Dienstorientierte Integration von Managementwerkzeugen

zur Erlangung des akademischen Grades eines

Doktors der Ingenieurwissenschaften

von der Fakultät für Informatik
des Karlsruher Instituts für Technologie (KIT)

genehmigte

Dissertation

von

Ingo Pansa

aus Tübingen

Tag der mündlichen Prüfung:	03. November 2011
Erster Gutachter:	Prof. Dr. Sebastian Abeck
Zweiter Gutachter:	Prof. Dr. Bernhard Neumair

"Es ist besser, ein Licht anzuzünden, als über die Dunkelheit zu klagen."

Dankeschön.

Die vorliegende Arbeit entstand während meiner Tätigkeiten am Karlsruher Institut für Technologie (KIT), genauer genommen am Lehrstuhl *Cooperation&Management* (C&M), Institut für Telematik, Fakultät für Informatik, sowie in der *Abteilung Technische Infrastruktur* (ATIS). Und wie bei allen wissenschaftlichen Arbeiten mit vergleichbarem Ausmaß auch wäre diese Arbeit nicht möglich gewesen, wenn eine Vielzahl verschiedener Personen mich hierbei nicht unterstützt hätte.

Aus diesem Grund möchte ich daher an dieser Stelle zunächst und ausdrücklich meinem Doktorvater und Referenten Prof. Dr. Sebastian Abeck danken. Nicht nur war er jederzeit ein interessierter Zuhörer und kritischer Fragesteller, sondern hat mir auch stets Vertrauen, Hilfe, Unterstützung und Feedback gegeben, den vielen kleinen und großen Herausforderungen während meiner wissenschaftlichen Arbeit zu begegnen. Prof. Dr. Bernhard Neumair danke ich herzlich für die Übernahme des Korreferats.

Meinem ehemaligen Mentor, Chef und Förderer Klaus Scheibenberger gebührt an dieser Stelle besonderer Dank. Er gab mir bereits während meines Studiums die Chance, eigenständig in ein komplexes Themengebiet einzutauchen und hat somit das Fundament zu den Ergebnissen in der vorliegenden Arbeit gelegt. Danken möchte ich auch Dr. Christian Mayerl dafür, dass ich im Jahre 2007 in seiner „Arbeitsgruppe IT-Management“ einen tiefergehenden Einblick in spannende Forschungsfragestellungen in dieser Domäne erhalten konnte.

Meinen (ehemaligen) Kollegen am Lehrstuhl (Dr. Michael Gebhart, Philip Hoyer, Aleksander Dikanski, Dr. Stefan Link, Dr. Christof Momm und Dr. Christian Emig) danke ich nicht nur für die vielen spannenden Diskussionen in unserer gemeinsamen Zeit, sondern besonders für die positive Atmosphäre, den Teamgeist, die Professionalität, die Leidenschaft, die Offenheit und die Ehrlichkeit, die jederzeit Spaß gemacht haben und einige schwierige Phasen erträglicher erschienen ließen.

Meinen Kollegen in der ATIS danke ich vor allem dafür, dass sie auch in kritischen Zeiten immerzu meine Arbeit direkt und indirekt mit unterstützt haben.

Die von mir betreuten Studenten haben zu der vorliegenden Arbeit unmittelbar beigetragen, da ich in der Betreuung ihrer Diplom- und Studienarbeiten, Praktika und Seminare nicht nur wesentliche Erkenntnisse vertiefen konnte, sondern auch in spannenden Diskussionen viele Aspekte von unterschiedlichen Perspektiven her analysieren konnte. Ich möchte hier an dieser Stelle daher besonders danken: Philip Walter, Felix Palmen, Matthias Reichle, Christoph Leist, Lukas Menzel, Vasileios Dutsias, Selim Ok, Saswat Mohanty und Michael Völlinger.

Zuletzt – aber nicht an letzter Stelle – danke ich meiner Familie, meinen Eltern Dieter und Elisabeth, meiner Schwester Rebecca und meiner zukünftigen Frau Nadja. Sie gaben mir stets die Kraft, den Mut und die Zuversicht, die Arbeit voranzutreiben und abzuschließen. Ohne sie wäre dies alles nicht möglich gewesen.

Karlsruhe, November 2011

Kurzfassung

Zielgerichtete Lösungen zur Realisierung fachlicher Anforderungen bestehen heutzutage aus einer Vielzahl unterschiedlicher Netz-, System- und Anwendungskomponenten, die als Gesamtheit betrieben und als IT-Dienst bereitgestellt werden. Die Verantwortung für einen IT-Dienst liegt dabei beim Dienstleister. Dieser setzt für den Betrieb in der Regel unterschiedliche, teils hochgradig spezialisierte Managementwerkzeuge ein, die von unterschiedlichen Fachabteilungen innerhalb einer Dienstleisterorganisation genutzt werden. Eine Integration dieser Managementwerkzeuge in eine offene Plattform ermöglicht eine ganzheitliche Sicht auf die zu betreibenden IT-Dienste. Hierbei stand bislang eine Integration der Werkzeuge für den technisch-orientierten Betrieb der Infrastruktur im Vordergrund. Neuere Ansätze definieren darauf aufbauend betriebliche Prozesse, um eine kooperative Abarbeitung anfallender Aufgaben über die Grenzen verschiedener Abteilungen hinweg zu ermöglichen. Die Beherrschung durch eine automatisierte Ausführung dieser teilweise sehr komplexen Prozesse stellt die Grundlage dar, um den Betrieb von IT-Diensten effektiv, effizient und an Unternehmenszielen ausgerichtet zu ermöglichen. Der Einsatz einer dienstorientierten Architektur zur Gestaltung einer offenen Plattform ermöglicht hierbei die automatisierte Ausführung von betrieblichen Prozessen auf Basis bestehender Managementwerkzeuge. Zentraler Gedanke dieses Ansatzes ist es, eine Implementierung der betrieblichen Prozessen durch Managementdienste zu realisieren, gleichzeitig aber auch flexibel und anpassbar auf zukünftige Änderungen reagieren zu können. Besondere Eigenschaften dieser Managementdienste – wie beispielsweise Standardisierung, Prozessorientierung und Wiederverwendbarkeit – spielen daher eine wichtige Rolle.

Bestehende Arbeiten im Kontext des Entwurfes von Managementdiensten sind entweder lediglich auf die Integration technischer Managementinformation beschränkt oder betrachten einzelne abgeschlossene Managementbereiche oder Managementwerkzeuge. Eine Unterstützung des Entwicklungsvorgehens durch geeignete Modelle für den nachvollziehbaren Entwurf standardisierter, prozessorientierter und wiederverwendbarer Managementdienste wird bislang nicht betrachtet.

Ziel der Arbeit ist es daher, anhand eines systematischen und nachvollziehbaren Entwicklungsvorgehens die für den Entwurf von standardisierten, prozessorientierten und wiederverwendbaren Managementdienste relevanten Modellaspekte und -überführungen zu beschreiben. Hierzu liefert die vorliegende Arbeit zwei grundlegende Beiträge.

Im ersten Beitrag werden die zu entwerfenden Managementdienste betrachtet. Im Mittelpunkt steht hierbei ein Metamodell der Domäne IT-Management, das die für den Entwurf von Managementdiensten mit den genannten Aspekten abstrahiert. Das Metamodell wird zunächst auf Basis einer Taxonomie beschrieben und anschließend durch den Einsatz der Web Ontology Language (OWL) konkret spezifiziert. Auf Basis dieses Metamodells können Analysemodelle erstellt werden, die die spätere Grundlage für den Entwurf von zu implementierenden Managementdiensten bilden. Die Überführung dieser Analysemodelle in Dienstmodelle wird durch den Einsatz der von der Object Management Group (OMG) vorgestellten Service-oriented Architecture Modeling Language (SoaML) unterstützt. Die Eigenschaft der Wiederverwendbarkeit der entwickelten Managementdienste ist von zentraler Bedeutung, um die anfallenden Integrationsaufgaben zu lösen und flexibel und anpassbar auf zukünftige Anforderungen reagieren zu können. Als ergänzender Beitrag wird daher aufgezeigt, inwiefern verschiedene Aspekte von Wiederverwendbarkeit an den SoaML-basierten Dienstmodellen bestimmt werden können. Drei ausgewählte Aspekte werden hierbei durch die Angabe formaler Berechnungsvorschriften vertieft betrachtet.

Im zweiten Beitrag wird ein systematisches und nachvollziehbares Entwicklungsvorgehen vorgestellt, das den Entwurf von Managementdiensten beschreibt, die zur Ausführung von Betreiberprozessen genutzt werden können. Das vorgestellte Entwicklungsvorgehen vertieft und erweitert bestehende Ansätze für den Entwurf dienstorientierter Architekturen. Hierdurch wird sowohl die Integration bestehender Managementwerkzeuge in eine offene Plattform ermöglicht, als auch eine Unterstützung zur automatisierten Ausführung von Betreiberprozessen geschaffen. Auf Basis des im ersten Beitrag spezifizierten Metamodells der Domäne wird aufgezeigt, wie Referenzmodelle für verschiedene Managementbereiche auf der Grundlage internationaler Standards konstruiert und in einem durchgehenden Entwicklungsvorgehen eingesetzt werden können. Für die an der Störungsbearbeitung beteiligten Managementbereiche werden Referenzmodelle in Form einer OWL-Ontologie vorgestellt. Bestandteil der vorgestellten Methode ist die Modellierung der Betreiberprozesse durch die vorab erstellten Domänenmodelle. Diese werden im weiteren Verlauf systematisch auf SoaML-Dienstmodelle überführt. Die Beschreibung der einzelnen Modellüberführungen erfolgt systematisch und unabhängig von konkreten Szenarien und kann hierdurch in verschiedenen Entwicklungsprojekten zum Einsatz kommen.

Die Tragfähigkeit des vorgestellten Ansatzes wird an Beispielen in der Anwendung in verschiedenen Entwicklungsprojekten demonstriert, die im Rahmen der wissenschaftlichen Tätigkeit durchgeführt wurden. Konkret wird an einem Prototyp-Projekt zur Konstruktion von prozessorientierten Werkzeugschnittstellen für Dienste der Störungsbearbeitung aufgezeigt, wie vorhandene Managementwerkzeuge im Kontext der automatisierten Ausführung von Incident- und Problem-Management-Prozessen gemäß ISO20000-1:2005 genutzt werden können. Die entworfenen prototypischen Managementdienste stellen die Grundlage dar, um eine systemunterstütze Abarbeitung kritischer Aufgaben innerhalb der Abteilung technische Infrastruktur (ATIS) am Karlsruher Institut für Technologie (KIT) auf Basis der bestehenden Managementwerkzeuge zu ermöglichen.

Insgesamt liefert die vorliegende Arbeit einen Beitrag, um den Entwurf von Managementdiensten durch geeignete Referenzmodelle nachvollziehbar zu gestalten. Die dem Metamodell zugrunde liegende Taxonomie der Domäne bildet die Grundlage, weitergehende Standardisierungsbemühungen hinsichtlich einer offenen, auf prozessorientierten und wiederverwendbaren Managementdiensten basierenden Plattform konstruktiv zu unterstützen. Vor allem der Einsatz von standardisierten Modellierungssprachen stellt die praktische und nachvollziehbare Anwendbarkeit des Ansatzes sicher. Die Bewertung der Wiederverwendbarkeit der mit dem vorgestellten Ansatz entworfenen Managementdienste liefert beteiligten Entwicklern die Möglichkeit, bereits frühzeitig in den Analyse- und Entwurfsmodellen wichtige Eigenschaften von Managementdiensten zu bestimmen und die weiteren Entwicklungsschritte entsprechend anzupassen.

Inhaltsverzeichnis

1 Einleitung

1.1 Motivation

Dem rechnergestützten Austausch von Informationen kommt heutzutage eine stetig wachsende Bedeutung zu. So ist vor allem in der Geschäftswelt der Einsatz von Rechnern und Rechnernetzen nicht mehr wegzudenken, um dem steigenden Bedarf an automatisierten Verfahren zu begegnen. Beispielsweise verfügen 96 % aller Unternehmen mit mehr als 250 Mitarbeitern in Deutschland über ein eigenes betriebsinternes Netzwerk [DSB09], um die Verarbeitung und den Austausch von Daten und die Koordination von betrieblichen Abläufen zu unterstützen. Die Gesamtheit aller rechnergestützten Verfahren wird als Informationstechnik (engl. *Information Technology*, IT) bezeichnet. Der wachsenden Abhängigkeit von IT steht die Notwendigkeit gegenüber, den Betrieb der Rechner und Rechnernetze sicherzustellen und darüber hinausgehend den fachlichen Anforderungen der Unternehmen möglichst optimal anzupassen. Gleichzeitig soll die IT als beherrschbarer Kostenfaktor für ein Unternehmen einplanbar sein [Ca03]. Diese als IT-Management bezeichnete Domäne umfasst alle Maßnahmen für einen unternehmenszielorientierten, effektiven und effizienten Betrieb eines verteilten Systems mit seinen Ressourcen [HA+99].

Die Notwendigkeit, die einzelnen Aspekte dieser Domäne zu strukturieren ergibt sich aus den vielfältigen Anforderungen, die an den Betrieb dieser Systeme gestellt werden können. So bestehen zielgerichtete Lösungen zur Realisierung fachlicher Anforderungen heutzutage aus einer Vielzahl unterschiedlicher Komponenten (Netzkomponenten, Systemkomponenten und Softwarekomponenten), die als Gesamtheit betrieben und als IT-Dienst bereitgestellt werden [Ma99]. Die Verantwortung für einen Dienst liegt dabei beim Dienstleister. Dieser setzt für den zielgerichteten Betrieb in der Regel eine Vielzahl unterschiedlicher, teils hochgradig spezialisierter Managementwerkzeuge ein. Diese stellen Funktionalität bereit, um auf Managementinformation zugreifen zu können oder betriebliche Aspekte einer Komponente abzudecken (Überwachung, Steuerung). So existieren verschiedene Werkzeuge zur Überwachung und Steuerung von Netzwerken und deren Komponenten (z. B. Überwachung zugesicherter Bandbreiten), zur Überwachung und Steuerung von Serversystemen (z. B. Überwachung von Festplattenauslastungen) oder zur Überwachung und Steuerung von Softwareanwendungen (z. B. Überwachung von Warteschlangen). Wenngleich die einzelnen Werkzeuge für sich allein genommen genutzt werden, ist offensichtlich, dass die Gesamtheit aller Managementwerkzeuge für den Betrieb eines verteilten Systems wiederum ein verteiltes System darstellt. Analog zur Forderung, die Funktionalität und die Datenbasis der einzelnen Komponenten des zu betreibenden Systems anhand eines integrierten Gesamtkonzeptes zu betrachten, ist eine integrierte Gesamtsicht für die Belange des IT-Managements wünschenswert [SM+06]. Hierbei interessieren vor allem Interoperabilitätsstandards zur Sicherstellung der Integrationsfähigkeit der einzelnen Werkzeuge. Historisch gesehen entwickelten sich für einzelne Managementbereiche unterschiedliche und teilweise sogar gegenläufige Konzepte [HA+99, Pa07]. Der Wunsch, diese einzelnen Konzepte anhand zielgerichteter Eigenschaften miteinander in Einklang zu bringen ist demnach mindestens genauso alt, wie die Anforderung, Konzepte für den Betrieb verteilter Systeme zu entwickeln.

