplatten in GB.

PaaS	`Features`	Liste der bereitgestellten Funktionalitäten.
UserRole	–	–
BusinessProcess	–	–
InformationEntity	–	–
Configuration	`Version`	Versionsnummer einer Konfiguration.
ConfigurationAttribute	–	–

Tabelle A.2: Mögliche Attributierung aller Entitäten des Metamodells

Vollständiges Metamodell

Abbildung A.1 zeigt alle Entitäten des Metamodells sowie deren Relationen.

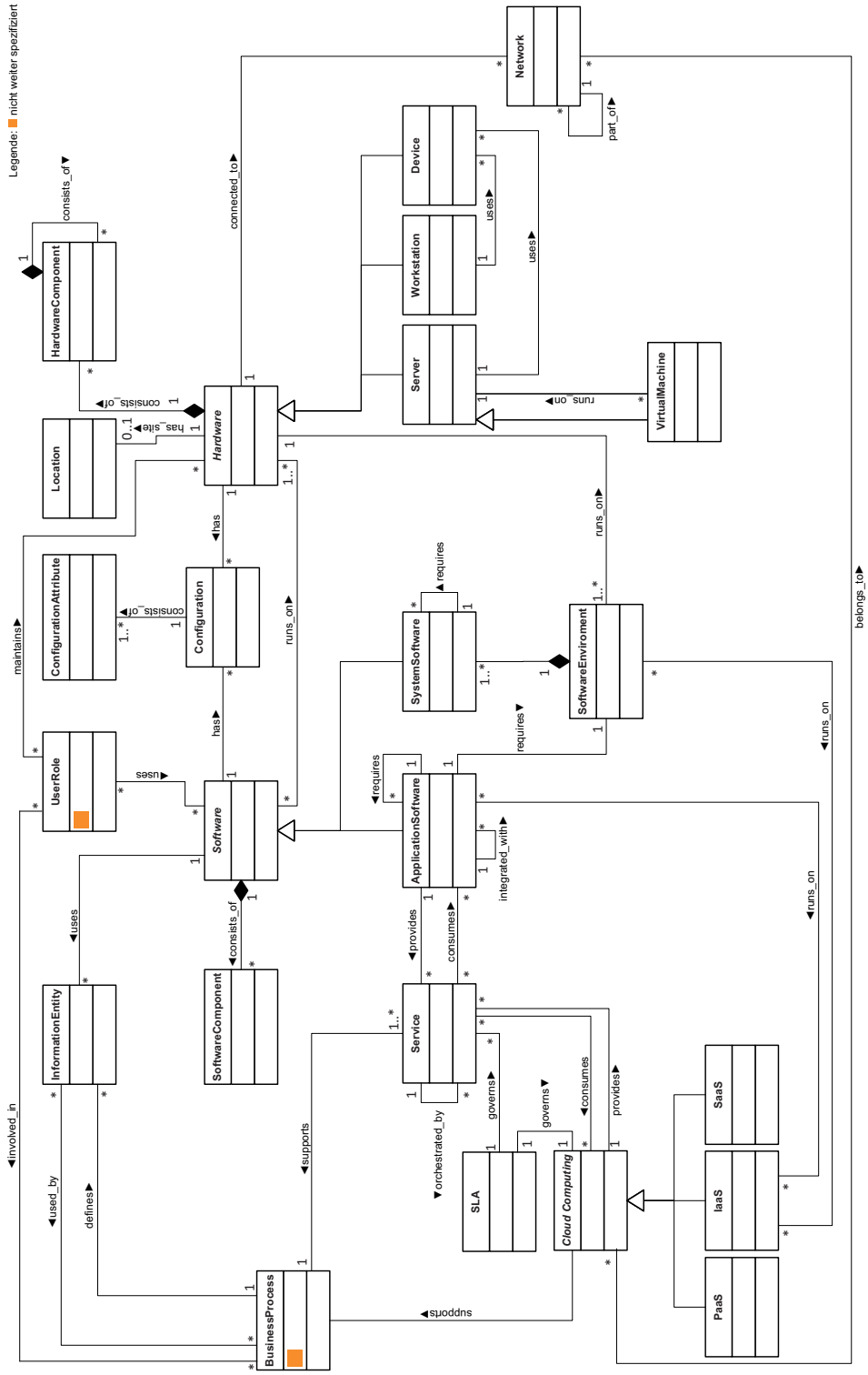

Abbildung A.1: Entwickeltes Metamodell

Selbstständigkeitserklärung

Hiermit erkläre ich, dass ich die vorliegende Arbeit selbstständig und nur mit erlaubten Hilfsmitteln angefertigt habe.

Magdeburg, den 13. Oktober 2011

Matthias Splieth