Die dienstorientierte Architektur (engl. *Service-oriented Architecture*, SOA) setzt sich als Ansatz für den architektonischen Gesamtentwurf von verteilten Softwaresystemen in jüngster Zeit verstärkt durch [Ar04]. Zentraler Gedanke der dienstorientierten Architektur ist es, die fachliche Unterstützung von Geschäftsprozessen durch lose gekoppelte, autonome und wiederverwendbare Softwaredienste zu realisieren [KB+05, DJ+05, SS08, Er08]. Hierbei werden prozessbezogene komponierte Dienste durch Verschalten von Basisdiensten realisiert. Die Basisdienste stellen oft Integrationsschnittstellen zu bestehenden Anwendungen dar und ermöglichen hierdurch den flexiblen Zugriff auf existierende Daten und Funktionen in einer Gesamtarchitektur, die von den zugrunde liegenden Technologien abstrahiert und somit bezüglich zu unterstützender Prozesse eine einfachere Anpassung bei sich ändernden Rahmenbedingungen ermöglicht. Der Vorteil bei diesem Ansatz liegt darin, dass die Softwaredienste einer dienstorientierten Architektur unabhängig voneinander entwickelt, betrieben oder gepflegt werden können. Durch den klaren Geschäftsbezug können prozessorientierte Ansätze durch automatisierende Systeme umgesetzt werden. Durch die Integration bestehender Softwaresysteme kann ein einfacher Übergang zu diesem Ansatz ermöglicht werden. Von besonderem Interesse sind die vielfältigen Möglichkeiten bei der Konstruktion von Schnittstellen zu bestehenden Softwareanwendungen sowie die Gestaltung von dienstorientierten Architekturen für spezielle Domänen.

Aufgrund der Eigenschaften der dienstorientierten Architektur als Softwarearchitektur zur Integration von bestehenden Anwendungen kann diese als wesentlicher Baustein bei der Gestaltung eines ganzheitlichen Ansatzes auch im IT-Management genutzt werden [MB+04, Ku05, Pa07]. Der einfache Ansatz, eine Integration auf Ebene der Informationsartefakte der einzelnen Managementwerkzeuge zu realisieren (z. B. durch Kopieren von Managementinformation von einem Werkzeug in mehrere andere Werkzeuge) lässt sich bereits heute technisch einfach realisieren [Li03]. Dieser einfache Integrationsansatz führt jedoch schnell zu einer unüberschaubaren Anzahl einzelner Integrationspunkte [HA+99]. In [CH+05] wird daher vorgeschlagen, eine Integration bestehender Anwendungen unter Hinzunahme der Funktionalität dieser Anwendungen anzustreben. Diese Funktionalität orientiert sich zunächst nicht an den funktionalen Fähigkeiten der Werkzeuge, sondern wird anhand von Vorgaben entwickelt, die aus fachlichen Anforderungen motiviert werden können. Diese als Basisdienste bezeichneten Dienste können komponiert und zu höherwertiger Funktionalität verschaltet werden. Diese komponierten Dienste bilden dann idealerweise die zu unterstützenden Managementprozesse und ermöglicht somit eine Automatisierung dieser Abläufe. Um den Entwurf sowohl dieser Basisdienste als auch der komponierten Dienste zu unterstützen, ist ein Vorgehen notwendig, dass bestehende Managementansätze einschließt, darauf aufbauend aber ein hinreichend abstrahiertes Fundament besitzt, um den dienstorientierten Ansatz in beliebigen Szenarien anwenden zu können. Bestehende Ansätze aus dem Bereich der umfassenden Managementreferenzarchitekturen betrachten vor allem die Unterstützung von technischen Aspekten. Eine durchgängige und technologieunabhängige Vorgehensweise, die die wesentlichen Aspekte beim Entwurf einer dienstorientierten Softwarearchitektur im Kontext des Betriebes vernetzter IT-Dienste einbezieht, wird bislang nicht verfolgt. Die vorliegende Arbeit nimmt diese Feststellung auf und beschreibt anhand eines systematischen Vorgehens den Entwurf sowohl grundlegender Managementbasisdienste als auch den Entwurf von komponierten Managementprozessdiensten, die zur Ausführung von Betreiberprozessen notwendigen sind.

1.2 Gegenstand der vorliegenden Arbeit

Die Organisation von betrieblichen Abläufen wird bei IT-Dienstleistern bereits heute in Form von Betreiberprozessen durchgeführt [HA+99]. Im Gegensatz zu den Vorgaben der existierenden standardisierten Rahmenwerke sind die Betreiberprozesse jedoch oftmals nicht ausdrücklich definiert, dokumentiert oder an Vorgaben aus Standardspezifikationen ausgerichtet [GQ+07]. Sehr häufig werden hier teilweise komplette Betreiberprozesse auf die von einem einzelnen Werkzeug angebotene Funktionalität umgesetzt. Es entsteht eine zu enge Kopplung zwischen der fachlichen Ausrichtung der Betreiberprozesse auf der einen Seite und der technologischen Ausrichtung der Managementwerkzeuge auf der anderen Seite. Eine Umstrukturierung auf Ebene der Betreiberprozesse mit dem Ziel, Teilschritte zu automatisieren, wird somit unnötig erschwert. Die Unterstützung aufkommender Paradigmen wie beispielsweise dem *Cloud Computing* [FE10, BK11] erfordert jedoch genau diese Flexibilität, so dass sich ändernde Anforderungen seitens des Kerngeschäfts einfach durch eine Anpassung von betrieblichen Abläufen abdeckt werden kann.

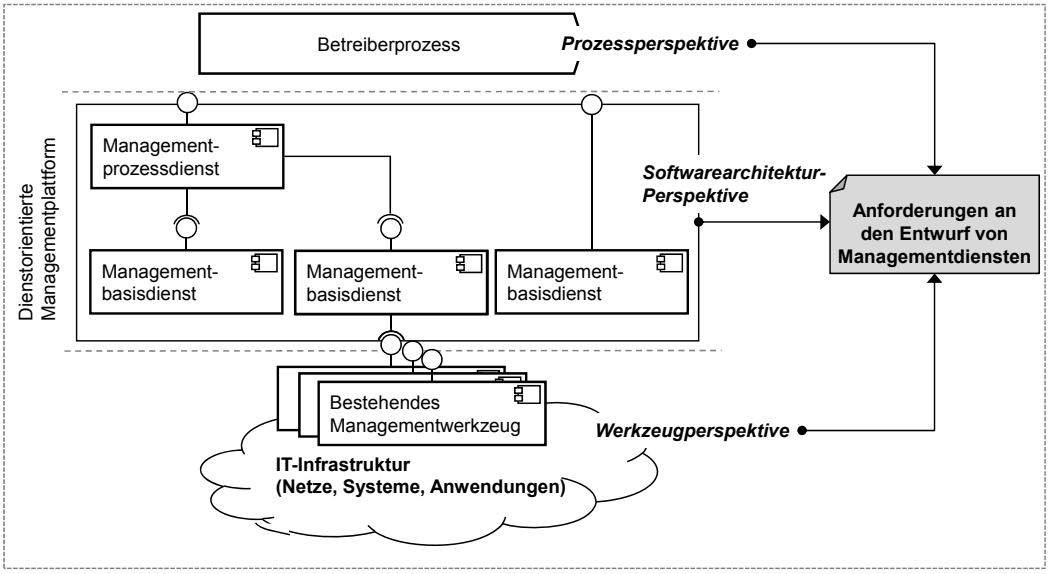

Abbildung 1 Gegenstand der Arbeit: Entwurf von Managementdiensten

Ein möglicher Ansatz zur Lösung dieser Herausforderung besteht in der Einführung von prozessorientierten Dienstschnittstellen zu den bestehenden Managementwerkzeugen [CH+05, Ku05, Pa07, To06, MS+07, MV+05, MT+06]. Hierbei wird von der Information der einzelnen Managementwerkzeuge abstrahiert und auf die Erstellung von Zugriffspunkten fokussiert, die die Funktionalität der einzelnen Werkzeuge an den Anforderungen der zu unterstützenden Betreiberprozesse ausrichtet. Diese beiden Aspekte werden als **dienstorientierte Integration von Anwendungen** (engl. *Service-oriented Application Integration*) zur Beschreibung des Entwurfes der einzelnen Dienste bzw. als **geschäftsprozessorientierte Integration von Anwendungen** (engl. *Business Process-oriented Application Integration*) zur Beschreibung der Umsetzung der zu realisierenden Betreiberprozesse bezeichnet. In Abbildung 1 finden sich diese beiden Aspekte in Form der Prozessperspektive sowie in der Form der Werkzeugperspektive wieder. Die Perspektive der

Softwarearchitektur verbindet diese beiden Perspektiven dahingehend, dass durch den Begriff der Architektur eine Strukturierung des die Prozesse realisierenden Softwaresystems eingeführt wird. Hierbei sind verschiedene Aspekte zu beachten, die insgesamt einen ganzheiltichen Blick auf die zu entwerfende Architektur unterstützen [Kr95].

Im Fokus der dienstorientierten Integration steht die Idee, eine gemeinsame Geschäftslogik auf eine implementierungsunabhängige Art und Weise bereitzustellen [Li03]. Dabei werden diejenigen Geschäftsobjekte innerhalb einer Organisation identifiziert, die im Rahmen von Geschäftsprozessen wiederverwendet werden können. Für die Integration bestehender Managementwerkzeuge bedeutet dies, die relevanten Managementobjekte auf Ebene der Managementwerkzeuge zu identifizieren, die in weiteren Managementprozessen wiederverwendet werden können. Neben der Managementinformation der einzelnen Werkzeuge wird ebenfalls die angebotene Funktionalität betrachtet.

Die geschäftsprozessorientierte Integration kann als konsequente Fortentwicklung der dienstorientierten Integration aufgefasst werden, indem neben der Möglichkeit der Wiederverwendung auf Ebene der Anwendungen auch komplexe Aktivitäten gekapselt und in weiteren Betreiberprozessen genutzt werden [Li03]. Eine separate Betrachtung beider Teilaspekte ist sinnvoll, da die dienstorientierte Integration die Betrachtung der Integration bestehender Anwendungen als Gegenstand hat, wohingegen die geschäftsprozessorientierte Integration den Aspekt der organisatorischen Strukturen fokussiert. Eine Integration auf Ebene der Benutzerschnittstellen (portalorientierte Anwendungsintegration, engl. *Portal-oriented Application Integration*) wird in der vorliegenden Arbeit nicht weiter thematisiert. Die speziellen Anforderungen an die Betrachtung der Benutzerinteraktion in dienstorientierten Architekturen werden beispielsweise in [Li09] diskutiert.

Wie in Abbildung 1 dargestellt, ergeben sich aus den drei betrachteten Perspektiven jeweils spezifische Anforderungen an den Entwurf von Managementdiensten. Während von der Prozessperspektive die Anforderung abgeleitet werden kann, dass die Schnittstellen der zu entwerfenden Dienste insgesamt eine fachliche Ausrichtung besitzen sollten, bedingt die Werkzeugperspektive die Analyse der grundlegenden, in der Domäne erforderlichen Managementfunktionen, um so ein gemeinsames Verständnis über eine einheitliche Begriffsbildung der anzubietenden Managementfunktionen zu schaffen. Die Perspektive der Softwarearchitektur letztlich führt zu Anforderungen, die die grundlegende Strukturierung der zu entwerfenden Managementdienste betreffen.

Zusammengefasst kann festgestellt werden, dass die Integration bestehender Managementwerkzeuge die Ausführung automatisierbarer Betreiberprozesse unterstützen kann [MV+05, MT+06]. Hierzu ist jedoch ein systematisches, nachvollziehbares und in der Praxis anwendbares methodisches Vorgehen erforderlich, dass die wesentlichen Anforderungen der drei betrachteten Perspektiven einschließt.

1.3 Problemstellungen

Die eingesetzten Managementwerkzeuge stellen in der Regel keine Schnittstellen zur Verfügung, die dienstorientiert sind und somit die erforderliche betreiberprozessorientierte und wiederverwendbare

Ausrichtung nicht aufweisen. Dies stellt eine durchgängige Methode zum Entwurf einer flexiblen Unterstützung für automatisierbare Betreiberprozesse vor große Herausforderungen und führt zu wesentlichen Fragestellungen.

Definition grundlegender Funktionalität für Managementdienste

Als Grundlage zur Gestaltung einer auf den Prinzipien der dienstorientierten Architektur basierenden Managementplattform wird die Anhebung der bestehenden Managementwerkzeugschnittstellen in Form von Managementdiensten benötigt. Hierbei werden die Funktionen der Dienste sowie die beim nachrichtenorientierten Funktionsaufruf notwendigen Informationen durch eine standardisierte Syntax beschrieben. Während die Anhebung von Werkzeugfunktionen durch den Entwurf von dienstorientierten Schnittstellen bereits heute prinzipiell möglich ist (z. B. durch den Entwurf von Webservice-basierten Schnittstellen [DMTF-WS-Management, OASIS-MUWS]), führt die direkte Umsetzung zu Managementdiensten, die nicht die erforderliche Orientierung am zu unterstützenden Betreiberprozess aufweisen oder nicht wiederverwendbar sind. Es muss demnach sichergestellt werden, dass sich die entworfenen Dienste an den für die Domäne IT-Management verfügbaren Standards orientieren, um eine Integration von Managementwerkzeugen an den zu unterstützenden Betreiberprozessen zu erleichtern [Ku05].

Integration von Managementwerkzeugen

Die Integration von Managementwerkzeugen wird technisch gesehen durch die Nutzung von Adapter-Schnittstellen realisiert. Hierbei werden Transformationen der von den Werkzeugen angebotenen Schnittstellen auf die aus der Architektur vorgegeben benötigten Schnittstellen vorgenommen. Dabei muss neben der Abbildung von Werkzeugfunktionen auch die Abbildung der Ein- und Ausgabeparameter dieser Funktionen betrachtet werden. Dieser Schritt wird vor allem dadurch erschwert, dass die vorhandenen Managementwerkzeuge die jeweilige Managementinformation anhand unterschiedlicher Informations- und Datenmodelle beschreiben [SB08]. Eine fachliche Abstraktion der Informationen ist hierdurch nicht direkt möglich. Erste Arbeiten hinsichtlich der Erfassung von Managementinformation durch semantische Ansätze sind verfügbar [VV+02], betrachten jedoch nicht die Definition funktionaler und an Standards ausgerichteter Managementdienste. Daneben ist es erforderlich, nicht nur grundlegende Werkzeugfunktionalität zu beschreiben und in einen nachvollziehbaren Integrationsprozess zu überführen, sondern darüber hinausgehend auch komplexe Werkzeugfunktionen, die als Implementierung zusammengesetzter Managementaktivitäten genutzt werden können.

Modellierung automatisierbarer Betreiberprozesse

Den Ausgangspunkt bei der Umsetzung eines automatisierten und prozessorientierten Ansatzes für den Betrieb von vernetzten Infrastrukturen stellt der zu automatisierende Betreiberprozess dar. Als Voraussetzung für eine durchgängige Methode für den Entwurf von Managementdiensten muss daher eine geeignete Modellierung von Managementprozessen vorliegen [IW+94, Sc99, CN04]. Von einem konzeptionellen Standpunkt aus betrachtet stellen Betreiberprozesse zunächst lediglich eine Spezialisierung gewöhnlicher Geschäftsprozesse dar [GD+09b,Da06, Br06], wodurch auch bereits bekannte Modellierungsansätze (z. B. *Business Process Modeling Notation* (BPMN) [OMG-BPMN],

UML-Aktivitätsdiagramme [OMG-UML-Super], annotierte Petri-Netze [AS11]) oder XML-basierte Prozessbeschreibungssprachen (z. B. *Web Services Business Process Execution Language* (WSBPEL) [OASIS-WSBPEL]) zum Einsatz kommen können. Von Vorteil sind der klare Domänenbezug bei Managementprozessen und damit die Möglichkeit, die Verfügbarkeit von unterschiedlichen Rahmenwerken zur Spezifikation der notwendigen Aktivitäten, Rollen oder Prozessentitäten zu nutzen. Daraus können einige grundlegende Anforderungen an die Modellierung zur Darstellung von Betreiberprozessen abgeleitet werden. Danciu [Da06] führt beispielsweise eine Kategorisierung bekannter Ansätze anhand typischer Merkmale von operationellen IT-Managementprozessen ein. Hamm [Ha09] ordnet Modellierungsansätze unter dem Aspekt der interorganisatorischen Verantwortung von verketteten IT-Diensten. In beiden Arbeiten wird eine Modellierung der Betreiberprozesse mit dem Ziel vorgenommen, Managementwerkzeuge für diese Prozesse neu zu konstruieren. Die Integration bestehender Werkzeuge wird nicht thematisiert. Durch diese entkoppelte Betrachtung wird eine durchgängige Methode erschwert, da Detailinformation der in den Prozessen eingesetzten Werkzeuge bereits frühzeitig zur Verfügung stehen würde.

Definition und Umsetzung komponierter Managementdienste

Bestehende Arbeiten betrachten die Automatisierung von Betreiberprozessen vor allem durch den Entwurf schwergewichtiger Informationssysteme. Dies hat den Nachteil, dass durch diese enge Kopplung sowohl die unterstützen Prozesse als auch die unterstützenden Softwaresysteme nur schwer an sich ändernde Rahmenbedingungen abgepasst werden können. Die Abbildung von modellierten Betreiberprozessen auf leichtgewichtige, durch Komposition aus bestehenden Managementbasisdiensten realisierte Komponenten, wird bislang nicht diskutiert. Damit eine modellgetriebene Integration bestehender Anwendungen wie beispielsweise in [HG+09, HG+10] beschrieben unterstützt werden kann, ist neben der Betrachtung der Modellierung der zu automatisierenden Betreiberprozesse vor allem auch die Definition einer nachvollziehbaren Umsetzung erforderlich.

1.4 Zielsetzung und Beiträge der Arbeit

Zusammengefasst kann festgestellt werden, dass der Wechsel hin zu einer dienstorientierten Sichtweise bei der Nutzung von IT-Systemen durch den Einsatz von standardisierten und offenen Ansätzen die flexible Verschaltung von Fachfunktionalität ermöglicht hat, vergleichbare Ansätze aber aber in der Domäne IT-Management bislang nicht hinreichend genau betrachtet wurden. Hierbei rückt vor allem die Betrachtung der der Standardisierung, Prozessorientierung und Wiederverwendbarkeit als wesentliche Eigenschaften von Managementdiensten zur Unterstützung von automatisierbaren Betreiberprozessen in den Vordergrund. Bestehende Arbeiten betrachten die für ein nachvollziehbares und wiedrrholt anwendbares Entwicklungsvorgehen erforderlichen Modellbildungen bislang nicht auf Basis geeigneter Domänenmodelle, wodurch eine Integration bestehender Managementwerkzeuge nach den Prinzipien der dienstorientierten Architektur erschwert wird.

Das **Ziel der vorliegenden Arbeit** besteht daher darin, den Entwurf von standardisierten, prozessorientierten und wiederverwendbaren Managementdiensten systematisch und nachvollziehbar zu beschreiben um somit eine Automatisierung von betrieblichen Abläufen auf Basis der bestehenden

Managementwerkzeuge zu ermöglichen. Um dieses Ziel zu erreichen, liefert die vorliegende Arbeit zwei Beiträge.

Beitrag B1: Metamodell für den Entwurf wiederverwendbarer Managementdienste

Die Nachvollziehbarkeit von relevanten Entwurfsentscheidungen ist in der Entwicklung komplexer Softwaresysteme wichtig. Modelle unterstützen den Entwurf von Software dahingehend, dass eine geeignete Abstraktion des zu entwerfenden komplexen Systems eine Dekomposition in handhabbare Teilsysteme ermöglicht [CN04, BD09]. Automatisierte Überführungen von Modellen zur schrittweisen Verfeinerung hinsichtlich einer gültigen Implementierung gewinnen in letzter Zeit verstärkt an Aufmerksamkeit [HG+09, HG+10, SV+07, OMG-MDA]. Aufgrund der Heterogenität bestehender Managementanwendungen und der Vielschichtigkeit der vorgenannten Problemstellungen muss jedoch zunächst gefordert werden, den Entwurf eines auf Managementdiensten basierenden Managementsystems überhaupt systematisch beschreiben und modellieren zu können [Sc99, BD09], bevor einzelne Entwurfsschritte durch eine automatische Transformation umgesetzt werden können. Während bestehende Entwicklungsvorgehen den objektorientierten Entwurf von Softwarekomponenten betrachten (z. B. [BD09]), bringt die Ausrichtung auf den Entwurf von dienstorientierten Architekturen zusätzliche Anforderungen an das zu entwickelnde System [TL+09]. Um einen fachlichen Bezug entworfener Dienste bezüglich der zu unterstützenden Prozessen erstellen zu können, ist die Modellierung der den Prozessen zugrunde liegenden Domäne erforderlich [EH+08]. Hierzu wird ein einheitliches und auf bestehenden Standards basierendes Metamodell der Domäne erforderlich, damit die Ergebnisse von Entwicklungsprozessen über verschiedene Szenarien hinweg nachvollzogen werden können. Ein einheitliches Metamodell der Domäne stellt letztlich auch den Ausgangspunkt dar, um den Entwurf hinsichtlich Wiederverwendbarkeit und Prozessorientierung zu lenken. Das Metamodell genügt den Anforderungen an ein integriertes Modell für die dienstorientierte Analyse und den dienstorientierten Entwurf. Bestehende Arbeiten betrachten diesen Aspekt bislang nicht in ausreichendem Maße.

Im Kern des ersten Beitrages steht daher ein **Metamodell der Domäne**, das die Semantik der einzelnen Modellelemente klar definiert und zueinander in Beziehung setzt. Das Metamodell wird auf Basis einer OWL-Ontologie spezifiziert [De02, Be04, GD+09, TS+06, W3C-OWL]. Die grundlegenden **Aspekte von Wiederverwendbarkeit** werden bezüglich dieses Metamodells analysiert und am Beispiel von ausgewählten Aspekten formal beschrieben.

Beitrag B2: Domänengetriebener Entwurf wiederverwendbarer Managementdienste

Die automatisierte Ausführung von Betreiberprozessen erfordert sowohl Managementinformation als auch Funktionalität bestehender Managementwerkzeuge. Die Konstruktion entsprechender Schnittstellen ist erforderlich. Diese sollen, zur Begegnung der identifizierten Problemstellungen, prozessorientiert sein, d. h., unabhängig von konkreten Technologien, Implementierungen oder dedizierten Plattformen [MV+05, MT+06]. Während der erste Beitrag die wesentlichen strukturellen Elemente eines Domänenmodells für die Analyse und den Entwurf wiederverwendbarer und prozessorientierter Managementdienste liefert, wird im zweiten Beitrag der systematische Einsatz dieses Ansatzes zur Lösung der beschriebenen Probleme betrachtet. Mithilfe dieses Domänenmodells können fachlich-motivierte Modelle zur Erfassung von Anforderungen definiert werden und im

weiteren Verlauf eines Softwareentwicklungsprozesses zunächst in Dienstkandidatenmodelle und letztlich in konkrete Dienstmodelle umgesetzt werden.

Im Kern des zweiten Beitrages steht daher ein **methodisches Vorgehen**, betreiberprozessorientierte und wiederverwendbare Managementdienste auf Basis des vorgestellten Metamodells zu entwerfen und für die Abbildung automatisierbarer Betreiberprozesse einzusetzen.

Tragfähigkeitsnachweis

Zur Demonstration der Tragfähigkeit werden die Betreiberprozesse im Rahmen der Störungsbearbeitung bei einem akademischen Dienstleister (Abteilung Technische Infrastruktur (ATIS) der Fakultät für Informatik) untersucht. Anhand der bestehenden Managementwerkzeuge wird die Integration bezüglich der Orientierung an den zu unterstützenden Betreiberprozessen aufgezeigt, wobei herausgearbeitet wird, inwiefern der vorgestellte domänengetriebene Entwicklungsansatz die Wiederverwendbarkeit von Managementdiensten günstig beeinflussen kann.

Konkret werden am Beispiel der Prozesse *Incident Management* [ISO05] und *Problem Management* [ISO05] zunächst Domänenmodelle definiert, um darauf aufbauend die zur Unterstützung dieser Prozesse notwendigen Managementdienste zu entwerfen. Zielsetzung bei diesem Projekt war unter anderem die Integration der in der ATIS eingesetzten bestehenden Managementwerkzeuge. Auf Basis des vorgestellten Ansatzes zur Bewertung der Wiederverwendbarkeit der entworfenen Dienste wird die Qualität der eigenen Lösung in einem realen Entwicklungsprozess evaluiert.

1.5 Prämissen der Arbeit

Die vorliegende Arbeit thematisiert die Integration von Managementwerkzeugen mit dem Ziel, automatisierbare Betreiberprozesse zu realisieren. Da der Fokus der Arbeit auf der Beschreibung eines systematischen Entwicklungsvorgehens sowie der hierzu erforderlichen Modelle liegt, werden einige grundlegende Annahmen getroffen, die die Tragfähigkeit insgesamt sicherstellen. Im Ausblick der Arbeit wird an ausgewählten Prämissen aufgezeigt, welche weiteren Verbesserungen auf der Grundlage der hier vorgestellten Ergebnisse möglich sind.

Prämisse P1: Webservice-basierte Implementierung

In der Vergangenheit wurden unterschiedliche Zielplattformen vorgeschlagen, um Managementarchitekturen gemäß den geforderten Bedingungen zu entwickeln. Allen ist gemein, dass jeweils eigenständige Anwendungsprotokolle mit jeweils eigenständiger Syntax zur Beschreibung der Kommunikationsstrukturen eingesetzt wurden. Eine durchgehende Sicherstellung von Dienstqualität über Betreibergrenzen hinweg wird somit erschwert, da für unterschiedliche Ansätze Managementübergänge notwendig werden [Ke98]. Mit der Durchdringung des Internets und der hieraus resultierenden Nutzung der dem Internet zugrunde liegenden Anwendungsprotokolle (wie dem *Hypertext Transfer Protocol* (HTTP) [RFC2616], *extensible Markup Language* (XML) [W3C-XML]) entwickelte sich mit der Webservice-Technologie eine Möglichkeit, verteile Softwarearchitekturen auf Basis offener und standardisierten Ansätze zu realisieren [AL+03, DJ+05]. Als Zielplattform zur

Realisierung der Managementdienste werden daher Webservices zur Implementierung herangezogen [Ku05, Pa07].

Prämisse P2: Modellierung von Diensten mit SoaML

Die Unterstützung der Modellierung von Diensten auf Basis standardisierter Modellierungssprachen ist gegenwärtig aktueller Betrachtungsgegenstand vieler unterschiedlicher Forschungsansätze. Beispielsweise können strukturelle Aspekte von Dienstkomponenten oder Dienstschnittstellen mit UML-Klassendiagrammen [OMG-UML-Super] modelliert werden, die Nutzung von Interaktionsprotokollen mit UML-Sequenz- oder UML-Aktivitätsdiagrammen [OMG-UML-Super] oder die Beziehungen in einer komplexen Dienstarchitektur mit UML-Kompositionsstrukturdiagrammen [OMG-UML-Super]. Mit der Arbeit an einem standardisierten, auf Basis der UML konzipierten Modellierungsansatz wird die Hoffnung verbunden, die vom Standpunkt der Softwareentwicklung relevanten Aspekte in einer integrierten Modellierungssprache zu vereinigen. Die Entwicklung dieser Modellierungssprache für Dienste und Dienstarchitekturen (*Service-oriented architecture Modeling Language*, SoaML [OMG-SoaML]), wird maßgeblich von der OMG vorangetrieben. Obgleich des frühen Spezifikationsstadiums einer Standarddefinition der SoaML existiert bereits eine Unterstützung in vielen unterschiedlichen Entwicklungswerkzeugen.

Prämisse P3: Entwurf von Anwendungsdiensten

Die Etablierung dienstorientierter Architekturen schließt auch eine Betrachtung des vollständigen Lebenszyklus der durch die Elemente der Architektur realisierten Dienste ein. Hier treten neben Fragestellungen hinsichtlich des eigentlichen Entwurfes und Implementierung der Komponenten auch wirtschaftlich motivierte Fragestellungen auf. Hierzu zählen beispielsweise Geschäftszielvereinbarungen, der geschäftliche Nutzen eines Dienstes oder rechtliche Rahmenbedingungen für den Betrieb eines Dienstes. Prinzipiell können Geschäftsdienste von Anwendungsdiensten unterschieden werden [EH+08]. Anwendungsdienste bezeichnen diejenigen Elemente, die direkt durch Softwareartefakte umgesetzt werden können und wesentliche Eigenschaften dienstorientierter Paradigmen aufweisen (lose Kopplung, Wiederverwendbarkeit, Nachrichtenorientierung). Geschäftsdienste bauen darauf auf, beziehen jedoch auch geschäftsorientierte Aspekte mit ein, die nicht direkt in Bezug zur Entwicklung von Softwaresystemen stehen. Die vorliegende Arbeit betrachtet daher ausschließlich den Entwurf von Anwendungsdiensten.

Prämisse P4: Betrachtung fachfunktionaler Anforderungen

Da für die Betrachtung qualitativer Anforderungen an ein Managementsystem vor allem auch Laufzeitaspekte eines fertigen Managementsystems beachtet werden müssen, wird durch die Fokussierung auf die Analyse und den Entwurf in der vorliegenden Arbeit vor allem die Umsetzung von fachfunktionalen Anforderungen betrachtet.

1.6 Aufbau der Arbeit

Eine kurze Einführung in das Themengebiet sowie grundlegende Problemstellungen und die zur Adressierung dieser Problemstellungen vorgeschlagenen Lösungen werden in **Kapitel 1** dargelegt. In

Kapitel 2 werden die Grundlagen dieser Arbeit eingeführt sowie grundlegende Begriffe definiert. Zur Bewertung bestehender Ansätze hinsichtlich der notwendigen Anforderungen zur Lösung der genannten Problemstellungen wird in **Kapitel 3** ein Anforderungskatalog erarbeitet, anhand dessen Ansätze aus Forschung und Praxis bezüglich deren Schwächen evaluiert werden können.

Darauf aufbauend werden in Kapitel 4 und 5 die Beiträge der vorliegenden Arbeit präsentiert. In **Kapitel 4** wird ein Metamodell der Domäne IT-Management vorgestellt, mit dessen Hilfe betreiberprozessorientierte Managementdienste entworfen werden können. Das Metamodell unterstützt wesentliche Aspekte der dienstorientierten Analyse und des dienstorientierten Entwurfes. Es werden verschiedene Aspekte der Wiederverwendbarkeit von Managementdiensten untersucht. In **Kapitel 5** wird auf Basis dieses Metamodells die Integration bestehender Managementwerkzeuge in die entworfenen Managementdienste beschrieben. Hierzu wird ein Ansatz vorgestellt, der die Konstruktion von betreiberprozessorientierten Dienstschnittstellen zu bestehenden Managementwerkzeugen ermöglicht, um diese Werkzeuge als Implementierung sowohl von Managementbasisdiensten als auch von Managementprozessdiensten nutzen zu können. **Kapitel 6** demonstriert die Tragfähigkeit, indem konkrete Anwendungen des vorgestellten Ansatzes vorgestellt werden. Eine kurze Zusammenfassung der Arbeit, eine kritische Diskussion der eigenen Ergebnisse sowie ein Ausblick mit dem Hinweis auf weitergehende Fragestellungen erfolgt in **Kapitel 7**. In den **Anhängen** befinden sich ein Glossar, ein Abkürzungsverzeichnis, ein Verzeichnis der referenzierten Literatur sowie ein Index, ein Abbildungsverzeichnis und ein Tabellenverzeichnis.

1.7 Typografische Konventionen und Rechtschreibung

Die Rechtschreibung in der vorliegenden Arbeit entspricht den Vorgaben des zum Erstellungszeitpunkt gültigen Duden [Du10]. Die Sprache der Wortschrift ist Deutsch. Fachbegriffe sind, sofern eine sinngemäße Übersetzung vorliegt, übersetzt. Ausgenommen von dieser Festlegung sind Fachbegriffe, die Elemente von international anerkannten Standards bezeichnen.

Folgende typografischen Konventionen werden zur Verdeutlichung von bestimmten herausragenden Sachverhalten innerhalb der Wortschrift festgelegt:

- **Fettschrift** kennzeichnet wichtige Sachverhalte.
- *Kursivschrift* kennzeichnet Ausdrücke aus fremden Sprachen sowie Zitate.
- `Courierschrift` kennzeichnet Code-Auszüge sowie Bezüge zu Elementen von Modellen.

Diese Konventionen gelten für die Textabschnitte der Arbeit, nicht jedoch für Überschriften, Bildelemente und Bildunterschriften. Ausgenommen sind ebenfalls die Elemente aus den Anhängen.

2 Grundlagen

In diesem Kapitel werden die wesentlichen Grundlagen beschrieben, die zum Verständnis der vorliegenden Arbeit erforderlich sind. Auf weitergehende Aspekte wird jeweils durch die Angabe von entsprechenden Literaturhinweisen verwiesen.

2.1 Der Begriff IT-Dienst

Zentraler Begriff in der vorliegenden Arbeit ist der Begriff **Dienst**. Da der Dienstbegriff in verschiedenen Formen und in verschiedenen Bedeutungen zum jeweiligen eingesetzten Kontext erscheint (Managementdienst, dienstorientierte Architektur, dienstorientierte Integration, dienstorientierte Analyse und Entwurf), erfolgt zunächst eine Beschreibung des generischen Dienstbegriffes anhand wesentlicher und anerkannter grundlegender Definitionen.

Unabhängig von einer konkreten Domäne kann der Begriff Dienst bzw. der Begriff Dienstleistung gemäß dem Deutschen Institut für Normierung (DIN) in DIN EN 9000:2005 wie folgt aufgefasst werden:

> *„als das Ergebnis mindestens einer Tätigkeit, die notwendigerweise an der Schnittstelle zwischen dem Lieferanten und dem Kunden ausgeführt wird und üblicherweise immateriell ist.“*

Zentrale Aussage ist die Tatsache, dass das vom Kunden gewünschte Ergebnis an einer klar definierten Schnittstelle geliefert wird. Die Tätigkeit, die zum gewünschten Ergebnis führt, wird durch die Schnittstelle vor dem Kunden verborgen (*Black-Box*-Prinzip) und obliegt der alleinigen Verantwortung des Lieferanten. Diese Beziehung zwischen Dienstnehmer (Kunde) und Dienstgeber (Lieferant) ist wesentliches Charakteristikum eines dienstorientierten Ansatzes.

In ITIL Version 3 ist der Begriff Dienst (engl. *Service*) bezüglich des Mehrwerts für den Nutzer definiert [OGC07]:

> *„Ein Service ist eine Möglichkeit, einen Mehrwert für Kunden zu erbringen, indem das Erreichen der von den Kunden angestrebten Ergebnisse erleichtert oder gefördert wird. Dabei müssen die Kunden selbst keine Verantwortung für bestimmte Kosten oder Risiken tragen.“ [OGC07]*

Für ein Verständnis des in der vorliegenden Arbeit zugrunde gelegten Dienstbegriffes kann auf diesen Definitionen aufbauend eine einfache Taxonomie eingeführt werden. Einer der zentralen Begriffe hinsichtlich der Abstraktion von durch IT-Systeme angebotenen Funktionalitäten stellt der auf dem generischen Dienstbegriff verfeinerte Begriff **IT-Dienst** dar. Im Wesentlichen ist ein IT-Dienst ein Dienst, der durch den Einsatz von Informationstechnologie erbracht wird. Ein IT-Dienst kann als logische Komponente angesehen werden, die die Funktionalität eines IT-Systems mit (nicht notwendigerweise) untereinander vernetzten Elementen ganzheitlich erbringt. Die einzelnen Elemente dieses Systems können wiederum aus IT-Diensten bestehen. Neben dem reinen funktionalen Aspekt dieser logischen Komponente wird zusätzlich ein qualitativer Aspekt mit einbezogen. Hierdurch hebt

sich der Begriff IT-Dienst vom Begriff der (Software-)Komponente ab, da durch die Orientierung am Dienstbegriff der qualitative Aspekt inhärent verknüpft ist. Dies entspricht in etwa der Definition des Begriffes Anwendungsdienst aus [EH+08] und kann somit vom Begriff Geschäftsdienst durch die Fokussierung auf die durch Softwarekomponenten erbrachte Funktionalität eingegrenzt werden.

Die Nutzung der Funktionalität wird durch die Definition und Bekanntgabe einer verbindlichen Schnittstelle ermöglicht. Die Definition der einzelnen Dienstfunktionen erfolgt auf Basis fachlicher Anforderungen und abstrahiert daher von der zugrunde liegenden Implementierung, deren Technologie sowie sämtlichen plattformspezifischen Details. Dienste sind daher immer fachlich motiviert. Um die Funktionalität eines Dienstes nutzen zu können, muss neben deren Schnittstellenspezifikation auch ein Protokoll definiert werden, das die Art und Weise vorgibt, wie die einzelnen Funktionen genutzt werden können.

Abbildung 2 gibt eine grafische Übersicht über eine einfache Taxonomie des Begriffes IT-Dienst.

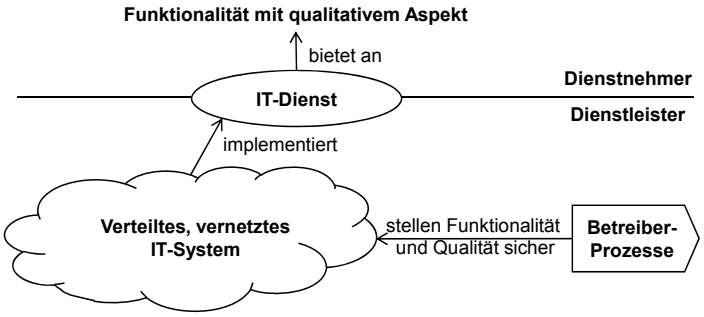

Abbildung 2 Einfache Taxonomie des Begriffes IT-Dienst mit den wesentlichen Aspekten

Prinzipiell lassen sich zwei grundlegende Rollen (Dienstnehmer und Dienstleister) unterscheiden. Der Dienstnehmer bezieht die von einem IT-Dienst angebotene Funktionalität und nutzt diese, um geschäftlich motivierte Ziele zu erreichen. Der Dienstleister ist verantwortlich für die Erbringung der Funktionalität hinsichtlich vereinbarter und zugesicherter qualitativer Aspekte. In der Regel wird zwischen Dienstnutzer und Dienstleister eine Vereinbarung getroffen, die die Funktionalität eines IT-Dienstes in Verbindung mit den zugehörigen qualitativen Aspekten vertraglich fixiert. Durch diesen Ansatz soll eine Art Verlässlichkeit für den Dienstnehmer geschaffen werden. Diese Vereinbarung (Dienstleistungsvereinbarung, DLV, engl. *Service Level Agreement*, SLA) ist demnach der Ausgangspunkt für einen Dienstleister, um einen zielgerichteten Betrieb der den Dienst erbringenden IT-Systeme zu realisieren. Der Dienstleister strukturiert die hierfür notwendigen Aktivitäten in Form definierter Betreiberprozesse (Managementprozesse), um so die vielfältigen Aspekte für den Betrieb abzudecken.

Die in den DLV vereinbarten qualitativen Aspekte können unterschiedlicher Natur sein, orientieren sich idealerweise jedoch – ebenso wie die Spezifikation der Dienstfunktionalität – an den Anforderungen der Dienstnehmer. Diese Anforderungen entstammen der Analyse der Geschäftsziele der Dienstnehmer und werden daher in der Regel aus einer Betrachtung der Geschäftsprozesse

abgeleitet, in denen diese IT-Dienste genutzt werden. Gängige Beispiele aus der Praxis für eine Verbindung aus fachlich-motivierter Funktionalität und zugehörigem qualitativen Aspekt tendieren jedoch häufig dazu, technische Eigenschaften einer Funktionalität in Bezug zu den qualitativen Aspekten zu stellen.

2.2 Dienstorientierte Architektur

Die strukturellen Zusammenhänge in verteilten Systemen lassen sich sehr gut auf Basis des Architekturbegriffes darstellen. Eine (Software-)Architektur bezeichnet die Menge der einzelnen Elemente eines Softwaresystems sowie deren Beziehungen untereinander [RH06]. Historisch gesehen haben sich unterschiedliche Stile von architektonischen Mustern ausgeprägt [AL+03]. Während sich in den Frühzeiten des Einsatzes elektronischer Rechensysteme vor allem entkoppelte Mainframe-Systeme identifizieren lassen, ist mit der beginnenden Vernetzung von Systemen Ende der 1980er-Jahre eine Abkehr von dem Ansatz, wenige leistungsstarke Systemen hin zu einem Verbund von prinzipiell gleichwertigen Systemen einzusetzen, erkennbar. Einen ersten Höhepunkt erlebt dieser Ansatz durch das Auftreten von objektorientierten Programmiersprachen zur Implementierung von Objekten, die durch den Einsatz standardisierter Netzwerkprotokolle in einem verteilten System genutzt werden können.

Einer der populärsten Vertreter dieses Ansatzes ist die von der *Object Management Group* (OMG) spezifizierte *Common Object Request Broker Architecture* (CORBA) [OMG-CORBA]. Im Kern der Architektur von CORBA steht ein zentraler Bus, der zum Austausch von Nachrichten zwischen den an den Bus angeschlossenen Objekten vermittelt. Diese Objekte werden als autonome Komponenten aufgefasst, die eine klar definierte Schnittstelle aufweisen und eine von der Schnittstelle verborgene Implementierung besitzen. Durch diesen Ansatz kann die Implementierung geändert werden, ohne Auswirkung auf die definierte Schnittstelle zu zeigen. Um einen einfachen Grad an Abstraktion der zugrunde liegenden Objektimplementierung zu erzielen, werden die Schnittstellen zu den einzelnen Objekten mit einer eigenen Auszeichnungssprache – der *Interface Description Language* (IDL) [OMG-CORBA] – beschrieben. Wesentlicher Nachteil von CORBA ist die enge Bindung an implementierungsnahe Konzepte der objektorientierten Softwareentwicklung sowie die schlechte Skalierbarkeit von CORBA-basierten Anwendungen im Internet-Maßstab [He08].

Eine logische Fortentwicklung der dem CORBA-Ansatz zugrunden liegenden Paradigmen mündet in der Idee, die einzelnen Elemente eines verteilten Systems in Form lose gekoppelter Dienste mit klarem fachlichen Bezug umzusetzen. Handelt es sich in der Gesamtheit der Elemente dieser Architektur um Dienste, spricht man von einer dienstorientierten Architektur (engl. *Service-oriented Architecture*, SOA) [DJ+05]. Hierbei wird der komponentenorientierte Ansatz der einzelnen Elemente beibehalten, jedoch werden die Schnittstellen an fachlichen Anforderungen der zu unterstützenden Geschäftsprozesse ausgerichtet. Mit dem fachlichen Bezug der Dienste wird weiterhin eine Dienstnehmer-/Dienstgeberbeziehung eingeführt, die über die einfache Verteilung der Komponenten in einem Netzwerk hinausgeht. Wesentliches Konzept bei der Nutzung von Diensten ist daher neben der Betrachtung von funktionalen Eigenschaften auch die Einbeziehung von nicht-funktionalen Eigenschaften wie beispielsweise Qualitätsaspekte oder betriebliche Zusicherungen.

In der verfügbaren Literatur (z. B. in [DJ+05, MS+07, KB+05, Er06]) werden unterschiedliche Perspektiven auf den Begriff der dienstorientierten Architektur eingenommen. Daher ist es unerlässlich, anerkannte Definitionen vor dem Hintergrund der in Kapitel 1 genannten Problemstellungen zu betrachten sowie darauf aufbauend eine für den Kontext der vorliegenden Arbeit eindeutige Begriffsbildung des Begriffes der dienstorientierte Architektur festzulegen.

So führen Bieberstein et al. im Jahre 2006 folgende Definition ein [BB+06]:

> *"A service-oriented architecture is a framework for integrating business processes and supporting IT infrastructure as secure, standardized components – services – that can be reused and combined to address changing business priorities."*

Die *Organization for the Advancement of Structured Information Standards* (OASIS) definiert den Begriff SOA – ebenfalls im Jahre 2006 – wie folgt [OASIS-SOA-RM, OASIS-SOA-RA]:

> *"Service Oriented Architecture (SOA) is a paradigm for organizing and utilizing distributed capabilities that may be under the control of different ownership domains."*

Es ist ersichtlich, dass mit dem Begriff SOA zunächst keine technische Lösung betrachtet wird, sondern vielmehr ein Ansatz, um Fähigkeiten vernetzter, verteilter IT-Infrastrukturen an den fachlichen Anforderungen auszurichten. Dies wird insbesondere in der Definition von Dostal, Jeckle et al. ersichtlich [DJ+05]:

> *„Unter einer SOA versteht man eine Systemarchitektur, die vielfältige, verschiedene und eventuel inkompatible Methoden oder Applikationen als wiederverwendbare und offen zugreifbare Dienste repräsentiert und dadurch eine plattform- und sprachenunabhängige Nutzung und Wiederverwendung ermöglicht.*

Da der fachliche Bezug, sowie die Wiederverwendbarkeit und die Orientierung an den zu unterstützenden Prozessen bei den genannten Definitionen ganz klar in den Vordergrund gestellt wird, kann somit eine für die Domäne IT-Management gültige Definition festgelegt werden. Die nachfolgende Definition deckt sich mit den Definitionen zu grundlegenden Arbeiten im Bereich dienstorientierter Ansätze im IT-Management [Ku05, Pa07].

Definition für den Kontext dieser Arbeit:

> *"Eine dienstorientierte Architektur im Kontext des Betriebes von IT-Systemen ist eine verteilte Softwarearchitektur, die sowohl fachliche Anforderungen seitens standardisierter Betreiberprozesse einbezieht sowie Konzepte bestehender Managementarchitekturen und Managementwerkzeuge einschließt. Die Elemente einer dienstorientierten Architektur im IT-Management sind wiederverwendbare und an standardisierten Betreiberprozessen ausgerichtete Managementdienste."*

Auf dieser Definition aufbauend werden im Folgenden die wesentlichen Konzepte für die Ausgestaltung einer dienstorientierten Architektur zur Integration von Managementwerkzeugen

erläutert. Anschließend wird mit dem Aspekt der Wiederverwendbarkeit eine Eigenschaft von Diensten behandelt, die im Fokus der vorliegenden Arbeit einen zentralen Stellenwert einnimmt.

Der in Abschnitt 2.1 eingeführte Begriff IT-Dienst wird für das Konzept der dienstorientierten Architektur aufgegriffen und weiter verfeinert. Während bei der abstrahierten Auffassung des Begriffes IT-Dienst die generische Ausgestaltung der logischen Dienstkomponente bzw. des betrieblichen Aspektes im Vordergrund steht, werden bei der Adaption des Begriffes für das Konzept der dienstorientierten Architektur einige Vorgaben getroffen, die im Wesentlichen als Erkenntnis der letzten Jahre in der Forschung auf diesem Gebiet angesehen werden können.

2.2.1 Prinzipieller Aufbau der Architektur

Mit der Gestaltung einer dienstorientierten Architektur wird die klassische Anwendungsarchitektur – bestehend aus drei logischen Schichten – erweitert und fortgeführt. Hierbei wird die Ebene der Geschäftslogik in zwei Teile untergliedert und besteht aus der Schicht mit den Basisdiensten und der Schicht mit den aus der Verschaltung der Basisdienste hervorgegangenen Dienstkompositionen [EL+06]. Eine Auftrennung dieser beiden Teilebenen ist vor dem Hintergrund unterschiedlicher Anforderungen an die Implementierung der jeweiligen Elemente sinnvoll.

In Abbildung 3 wird dieser Sachverhalt verdeutlicht. Ersichtlich sind die einzelnen Ebenen der klassischen Drei-Schichten-Architektur (Präsentation, Geschäftslogik, Datenhaltung) sowie die Trennung der Geschäftslogikebene in die Teilschichten Prozess- und Kompositionsschicht sowie Basisdienste.

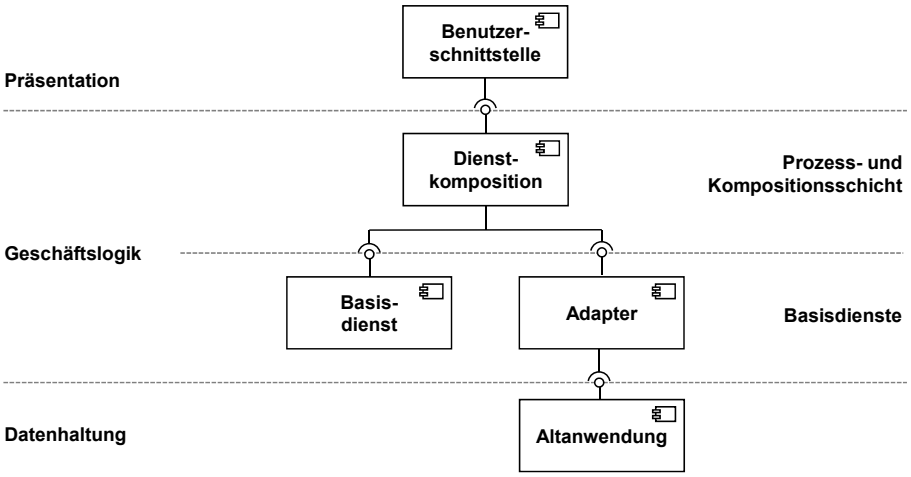

Abbildung 3 Prinzipielle Strukturierung einer dienstorientierten Architektur

Die Trennung in komponierte Dienste und Basisdienste ist vor allem notwendig, da die beiden Teilaspekte durch unterschiedliche technologische Ansätze umgesetzt werden. Diese ergänzen sich auf Ebene der Implementierung (z. B. durch den Einsatz von XML zur Beschreibung von Schnittstellen). Teilweise sind durch diese beiden Teilaspekte aber auch gegenläufige Konzepte hinsichtlich des Softwareentwicklungsprozesses zu betrachten [EL+06].

Dienstmodell

In Abbildung 4 ist ein Modell für den Aufbau von abstrakten, also von konkreten Implementierungen unabhängigen Diensten einer dienstorientierten Architektur dargestellt [EK+07]. Auf Basis dieses Modells können alle wesentlichen Aspekte von Diensten einer dienstorientierten Architektur beschrieben werden. Letztlich stellt dies auch den Ausgangspunkt dar, um die beschriebenen Managementdienste zur Integration von bestehenden Managementwerkzeugen richtig einordnen zu können.

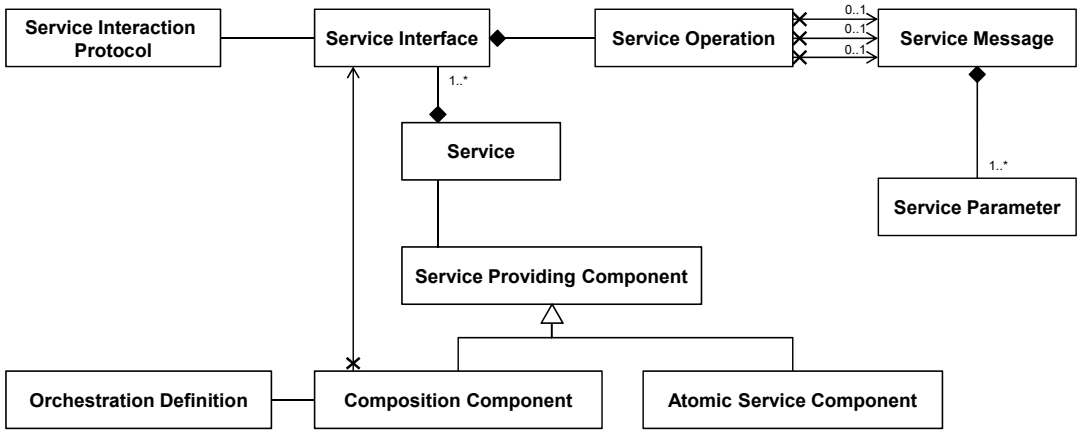

Abbildung 4 Abstraktes Dienstmodell nach [EK+07]

Wie dargestellt besteht ein Dienst aus mehreren diensterbringenden Komponenten (engl. *Service Providing Component*), die in einer kooperativen Arbeitsteilung die an der Dienstschnittstelle (engl. *Service Interface*) spezifizierte Dienstfunktionalität erbringt. Grundlage zur Nutzung einer Dienstoperation ist ein einzelne Nachricht (engl. *Service Message*) an den Dienst, die die durch die genutzte Operation (engl. *Service Operation*) erforderlichen Parameter (engl. *Service Parameter*) enthalten. Ein eindeutiges Interaktionsprotokoll (engl. *Service Interaction Protocol*) legt fest, in welcher Reihenfolge welche Nachrichten an den Dienst gesendet bzw. im Falle einer asynchronen Nachrichtenkommunikation von diesem empfangen werden. Die diensterbringende Komponente kann in eine zusammengesetzte Komponente (engl. *Composition Component*) oder eine atomare Komponente (engl. *Atomic Service Component*) unterteilt werden. Eine dedizierte Definition der Orchestrierung (engl. *Orchestration Definition*) legt die Komposition von zusammengesetzten Diensten fest.

2.2.2 Wiederverwendbarkeit von Diensten

Mit dem Ansatz der dienstorientierten Architektur wird vor allem die Hoffnung verknüpft, bestehende Anwendungen (engl. *Legacy Applications*, Altsysteme, Altanwendungen) in Erweiterungs- oder Integrationsprojekten weiter zu nutzen, diese jedoch mit Schnittstellen zu versehen, die an den zu unterstützenden Geschäftsprozessen ausgerichtet sind. Idealerweise kann somit der Prozessgedanke einfacher umgesetzt werden, da die entwickelten Dienstschnittstellen aus geschäftlichen

Anforderungen motiviert werden. **Wiederverwendung** von bestehenden Softwareartefakten demnach ist ein zentrales Element beim Entwurf komplexer Systeme.

Wiederverwendbarkeit bezeichnet die Eigenschaft eines Informationsartefaktes, die sich auf die Fähigkeit bezieht, wiederverwendet werden zu können [Po97]. Entscheidend für die Beurteilung ist, dass das betroffene Informationsartefakt in einem anderen als ursprünglich vorgesehenen konkreten Anwendungsbezug wiederverwendet werden kann. Erl bezeichnet dies als *Multi Purpose Reuse* (in etwa: Mehrzweckwiederverwendung) [Er08], wodurch vor allem darauf hingewiesen werden soll, dass im Kern der Betrachtungen der Wiederverwendbarkeit von Diensten die Fähigkeit steht, in einem anderen als dem ursprünglich vorgesehenen Szenario eingesetzt zu werden [WY+08].

Während die Begrifflichkeit Wiederverwendung zunächst dem Bereich der objektorientierten Softwareentwicklung (OOSE) entstammt und vom eigentlichen Fokus her die Wiederverwendung innerhalb einer Organisation bei verschiedenen Anwendungsbezügen bezeichnet, wird für die vorliegende Arbeit der Begriff auf den dienstorientierten Entwurf von Anwendungssoftware erweitert. Von Interesse ist, welche bekannten Aspekte der Wiederverwendung aus dem Bereich der OOSE genutzt werden können, bzw., welche Konzepte angepasst oder erweitert werden müssen, um für den Entwurf dienstorientierter Systeme genutzt werden zu können. Da unterschiedlichste Informationsartefakte wiederverwendet werden können (Dokumentation, Entwurfsmodelle, ausführbarer Code) und damit auch unterschiedliche Vorgehensweisen für die Erschaffung wiederverwendbarer Artefakte verknüpft sind [FV99], bezieht sich der Begriff Wiederverwendung für den Rahmen dieser Arbeit auf den Entwurf wiederverwendbarer Managementdienste. Konkret bedeutet dies, dass die relevanten Artefakte im Entwurfsprozess (Dienstkandidaten, Dienstschnittstellen und logische Dienstkomponenten) betrachtet und analysiert werden.

Vorteile durch Wiederverwendung

Durch die Wiederverwendung von Informationsartefakten wird vor allem die Hoffnung verbunden, in einem Softwareentwicklungsprozess verschiedene Vorteile zu erzielen:

- Steigerung der **Produktivität**: Durch die Tatsache, dass bestehende Artefakte wiederverwendet werden, müssen eventuell weniger Artefakte neu konstruiert werden. Vor allem in Bezug auf zu implementierende Softwarekomponenten kann dies zu einer Steigerung der Produktivität in einem Entwicklungsprozess führen, da die Umsetzung von Entwurfsartefakten in ausführbarem Code immer noch mitunter einen der größten Anteile am Entwicklungsprozess einnimmt, wenngleich Ansätze wie beispielsweise die modellgetriebene Softwareentwicklung (engl. *Model Driven Software Development*, MDSD) versuchen, dem entgegenzuwirken. Durch die Einführung eines verbindlichen Referenzmodells für einen abgeschlossenen Problembereich (z. B. dem in der vorliegenden Arbeit betrachteten Bereich IT-Management) können bestimmte Aspekte durch dieses Referenzmodell festgelegt werden. Die Wiederverwendung dieses Referenzmodells in beliebigen Entwicklungsprozessen kann den Entwurf dann dahingehend unterstützen, dass bestimmte Sachverhalte in Form von Mustern direkt genutzt werden können.

- Steigerung der **Zuverlässigkeit** des entwickelten Softwaresystems: Wird für die Umsetzung eines Softwaresystems auf getestete und als zuverlässig bezüglich gegebenen Anforderungen beurteilte Softwareartefakte zurückgegriffen, kann die Zuverlässigkeit des neuen Gesamtsystems einfacher

beurteilt werden, da für bestehende Komponenten keine neuen Tests mehr durchgeführt werden müssen. Die Umsetzung dieses Aspektes findet sich z. B. in der Nutzung von Testfällen und automatisierenden testgebenden Verfahren wieder. Dies entspricht im Wesentlichen dem Teile-und-Herrsche-Prinzip.

- Konzentration auf die **Verschaltung** bestehender Komponenten: Sind bestehende Komponenten dahingehend abstrakt ausgelegt, dass in der Schnittstellenbeschreibung von konkreten eingesetzten Technologien abstrahiert wird, kann die Umsetzung zusätzlicher fachlicher Anforderungen in einem bestehenden System durch Verschaltung vorhandener Basisfunktionalität erreicht werden. Dieser Aspekt wird vor allem durch den Einsatz standardisierter Beschreibungsansätze für die Definition verbindlicher Schnittstellen bei Webservices erreicht (Einsatz von XML zur Festlegung der Schnittstellensprache *Web Service Description Language*, WSDL [W3C-WSDL]), sowie Einsatz standardisierter Beschreibungsansätze zur Definition der Verschaltungslogik von bestehenden Webservices (Einsatz von XML zur Festlegung der Verschaltungssprache *Business Process Execution Language*, BPEL [OASIS-WSBPEL]). Der Einsatz von Webservices begünstigt die Wiederverwendung daher prinzipiell.

- Zwang zur **Modularisierung**: Wird bei einem Entwurf von vornherein auf die Anforderung geachtet, entstandene Artefakte wiederzuverwenden, impliziert dies in der Regel, dass die einzelnen Elemente in abgeschlossene Module gekapselt wurden. Die Zerlegung eines komplexen Gesamtproblems in einzelne, abgeschlossene und voneinander unabhängige Teilprobleme ist ein in der Softwareentwicklung gängiges Muster, um bestehende Ansätze einfließen lassen zu können. Die Auftrennung von Diensten in atomare Basisdienste und komponierte Dienste fördert die Modularisierung.

- Größere **Adaptierbarkeit**: Letztlich als Erweiterung der Vorteile der Modularisierung und Verschaltung führt der Entwurf von Softwaresystemen auf Basis bestehender wiederverwendbarer Artefakte zu einer besseren Anpassungsfähigkeit eines Systems. Diese Anforderung ist vor allem dahingehend gerichtet, bestehende Systeme an zusätzlich aufkommenden fachlichen Anforderungen ausrichten zu können, um beispielsweise sich ändernde geschäftliche Rahmenbedingungen angleichen zu können.

Grundlegend kann zwischen zwei verschiedenen Formen der Wiederverwendung unterschieden werden [Po97]: **opportunistische Wiederverwendung** (geschieht ungeplant durch die Anpassung bestehender Quellcodes, um neue Anforderungen umzusetzen) sowie **systematische Wiederverwendung**. Bei der opportunistischen Wiederverwendung wird vor allem die nicht modifizierte Wiederverwendung bestehender Implementierungsartefakte betrachtet, wobei zusätzliche Anforderungen durch Verschaltung bestehender Elemente erreicht werden, sowie durch Übergabe von Parametern, bzw. die Nutzung von Funktionen moderner objektorientierter Softwarecompiler wie beispielsweise Vererbung und Polymorphismus. Systematische Wiederverwendung setzt voraus, dass ein systematischer Entwicklungsprozess ausgeführt wird, wobei idealerweise bestehende Komponenten bereits auf eine mögliche Wiederverwendung hin ausgerichtet wurden. Die im Weiteren betrachteten Aspekte der Wiederverwendung lassen sich daher in die Klasse der systematischen Wiederverwendung einordnen.

Da sich die Wiederverwendbarkeit immer auf zukünftige Entscheidungsfindungen bezieht, ist ersichtlich, dass hierdurch eigentlich ein Wahrscheinlichkeitsmaß definiert wird. Zwar wird gewünscht, ein möglichst hohes Maß an Wiederverwendbarkeit zu erreichen. Bezogen auf den Entwurf von Managementdiensten definiert diese Angabe jedoch einen Erwartungswert, da durch die Wiederverwendung von Managementfunktionen in Form von Diensten ja gerade zukünftige Erweiterungen einer Managementlösungen einfacher umgesetzt werden sollen. Die Auslegung eines Entwicklungsvorgehens auf Wiederverwendung stellt demnach zunächst eine zusätzliche Anstrengung dar.

Zur genauen Definition des Begriffes Wiederverwendung wird für die vorliegende Arbeit zunächst auf die Definition von Wiederverwendbarkeit von Poulin zurückgegriffen [Po97]:

> *„(Software-)Wiederverwendung bezeichnet die Nutzung von bestehenden Quellen oder Softwarekomponenten, um eine neues Softwareprogramm oder eine neue Anwendung zu entwickeln.“*

Ausgehend von der Definition des Begriffes für den Rahmen dieser Arbeit werden unterschiedliche Aspekte der Wiederverwendung in Abschnitt 4.3 beschrieben. Hierbei wird eine Erweiterung bestehender Konzepte um Ansätze der dienstorientierten Architektur diskutiert.

Aspekte der Wiederverwendbarkeit

Um eine Methode für den Entwurf von Softwaresystemen bezüglich der Anforderung der Wiederverwendung optimal gestalten zu können, können nach Krueger [Kr92], Prieto-Diaz [Pr93], Erl [Er08] und Poulin [Po07] verschiedene grundlegende Aspekte bezüglich des Einflusses auf die Wiederverwendbarkeit unterschieden werden. Diese sind: Abstraktion, Auswahl, Spezialisierung, Integration (alle [Kr92]), Agnostizität, generische Logik, generischer Vertrag, Nebenläufigkeit (alle [Er08]), Kohäsion, Autonomie, Nützlichkeit, Komplexität (alle [Po07]). Ergänzt werden diese Aspekte durch weitere fachlich-motivierte Aspekte (Vollständigkeit, Disjunktheit, Namenskonventionen). Prieto-Diaz betrachtet vor allem unterschiedliche Ausprägungen von Vorgehen für den Entwurf wiederverwendbarer Softwarekomponenten.

In nachfolgender Tabelle werden diese verschiedenen Aspekte in Bezug auf die Bedeutung für die Elemente eines Entwurfes von Managementdiensten beschrieben und konkretisiert.

Aspekt	**Bedeutung für den Dienstentwurf**
(A1) Abstraktion	- Definition verbindlicher und fachlich-motivierter Dienstschnittstellen - Spezifikation von Semantik für Daten und Funktionen - Unabhängigkeit von konkreten Implementierungsdetails
(A2) Auswahl	- Klassifikation und Katalogisierung

	- Einheitliche Syntax - Unterstützung durch Entwicklungsumgebungen und technischer Infrastruktur (Dienstverzeichnis) erforderlich
(A3) Spezialisierung	- Trennung von Schnittstelle und Implementierung - Generische Dienstaufrufe
(A4) Integration	- Offene Schnittstellen - Unterstützung durch Ausführungsumgebungen erforderlich
(A5) Agnostizität	- funktionaler Kontext der Schnittstelle ist unabhängig von bestimmten Nutzungsszenarien
(A6) Generische Logik	- Die durch einen Dienst bereitgestellten Operationen sind generisch und abstrahieren von der zugrunde liegenden Logik
(A7) Generischer Vertrag	- Es können verschiedene unterschiedliche Ein- und Ausgangsnachrichten verarbeitet werden
(A8) Nebenläufigkeit	- Die einem Dienst zugrunde liegende Logik kann simultan genutzt werden
(A9) Kohäsion	- Wohldefinierte Schnittstellen zur Abdeckung eines funktionalen Kontextes
(A10) Autonomie	- Keine weiteren Dienste benötigt - Keine Vorgaben für die Abfolge von Dienstaufrufen
(A11) Nützlichkeit	- Bezug zu komplexen Konzepten aus der betrachteten Domäne
(A12) Komplexität	- Anzahl an Operationen in einer spezifizierten Dienstschnittstelle
(A13) Vollständigkeit	- Dienstschnittstellen werden vollständig spezifiziert und gegebenenfalls bezüglich bestimmter Muster ergänzt
(A14) Disjunkheit	- Basisdienste, die auf Managemententitäten operieren, haben eine vollständige Hoheit über die Entität

(A15) Namenskonvention	- Einhaltung bestimmter Konventionen zur impliziten Festlegung von Semantik (z. B. `CreateIncidentRecord` als Operationsname zum Anlegen eines Störungsbelegs) - Kann durch Modelltransformationen unterstützt werden

Tabelle 1 Aspekte beim Entwurf wiederverwendbarer Managementdienste

Im Kern der Betrachtung der Wiederverwendbarkeit von Managementdiensten steht die Dienstschnittstelle, da die Schnittstelle den Zugriffspunkt auf die vom Dienst bereitgestellte Implementierung bietet. Somit rückt das SoaML-Modellelement `ServiceInterface` [OMG-SoaML] in den Vordergrund, da hierdurch in einem Dienstentwurf eine konkrete Schnittstelle für Managementdienste dargestellt wird. Ergänzend werden alle weiteren relevanten Modellelemente eines SoaML-basierten Entwurfes betrachtet und bei der Untersuchung eines jeweiligen Aspektes in Beziehung zu diesem gestellt. Es ist ersichtlich, dass nicht alle Aspekte eine Relevanz für alle Modellelemente besitzen.

(A1) Abstraktion

Krueger zufolge ist Abstraktion der grundlegende Aspekt beim Entwurf wiederverwendbarer Softwareartefakte [Kr92]. Bei der Abstraktion werden konkrete Details möglicher verschiedener Lösungen ausgeblendet, um die für die umzusetzenden Anforderungen essenziellen Aspekte herauszustellen. Eine Abstraktion hat zum Ziel, eine konkrete Realisierung durch die Festlegung einer Spezifikation zu erschaffen. Die Spezifikation kann, bezogen auf den Softwareentwurf, als Schnittstelle, die Realisierung als Implementierung aufgefasst werden. Neben der Festlegung auf eine Syntax in einer Spezifikation muss insbesondere auch die Semantik der durch die Spezifikation eingeführten Elemente klar definiert werden. Dies stellt einen wichtigen Aspekt dar, um insbesondere den Entwurf von Managementdiensten hinsichtlich fachlicher Anforderungen auf den Grad an möglicher Wiederverwendung hin zu evaluieren. Der Einsatz einer Methode, die auf Basis wesentlicher Abstraktionen der zu unterstützenden Domäne verschiedene fachlich-motivierte Modelle konstruiert, begünstigt die Wiederverwendung daher maßgeblich.

Durch die Zielsetzung, dienstorientierte Schnittstellen zu bestehenden Managementwerkzeugen zu konstruieren, wird bereits ein gewisses Maß an Abstraktion – implizit mit der Ausrichtung an den Prinzipien der dienstorientierten Architektur – eingeführt. Im Entwurf von Managementdiensten bezieht sich der Aspekt der Abstraktion daher vor allem auf die Spezifikation von Dienstschnittstellen, damit aber letztlich auch auf die eines Dienstentwurfes zugrunde liegenden erforderlichen Dienstkandidaten. Mit dem Konzept, Schnittstellenbeschreibung von realisierender Implementierung zu trennen, entsteht zwangsläufig die Anforderung, gewisse Details der Lösung zu abstrahieren. Fokussiert zu betrachten ist daher neben der Umsetzung dieses Aspektes in Form des dienstorientierten Ansatzes daher vor allem die Frage, inwiefern die bei einem Managementdienst identifizierten Dienstoperationen von technischen Details abstrahieren und eine klaren Bezug zu den jeweiligen Konzepten der betrachteten Domäne aufweisen.

(A2) Auswahl

Um ein Softwareartefakt wiederzuverwenden, ist es erforderlich, dass dieses aufgefunden [PF87], verstanden und verglichen werden kann [WY+03]. Durch die Einhaltung festgelegter Normen können deswegen verschiedene Abstraktionen von verschiedenen Artefakten gleichermaßen zielführend genutzt werden. Die Anforderung der Vergleichbarkeit setzt voraus, dass gewisse Merkmale von ähnlichen Diensten als solche wahrgenommen bzw. im Umkehrschluss, dass gewisse Merkmale von verschiedenartigen Diensten als solche erkannt werden können. Diese Merkmale beziehen sich auf verschiedene Kategorien, die sich letztlich in ein Klassifikationsschema eingliedern lassen.

Das Prinzip der Auswahl zielt auf die Nutzung bereits bestehender Softwarteartefakte zur Wiederverwendung ab. Dieser Aspekt setzt demnach voraus, dass eine Unterstützung durch Entwicklungsumgebungen vorliegt. Ein wesentliches Kriterium in dienstorientierten Architekturen stellt die Existenz eines Kataloges dar, der die innerhalb einer Organisation verfügbaren Dienste umfasst. Dieses als Dienstverzeichnis bekannte strukturelle Element einer dienstorientierten Architektur ermöglicht das Auffinden von benötigten Diensten anhand deren beschriebener Fähigkeiten. Typischerweise werden Dienstverzeichnisse auf Basis der Standarddefinition *Universal Description, Discovery and Integration* (UDDI) [OASIS-UDDI] realisiert.

Um eine gezielte Auswahl zwischen verschiedenen verfügbaren Managementdiensten in einem Dienstverzeichnis treffen zu können, ist die Existenz einer eindeutigen Klassifikation erforderlich. Der in 4.2.3 vorgestellte Ansatz, die durch die Ableitung aus Domänenmodellen entstandenen Dienstkandidaten direkt in den entsprechenden SoaML-basierten Modellen semantisch zu erweitern, unterstützt diesen Aspekt wesentlich. Da für die Erfassung fachlicher Bezüge aus dem Domänenmodell Kataloge und hierzu gehörende Kategorieelemente eingesetzt werden, kann eine Klassifikation als Menge von disjunkten Kategorieelementen aufgefasst werden.

(A3) Spezialisierung

Spezialisierung ist ein Teilaspekt von Abstraktion. Durch die Aufteilung der Abstraktion in eine Spezifikation und eine Realisierung wird eine Möglichkeit eingeführt, durch Variabilität der Spezifikation eine Realisierung in unterschiedlichen Anforderungen einzusetzen. Beispielsweise ermöglicht die Definition eines generischen Dienstes für die Manipulation beliebiger Managemententitäten mit *Create*-, *Read*- und *Update*-Operationen eine weitestgehende Variabilität der Spezifikation, da es die Spezifikation einer Abstraktion durch einen variablen Teil erweitert.

Damit eine variable Realisierung ermöglicht wird, ist die Möglichkeit zur Spezialisierung bereits während der Spezifikation einzubeziehen. So kann beispielsweise ein in einem Unternehmen zentral definierter Datenhaltungsdienst genutzt werden, um durch spezielle Fassaden eine Spezialisierung hin zu verschiedenen fokussierten fachlichen Bereichen zu ermöglichen. Der Einsatz von Konfigurationsparametern ermöglicht weiterhin die gezielte Steuerung unterschiedlichen Verhaltens der implementierten Logik eines Dienstes.

(A4) Integration

Damit bestehende Softwareartefakte im Rahmen eines auf Wiederverwendung ausgelegten Entwicklungsprozesses eingesetzt werden können, ist neben der Möglichkeit, einen semantischen Bezug zwischen benötigter und tatsächlich vorhandener Funktionalität auch eine technische Grundlage zu schaffen, um die bestehenden Softwareartefakte nutzen zu können. Diese beiden grundlegend verschiedenen Ausrichtungen des als Integration bezeichneten Aspektes in [Kr92] verdeutlichen, dass neben dem fachlichen Aspekt (semantischer Bezug über die Ausrichtung an fachlichen Konzepten) vor allem auch der technische Aspekt beim auf Wiederverwendung ausgelegten Entwurfsvorgehen zu beachten ist.

Da in verteilten Systemen die Kommunikation der verschiedenen aktiven Komponenten dieses Systems über den Austausch von Nachrichten über ein Kommunikationsnetz erfolgt, ist die Einhaltung verbindlicher Kommunikationsprotokolle erforderlich. In dienstorientierten Architekturen, die auf Webservices basieren, hat sich in der Vergangenheit hier vor allem der Nachrichtenaustausch durch die Definition einheitlicher Nachrichtenformate mit XML sowie die Kommunikation über das zustandslose *Hypertext Transfer Protocol* (HTTP) durchgesetzt. Die Schnittstelle von Webservices wird dabei in der Regel durch die Definition einer WSDL-Datei (engl. *Web Service Description Language*) definiert. Die Einhaltung dieser Standards stellt ein Mindestmaß an Integrationsfähigkeit auf der technischen Ebene sicher, wodurch insbesondere die Betrachtung der semantischen Aspekte in den Vordergrund rückt.

(A5) Agnostizität

Damit ein entworfenes Softwareartefakt in einem anderen Kontext als dem ursprünglich vorgesehenen eingesetzt werden kann, ist es erforderlich, dass der Entwurf hinsichtlich einer möglichst großen Kontextunabhängigkeit durchgeführt wird [Er08]. Bezogen auf den Entwurf wiederverwendbarer Managementdienste betrifft sich dieser Aspekt insbeondere die Unabhängigkeit der einzelnen Operationen eines entworfenen Managementdienstes hinsichtlich eines konkreten Betreiberprozesses. Da ein wesentliches Ziel durch die Einführung einer dienstorientierten Architektur vor allem die automatisierte Ausführung von Betreiberprozessen ist, muss bei der Betrachtung der Kontextunabhängigkeit zwischen jenen Diensten unterschieden werden, die aufgrund ihres klaren Prozessbezuges eine geringe Unabhängigkeit von einem konkreten Kontext aufweisen, sowie jenen Dienste, die generische und von konkreten Prozessen unabhängige Funktionalität anbieten.

(A6) Generische Logik

Generische Dienste ermöglichen die Nutzung der implementierenden Dienstlogik in weiteren Szenarien als den ursprünglich geplanten. Im weiteren Sinne ist dieser Aspekt ähnlich dem der Spezialisierung, wobei beim Aspekt der Spezialisierung die Wiederverwendung durch die Möglichkeit der externen Konfigurierung ermöglicht wird. Generische Dienstlogik hingegen bezieht sich auf die Fähigkeit, den einen Dienst implementierenden Programmcode zu nutzen, um verschiedene Funktionen auszuführen.

Generische Dienstlogik ist wünschenswert, da sie die Wahrscheinlichkeit vergrößert, wiederverwendet werden zu können. Dem steht gegenüber, dass durch eine generische Dienstlogik die direkte

Verwendung eingeschränkt wird, da zusätzliche Anpassungen vorgenommen werden müssen, um weiteren Aspekten (Auswahl, Integration, Kohäsion, Nützlichkeit oder Vollständigkeit) zu genügen. Insgesamt ist daher ein Bezug zu fachlichen Konzepten mit der Erfüllung der Aspekte der Vollständigkeit und Disjunktheit einer generischen Dienstlogik vorzuziehen.

(A7) Generischer Vertrag

Neben der strukturellen Betrachtung der Operationen einer Dienstschnittstelle sind gewisse Einschränkungen bei der Nutzung eines Dienstes zu beachten. Diese werden im Vertrag zu einem Dienst festgelegt, der ein Protokoll zur Nutzung dieses Dienstes definiert. Hierzu gehört vor allem der Wunsch, unabhängig von konkreten Kommunikationsprotokollen einen Dienst zu nutzen sowie die Vermeidung von synchronen Dienstaufrufen.

(A8) Nebenläufigkeit

Die Kommunikation in einer dienstorientierten Architektur erfolgt prinzipiell nachrichtenbasiert (siehe auch Abschnitt 2.2.1 hierzu). In der Implementierung einer dienstorientierten Architektur stehen hierzu unterschiedliche Protokolle und Umsetzungen bereit. Mit der Orientierung am Prinzip der nachrichtenbasierten Kommunikation kann eine direkte Entkopplung von aufrufendem Dienstnehmer und bereitstellendem Dienstgeber erzielt werden. Insgesamt ist die Entscheidung, Nebenläufigkeit zu unterstützen, jedoch eine technische Betrachtung, da letztlich Nebenläufigkeit (also die Fähigkeit, Nachrichten von verschiedenen Dienstnutzern zum Empfang gleichzeitig entgegenzunehmen) durch eine konkrete Ausführungsumgebung unterstützt werden muss. Von der Perspektive des Entwurfes abstrakter Dienstschnittstellen stellt dieser Aspekt daher ein Implementierungsdetail dar, weshalb er in den weiteren Betrachtungen der Wiederverwendbarkeit entworfener Managementdienste keine weitere Vertiefung findet.

(A9) Kohäsion

Die Gruppierung logisch zusammenhängender Funktionalität zu einer gebündelten Einheit ist eines der wesentlichen Grundprinzipien eines strukturierten Entwicklungsvorgehens. Je stärker verschiedene Artefakte eines Entwurfes diese Eigenschaft aufweisen, desto einfacher kann eine klare Zuordnung dieser verschiedenen Artefakte zum jeweiligen zusammenhaltenden Aspekt festgestellt werden. Hieraus resultieren verschiedene Vorteile (beispielsweise: Erleichterung der Auffindbarkeit), letztlich steht dieser Aspekt in direktem Zusammenhang zum Aspekt der Vollständigkeit und Disjunktheit.

(A10) Autonomie

Wiederverwendung eines Artefaktes wird vereinfacht, wenn wenige Abhängigkeiten zu weiteren (externen) Artefakten bestehen. Inhärent verbunden mit der Ausrichtung an den Prinzipien der dienstorientierten Architektur ist die Erwartung, dass einmal entworfene Dienste über die Veröffentlichung der Dienstschnittstelle mehrfach genutzt und durch Komposition zu zusammengesetzten Diensten verbunden werden können. Idealerweise können zusammengesetzte Dienste auf Basis grundlegender und elementarer Dienste realisiert werden.

Damit die komplexen Dienste wiederverwendet werden können, ist neben der Möglichkeit zur Wiederverwendung dieser Dienste aber auch inhärent die Anforderung verbunden, die einen zusammengesetzten Dienst realisierenden Basisdienste wiederzuverwenden. Eine große Autonomie – im Sinne von geringer Anzahl an Abhängigkeit zu weiteren Diensten – verkleinert somit insgesamt den Aufwand, wiederverwendet zu werden.

(A11) Nützlichkeit

Damit ein Softwareartefakt wiederverwendet werden kann, ist es erforderlich, dass das Artefakt eine gewisse Vielseitigkeit oder Nützlichkeit aufweist. Hierbei ist vor allem der Bezug zu den verschiedenen Konzepten der jeweils konkret zu betrachtenden Domäne ersichtlich zu machen. Auf der fachlichen Ebene weist dieses Konzept daher starke Ähnlichkeit mit dem Aspekt der Integration auf, kann aber sehr viel schwieriger durch eine formale Betrachtung erfasst werden als die weiteren Aspekte. Neben der Integration ist beim Aspekt der Nützlichkeit vor allem die Möglichkeit zu untersuchen, inwiefern ein entworfener Managementdienst überhaupt die Eignung aufweist, vielfach eingesetzt zu werden, ohne weitere Anpassung vornehmen zu müssen. Fokussiert man diesen Aspekt, wird durch die Nützlichkeit die eigentliche Verwendbarkeit betrachtet.

(A12) Komplexität

Poulin zufolge ist die Komplexität – gemessen in Zeilen von Quellcode – ein wichtiges Kriterium bei der Beurteilung der Wiederverwendbarkeit erstellter Softwareartefakte [Po97]. Komplexe Artefakte – im Sinne von aus vielen Zeilen Quellcode bestehend – erschweren die Wiederverwendung, da ein vergrößerter Zeitaufwand für das Verständnis der Funktion eines wiederzuverwendenden Artefaktes erforderlich ist. Bezogen auf die Untersuchung der Wiederverwendbarkeit von Managementdiensten wird mit diesem Aspekt vor allem die einen Managementdienst definierende Dienstschnittstelle betrachtet. Damit ein konkreter Managementdienst wiederverwendet werden kann, ist ein Verständnis über die an der Schnittstelle angebotenen Dienstoperationen sowie die dazu gehörenden Signaturen erforderlich. Der Einsatz semantischer Annotation wie beispielsweise SAWSDL (*Semantic Annotations for WSDL and XML Schema* [W3C-SAWSDL]*)* unterstützt das Verständnis der Semantik definierter Dienstoperationen bei konkreten Dienstschnittstellen wesentlich; allerdings ist hierfür zum einen die Annotation bereits im Entwicklungsvorgehen zu berücksichtigen und zum anderen die Unterstützung durch geeignete Entwicklungswerkzeuge erforderlich. Ziel beim Entwurf wiederverwendbarer Managementdienste ist es daher, die Anzahl an Dienstoperationen an einer Dienstschnittstelle gering zu halten, wobei dieser Aspekt somit direkte Auswirkung auf die Aspekte Vollständigkeit, Disjunktheit und Agnostizität aufweist.

(A13) Vollständigkeit

Damit zukünftige Anforderungen zielgerichtet umgesetzt werden können, ist eventuell die Änderung bestehender Komponenten durch Hinzufügen zusätzlicher Funktionalität erforderlich. Wiederverwendung von Softwareartefakten impliziert demnach, dass solche Änderungen durchgeführt werden können. Da durch die Änderung bestehender Komponenten wiederum die Überarbeitung bezüglich der betrachteten Aspekte erforderlich werden kann, kann die Vorwegnahme bestimmter zukünftiger Erweiterungen dem entgegenwirken. Somit wird nicht nur einer wiederkehrenden

Überarbeitung aller zu betrachtenden Aspekte begegnet, sondern darüber hinausgehend wird auch die Auffindbarkeit vergrößert, da zukünftige Anforderungen direkt mit bereits bestehenden Softwareartefakten realisiert werden können.

Dieser Aspekt steht in direktem Zusammenhang zu den Aspekten Kohäsion und Disjunktheit. Ausgehend von der Annahme, identifizierte Komponenten zur Umsetzung von bestimmten Anforderungen werden durch nachträgliche Erweiterungen ergänzt, kann dies zu Überlappungen zwischen funktionalen Bereichen einzelner Komponenten eines Gesamtsystems führen (Disjunktheit), wodurch z. B. die Wartbarkeit stark beinträchtigt wird. Gleichwohl ist durch nicht vollständig und nachträglich erweiterte Komponenten der logische Zusammenhalt abgeschlossener Elemente beeinträchtigt. Insgesamt nimmt der Aspekt der Vollständigkeit daher einen zentralen Stellenwert bei der Beurteilung eines vorliegenden Entwurfes von Managementdiensten ein.

Eine Abschätzung aller zukünftigen Erweiterungen ist nur in begrenztem Rahmen, z. B. durch den Einsatz von Referenzmodellen in der Entwurfsphase, möglich. Es können jedoch einige grundlegende Muster im Entwurfsprozess Beachtung finden, die einen einfachen Kompromiss bezüglich des Umfangs von zusätzlichem Entwurfes- und Implementierungsaufwandes darstellen. Hierzu zählt vor allem die Beachtung verschiedener fachlich-motivierter Muster von angebotenen Operationen für datenzentrierte Systemkomponenten (beispielsweise Datenbankzugriff) oder die Ergänzung von inversen Operationen von kommunikationszentrierten Komponenten.

Spezielle fachliche Abwägungen können jedoch dazu führen, dass dieser Aspekt teilweise aufgeweicht wird. So finden sich in den vorgestellten datenzentrierten Managementdiensten die Definition von *Create*-, *Read*- und *Update*-Operationen bei als Vollständig zu bewertenden Dienstschnittstellen. Hier wird bewusst die beim CRUD (engl. *Create, Read, Update, Delete*) bekannte *Delete*-Operation ausgelassen, da diese nach der Standardspezifikation ISO/IEC20000-1:2005 [ISO05] explizit nicht vorgesehen wird.

(A14) Disjunkheit

Damit einmal entworfene Softwareartefakte in erweiterten Szenarien wiederverwendet werden können, ist es erforderlich, dass keine Seiteneffekte durch zusätzliche Abhängigkeiten entstehen. Bezogen auf den Entwurf wiederverwendbarer Managementdienste ist ein wesentliches Ziel daher die Reduzierung von Überschneidungen in den funktionalen Bereichen einzelner Managementdienste.

Dieser als Disjunktheit bezeichnete Aspekt bezieht sich auf die Definition von Dienstschnittstellen. Da durch die Integration bestehender Managementwerkzeuge im Kontext des Entwurfes der Dienste einer dienstorientierten Architektur prinzipiell eine Überschneidung zwischen funktionalen Bereichen entstehen kann, ist die Betrachtung dieses Aspektes bereits frühzeitig im Entwicklungsvorgehen einzubeziehen. Einer Überschneidung von – durch bestehende Managementwerkzeuge angebotene auf der einen und durch die Analyse der Anforderungen an zu entwerfende Managementdienste auf der anderen Seite – benötigten Managementfunktionen kann durch eine frühzeitige gemeinsame Betrachtung beider Aspekte im Entwicklungsvorgehen begegnet werden.

(A15) Namenskonventionen

Ein weiterer wichtiger Aspekt, um Wiederverwendung zu ermöglichen, ist die Einhaltung vorgegebener Namenskonventionen. Damit ein einheitliches Begriffsverständnis in einem Entwicklungsprozess etabliert werden kann, ist es erforderlich, dass alle beteiligten Akteure in diesem Entwicklungsprozess über ein gemeinsames Verständnis der betrachteten Domäne verfügen. Durch die Betrachtung der Klassifikation der verschiedenen Artefakte in einem Entwicklungsprozess wird sichergestellt, dass die jeweilige Semantik klar und eindeutig festgelegt wird. Ergänzend kann die Einhaltung von Namenskonventionen, z. B. bei der Benennung gleichartiger Entwurfselemente nach einem einheitlichen Schema, dafür sorgen, dass Namenskonflikte über die Grenzen verschiedener Entwicklerteams hinweg nicht auftreten.

Idealerweise werden, um die Einhaltung von Namenskonventionen sicherzustellen, Modell-zu-Modell-Transformationen definiert. Dadurch kann eine automatisierte Anwendung verschiedener Regeln gewährleistet werden, wodurch letztlich Fehlern, die sich durch manuelle Modellüberführungen im Rahmen der schrittweisen Verfeinerung in einem Entwicklungsprozess ergeben, vorgebeugt werden kann.

2.3 Entwicklung dienstorientierter Architekturen

Die Softwaretechnik (engl. *Software Engineering*) befasst sich mit der professionellen Entwicklung großer Softwaresysteme [RP06]. Eine zielorientierte Bereitstellung und der systematische Einsatz von Prinzipien zur arbeitsteiligen, ingenieursmäßigen Erstellung sind wesentliche Charakteristika der Softwaretechnik [Ba00].

2.3.1 Der Softwareentwicklungsprozess

Der komplette Lebenszyklus eines Softwaresystems teilt sich in verschiedene Phasen auf, die letztlich im Entwicklungsprozess widergespiegelt sind. Der Softwareentwicklungsprozess ist ein kreativer und hochgradig komplexer Konstruktionsprozess, der zum Ziel hat, fachliche Anforderungen in ein lauffähiges Softwareprodukt zu überführen [Ba00]. Es ist ersichtlich, dass vor dem Hintergrund dieser Feststellung nicht ein einziger korrekter Entwicklungsprozess für alle Arten von verschiedenen fachlichen Anforderungen und angestrebter softwaretechnischer Umsetzung existiert, sondern prinzipiell vielmehr jeder Entwicklungsprozess, der die vorgenannte Eigenschaft der Umsetzung erfüllt, zunächst zielführend ist.

Phasen im Entwicklungsprozess

Es lassen sich jedoch verschiedene Charakteristika eines Entwicklungsprozesses identifizieren, die bei einer abstrahierten Betrachtung in allen Prozessen auf unterschiedliche Art und Weise stärker oder schwächer ausgeprägt sind. Hierzu wird vor allem die Untergliederung in verschiedene Phasen des Entwicklungsprozesses gezählt, der sich prinzipiell in die Phasen **Analysephase**, **Entwurfsphase**, **Implementierungsphase** und **Inbetriebnahmephase** aufteilen lässt.

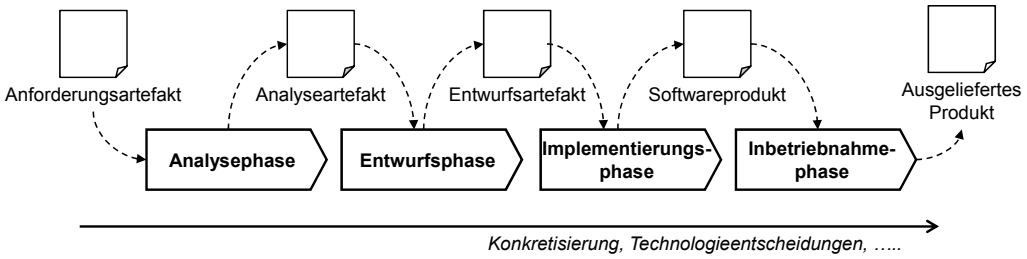

Abbildung 5 Prinzipieller Softwareentwicklungsprozess und relevante Artefakte

Abbildung 5 zeigt den Ablauf dieses prinzipiellen Softwareentwicklungsprozesses und verdeutlicht die wesentlichen, hierzu benötigten und erstellten Artefakte. Die dargestellten Artefakte (Anforderungsartefakt, Analyseartefakt, Entwurfsartefakt, Softwareprodukt, Ausgeliefertes Produkt) stellen logische Artefakte dar und unterteilen sich in der Regel in verschiedene konkrete Teilartefakte. Das **Anforderungsartefakt** ist Ausgangspunkt für eine strukturierte und organisierte Analysephase, in deren Verlauf das **Analyseartefakt** erzeugt wird. Die Erhebung von Anforderungen ist zwar Teil des Softwareentwicklungsprozesses, wird jedoch nicht von der Analysephase durchgeführt, sondern stellt vielmehr einen vorlaufenden Schritt dar, um Artefakte für die Analyse zu erzeugen. Hierdurch entsteht ein weiter Handlungsspielraum, der die spätere Analyse naturgemäß erschwert, da textuelle, semi-formalisierte und formalisierte Anforderungsartefakte zum Einsatz kommen können. Dadurch ist eine klare Semantik der beschreibenden Information oftmals nicht gegeben, weshalb verschiedene Ansätze zum Einsatz kommen können, die die Erfassung von Anforderungen direkt bei den späteren Nutzern ermöglicht und auf deren individuelle Fähigkeiten eingeht. Ziel der Analysephase sollte daher sein, die Spezifikation der vorgegebenen Anforderungen so zu gestalten, dass die Bedeutung der einzelnen Anforderungen ersichtlich bleibt, die Möglichkeit durch Fehlinterpretation jedoch weitestgehend eingeschränkt wird. Erforderlich sind demnach Maßnahmen, um frühzeitig eine auf nachvollziehbaren Formalismen begründete Spezifikation zu erzielen. Das Analyseartefakt wird als erstes Ergebnis des Softwareentwicklungsprozesses daher in der Regel auf Basis formalisierter Modellierungssprachen konstruiert. Die Konkretisierung von fachlichen Anforderungen erfolgt auf Basis des Analyseartefaktes und wird in der Entwurfsphase durchgeführt. Das hieraus resultierende **Entwurfsartefakt** weist bereits diejenigen Eigenschaften der zu konstuierenden Softwarelösung auf, die in der Implementierungsphase in ein fertiges **Softwareprodukt** umgesetzt werden. Die **Auslieferung des Produktes** erfolgt in der Inbetriebnahmephase, in der neben der Installation des Softwareproduktes benötigte Konfigurationen und zusätzliche Anpassungen durchgeführt werden.

2.3.2 Modellierung in der Softwareentwicklung

Der vorige Teilabschnitt zeigt auf, dass der Konstruktionsprozess für ein Softwaresystem verschiedene Teilphasen durchläuft. Die unterschiedlichen Phasen werden von Modellen unterstützt, die zweckgebunden die jeweilige Phase zielführend unterstützen. Ein Modell wird in [Pi00] wie folgt definiert:

> *"A model is a representation of reality intended for some definite purpose." [Pi00]*

Als Abbild der Realität mit der Bindung an einen genauen und bestimmten Zweck ermöglicht ein Modell daher, verschiedene Aussagen über den betrachteten Ausschnitt der Realität zu treffen. Durch die Abstraktion, die ein Modell gegenüber der Realität aufweist, können jedoch gezielt Eigenschaften ausgeblendet werden. Hierdurch entsteht die Möglichkeit, Modelle für einen bestimmten Einsatzzweck zu konstruieren. Durch den Einsatz verschiedener Modelle kann ein Softwareentwicklungsprozess überhaupt zielführend unterstützt werden, damit bereits während des Entwicklungsvorgehens verschiedene Aussagen über das fertige Softwareprodukt getroffen werden können.

Ontologien zur Beschreibung von Semantik

Um eine zielgerichtete Umsetzung von fachlichen Anforderungen auf die Fähigkeiten technischer Ausführungsplattformen entlang des Softwareentwicklungsprozesses zu ermöglichen, ist eine formale Darstellung der Bedeutung des Wissens des zu unterstützenden Fachbereiches notwendig. Diese formale Darstellung, die Beschreibung der Elemente eines Fachbereiches sowie deren strukturelle Beziehungen untereinander werden als Ontologie bezeichnet [CJ+02]. Eine Ontologie beschreibt daher ein gemeinsames Wissen, das bezüglich eines betrachteten Bereiches herrscht.

Der Begriff Ontologie leitet sich aus dem Griechischen ab und bezeichnet durch die Komposition der beiden Wörter *ontos* (für „das Sein“) und *logos* (für „das Wort“) sinngemäß die „Lehre vom Seienden“ [GD+09].

Die Bedeutung innerhalb der Informatik ist festgelegt mit der Zielsetzung, durch...

> *„Ontologien eine formale Spezifikation einer Konzeption zu erreichen“ [Gr93].*

Eine Konzeption bezeichnet die Abstraktion eines darzustellenden Sachverhalts, wobei eine Festlegung auf einen verbindlichen Formalismus erfolgt. Eine detailliertere Definition bezüglich unterschiedlicher Ausprägungen, Formen und Zielrichtungen einer Ontologie findet sich in [OMG-ODM]:

> *“An ontology defines the common terms and concepts (meaning) used to describe and represent an area of knowledge. An ontology can range in expressivity from a Taxonomy (knowledge with minimal hierarchy or a parent/child structure), to a Thesaurus (words and synonyms), to a Conceptual Model (with more complex knowledge), to a Logical Theory (with very rich, complex, consistent, and meaningful knowledge).”*

Wenngleich die Zielsetzung einer Ontologie durch die zweitgenannte verfeinerte Definition gleich bleibt, wird ein Handlungsspielraum bei der Festlegung der Darstellungsform ermöglicht. Prinzipiell kann eine Ontologie sowohl in einem durch Maschinen prozessierbaren Format dargestellt werden (z. B. durch XML), durch die Formulierung einer formalen Logik, durch die Niederschrift in natürlicher Sprache oder aber durch eine grafische Modellierungssprache [GD+09].

Prinzipiell lassen sich vier unterschiedliche Szenarien für den Einsatz einer Ontologie identifizieren. Im Kern steht jedes Mal die Notwendigkeit, Wissen über einen Sachverhalt zu formulieren. Während im Szenario der **Zusammenarbeit** der Fokus auf dem Wissensaustausch zwischen Menschen liegt

(z. B. im Rahmen eines komplexen Softwareentwicklungsprozesses), ist im Szenario **Interoperabilität** der Fokus auf den Wissensaustausch zwischen Rechnersystemen gelegt. Im Szenario **Modellierung** liegt der Fokus auf der Wiederverwendung von Konzepten des abstrahierten Problembereiches, während im Szenario **Unterrichtung** eine Ontologie genutzt wird, um bestimmte Sachverhalte für Menschen zu erläutern, die mit dem Sachverhalt nicht vertraut sind.

Um die Formulierung einer Ontologie im Rahmen eines Softwareentwicklungsprozesses zielgerichtet voranzutreiben, ist die Festlegung auf eine oder mehrere einzunehmender Perspektiven notwendig, da somit in gewissem Maße der Grad der Abstraktion der Ontologie, der Grad der Formalisierung der Spezifikation der Ontologie sowie die Einsatzmöglichkeiten determiniert werden.

Vokabular

Grundlegend für eine Ontologie ist die Beschreibung der durch die Ontologie abstrahierten Konzepte einer Domäne. Da hiermit die Bedeutung der einzelnen Elemente klar festgelegt werden soll, kann dieser Aspekt einer Ontologie als Vokabular der Domäne aufgefasst werden. Je nach Darstellungsform, Detaillierungsgrad oder Festlegung von Konventionen wird ein Vokabular als **kontrolliertes Vokabular**, als **Glossar** oder als **Thesaurus** bezeichnet [GD+09]. Der Grad an zusätzlicher semantischer Information (Synonymdefinitonen, Relationsdefinitionen) entscheidet über die Einstufung eines Vokabulars in eine der genannten Typifikationen. Gemeinsames Charakteristikum bei allen unterschiedlichen Ausprägungen ist die Tatsache, dass ein Vokabular auf die menschlichen Bedürfnisse bei der Darstellung von Sachverhalten ausgelegt ist und die semantischen Beziehungen daher in natürlicher Sprache verfasst werden. Diese Tatsache erschwert die Verarbeitung durch Maschinen. Dennoch stellt ein Vokabular die Grundlage einer Ontologie dar, da der Interpretationsspielraum für menschliche Anwender durch entsprechende Verfeinerungen hinreichend genau eingeschränkt werden kann, damit darauf aufbauend ein Ansatz zur formalen Spezifikation erfolgen kann.

Taxonomie

Eine Taxonomie ist eine hierarchische Klassifikation von Elementen einer Domäne [GD+09]. Hierbei werden grundlegende Charakteristika einer Ontologie umgesetzt, beispielweise durch den Einsatz eines vorab klar definierten Klassifikationsschemas. Durch diese Festlegung auf ein Metamodell zur Beschreibung der eigentlichen Information wird bereits die Möglichkeit eröffnet, eine Taxonomie durch Maschinen bearbeitbar darzustellen. Hierin liegt auch der Unterschied zu einem Vokabular begründet, das auf menschlichen Bedürfnisse ausgelegt ist. Zusammengenommen stellt ein Vokabular und eine Taxonomie die Grundlage dafür dar, das durch eine Ontologie spezifizierte Konzept einer Domäne formal abzubilden.

Zusammengefasst kann festgestellt werden, dass die Definition einer Ontologie zur Unterstützung eines Softwareentwicklungsprozesses einen integralen Bestandteil darstellt. Im weiteren Sinne kann nach der Definition in [OMG-ODM] bereits die Festlegung auf ein gemeinsames Vokabular als grundlegende Ausprägung einer Ontologie aufgefasst werden. Durch die Definition einer gemeinsamen Syntax und Semantik der entsprechenden Festlegungen sowie einer formalen Definition

dieser Syntax und Semantik wird die Ontologie im Rahmen von modellgetriebenen Entwicklungsvorgehen nutzbar [GD+09].

Unterschiedliche Zielsetzungen einer Ontologie

Da für verschiedene Einsatzzwecke verschiedene Ausprägungen einer Ontologie möglich sind, ist die genaue Festlegung gewünschter Zielsetzungen grundlegend. Es lassen sich die vier verschiedenen Einsatzszenarien Kollaboration, Interoperabilität, Wissensvermittlung und Modellierung identifizieren [GD+09].

Bei der **Kollaboration** wird die Zusammenarbeit verschiedener Fachspezialisten durch die Festlegung gemeinsamer Begrifflichkeiten erleichtert. Hier steht der Austausch von Wissen zur Laufzeit im Vordergrund, bei dem Teilnehmer verschiedener Fachdomänen involviert sind. Wird die Kollaboration von softwarebasierten Agenten betrachtet, wird eine Ontologie zur Spezifikation von Nachrichtenprotokollen eingesetzt. Die Nutzung von Information aus verschiedenen Quellen erfordert die Wahrung eines Mindestmaßes an **Interoperabilität** verschiedener Informationsanbieter. Die **Wissensvermittlung** zielt, ähnlich der Kollaboration, auf den Austausch von Information zwischen menschlichen Teilnehmern ab, erfolgt jedoch unidirektional. Wird eine Wiederverwendung der definierten Konzepte einer Ontologie angestrebt, rückt die **Modellierung** in den Blickpunkt. Hierbei wird eine Ontologie eingesetzt, um die Erkenntnis bestimmter Sachverhalte in weiteren Szenarien einzusetzen, die unabhängig vom ursprünglichen Verwendungszweck sind.

Service oriented architecture Modeling Language (SoaML)

Durch die vermehrte Umsetzung von SOA-Projekten wurden in jüngster Vergangenheit verschiedene Ansätze eingeführt und vorgestellt, um die Modellierung von Diensten und Dienstarchitekturen auf Basis bestehender Modellierungssprachen zu unterstützen. So kann beispielsweise der Entwurf von Dienstschnittstellen bzw. Dienstkomponenten durch UML-Klassendiagramme [OMG-UML-Super] unterstützt werden. Um einen einfachen semantischen Bezug zwischen dem Konzept eines Dienstes und dem entsprechenden Modellelement herzustellen, können mit dem Profilmechanismus der UML Stereotype eingeführt werden, um hierdurch auf Metamodellebene verschiedenartige Bedeutungen einzelner Modellelemente klar voneinander zu trennen.

Um zu verhindern, dass eine Vielzahl unterschiedlicher Eigenentwicklungen eine Interoperabilität auf Ebene des Softwareentwurfes erschweren, strebt die OMG derzeit die Spezifikation eines Modellierungsstandards an, um die wesentlichen Konzepte einer auf Diensten basierenden Softwarearchitektur mit den formalen Mitteln der UML auszudrücken. Diese von der OMG als *Service oriented architecture Modeling Language* (SoaML) bezeichnete Modellierungssprache befindet sich zum Zeitpunkt der Erstellung der vorliegenden Arbeit in einem finalen Standardisierungsprozess und liegt derzeit in der Version 1.0 Beta2 vor [OMG-SoaML]. Die SoaML basiert im Wesentlichen auf den Vorarbeiten des von IBM vorgestellten *UML Profile and Metamodel for Services* (UPMS, [OMG-UPMS]).

Die Zielsetzung von SoaML ist klar auf die Unterstützung eines architektonischen Entwurfes einer auf Diensten basierenden Architektur ausgelegt. Die von der SoaML vorgestellten Konzepte können

sowohl in Form eines UML-Profils als auch in Form einer eigenständigen Metamodell-Definition implementiert werden. Von der OMG wird der leichtgewichtige Ansatz des UML-Profils vorgezogen und auf Basis einer in XML-codierten Syntax für den plattformunabhängigen Austausch von Modellspezifikationen angeboten (engl. *Extensible Markup Language Metadata Interchange*, XMI [OMG-XMI]).

Die durch die SoaML vorgegeben Modellierungsmöglichkeiten bewegen sich prinzipiell auf verschiedenen Abstraktionsniveaus und zielen auf unterschiedliche Blickpunkte, die die durch eine dienstorientierte Architektur umgesetzten werden. Somit wird ein Einsatz dieser Sprache in sowohl in der dienstorientierten Analyse als auch im dienstorientierten Entwurf möglich.

Im Kern der Konzepte der SoaML stehen die Modellierungselemente Fähigkeit (engl. *Capability*), der Dienst (engl. *Service*), die Dienstschnittstelle (engl *Service Interface*), der Teilnehmer (engl. *Participant*), der angebotene und genutzte Zugangspunkt (engl. *Service Point* und *Request Point*), der Dienstvertrag (engl. *Service Contract*) sowie die Dienstarchitektur (engl. *Service Architecture*). In der Spezifikation des UML-Profils werden die englischen Bezeichner zusammengeschrieben (z. B. *ServiceInterface* oder *ServiceContract*).

Participant (Teilnehmer)

Ein Teilnehmer ist ein Objekt in einer dienstorientierten Architektur, das einen Dienst entweder anbietet (*provides*) oder einen Dienst benötigt (*requires*). Ein Teilnehmer kann als Repräsentation verschiedener real existierender Elemente aufgefasst werden. So können menschliche Teilnehmer, vollständige Organisationen, aber auch Komponenten von Informationssystemen durch das Modellelement `Participant` dargestellt werden. Je nach der Rolle, in der sich ein Teilnehmer befindet, wird er als *Service Provider* (falls er einen Dienst anbietet) oder *Service Consumer* (falls er einen Dienst einfordert) charakterisiert. In der UML-Profildefinition wird das Element `Participant` als stereotypisiertes Element vom UML-Metamodellelement `Class` abgeleitet.

Die Interaktionspunkte, an denen die eigentliche Dienstnutzung stattfindet, werden wieder je nach Bedeutung als `ServicePoint` oder als `RequestPoint` als stereotypisiertes Modellelement des UML-Metamodellelementes `Port` definiert.

Capability (Fähigkeit)

Fähigkeiten gruppieren Sammlungen von Funktionalitäten, die logisch zueinander gehören. In [OMG-SoaML] wird die Semantik des Elementes `Capability` wie folgt definiert:

> *"A Capability is the ability to act and produce an outcome that achieves a result. This element allows for the specification of capabilities and services without regard for the how a particular service might be implemented and subsequently offered to consumers by a Participant. It allows architects to analyze how services are related and how they might be combined to form some larger capability prior to allocation to a particular Participant."*

Demnach wird von den Teilnehmern, die einen Dienst realisieren, vollständig abstrahiert. Hierdurch eignet sich das Element, in der dienstorientierten Analyse zur Identifikation von Dienstkandidaten eingesetzt zu werden [Am10, De10]. Abhängigkeiten zwischen Dienstkandidaten können durch den Einsatz von `use`-Relationen dargestellt werden. Weiterhin kann eine modellierte `Capability` durch Operationen spezifiziert werden, was zur Modellierung von identifizierten möglichen Operationen der Dienstkandidaten für den späteren Entwurf genutzt werden kann.

ServiceInterface (Dienstschnittstelle)

Das `ServiceInterface` kann zur Modellierung von Dienstschnittstellen genutzt werden. In [OMG-SoaML] ist die Semantik dieses Modellelementes wie folgt definiert:

> *"A ServiceInterface defines the interface and responsibilities of a participant to provide or consume a service. It is used as the type of a Service or Request Port. A ServiceInterface is the means for specifying how a participant is to interact to provide or consume a Service. A ServiceInterface may include specific protocols, commands and information exchange by which actions are initiated and the result of the real world effects are made available as specified through the functionality portion of a service."*

Während durch die `Capability` lediglich die Fähigkeiten beschrieben werden, die ein Dienst anbietet und somit zunächst unverbindlich sind, wird durch die Definition eines `ServiceInterface` eine genaue Festlegung getroffen, wie der spezifische Dienst genutzt werden kann. Im Gegensatz zur Dienstfähigkeit werden die Dienstoperationen vollständig spezifiziert. Hierbei werden Interaktionsprotokolle definiert, in denen klar beschrieben wird, in welcher Reihenfolge der Nachrichtenaustausch geregelt ist, um verschiedene Dienstoperationen aufzurufen.

ServicesArchitecture (Dienstarchitektur)

Einen architektonischen Überblick über die Komposition mehrerer Dienste kann mit dem von der UML-Metaklasse `Collaboration` abgeleiteten Modellelement `ServicesArchitecture` definiert werden. Hierbei werden die einzelnen Dienstschnittstellen in deren spezifischen Rollen zueinander in Beziehung gesetzt.

Service (Dienst)

Ein Dienst (`Service`) in der SoaML ist ein Modellierungselement, um Funktionen darzustellen, die von einem Teilnehmer (`Participant`) an andere angeboten werden. Ein Dienst wird über einen klar definierten Zugangspunkt (`Port`) spezifiziert, durch den die Fähigkeiten (`Capabilities`), die dieser Dienst realisiert, genutzt werden können. Die Nutzung der Fähigkeiten eines Dienstes erfolgt auf Basis klar definierter Protokolle, einer Randbedingungen (engl. *Constraint*) und Schnittstellen.

Da ein Dienst die Fähigkeiten eines Teilnehmers anbietet, wird das Metamodellelement `Service` als Stereotyp des UML-Metamodellelementes `Port` definiert und in einem SoaML-Modell einer Instanz des Elementes `Participant` zugeordnet. Der Dienst wird durch seine Dienstschnittstelle (`ServiceInterface`) definiert, im Umkehrschluss wird ein Dienst durch einen Teilnehmer implementiert.

Kategorie (Category)

Zur Beschreibung verschiedener Aspekte oder Eigenschaften, die einem SoaML-Modell zugrunde liegen, kann mit dem Metamodellelement Kategorie (`Category`) Information hinsichtlich gruppierender oder klassifizierender Charakteristika an beliebige konkrete Instanzen von SoaML-Modellelementen zugwiesen werden. Hierbei kann ein Dienst in mehrere Kategorien eingeordnet werden, und eine Kategorie kann mehrere Dienste betreffen. Das Metamodellelement Kategorie wird als Verfeinerung des vom UML-Metamodellelement `Artifact` abgeleiteten Metamodellelement `NodeDescriptor` definiert.

Katalog (Catalog)

Die Definition eines Kataloges (`Catalog`) ermöglicht die Gruppierung und hierarchische Strukturierung verschiedener Kategorien.

2.4 Prozessorientierung im IT-Management

Dem Einsatz von Informationstechnologie wird eine herausragende Rolle zugesprochen, stellt es doch das Bindeglied zwischen fachbezogener Wertschöpfung und realisierender Werkzeuge dar. Vielfältige Erwartungen werden daher von den Nutzern an die Betreiber von IT-Systemen gestellt. Um den Betrieb dieser IT-Systeme qualitätsgesichert, zielorientiert und hinsichtlich effizientem Einsatz von Personalkosten und Sachmittelkosten vorausschaubar verantworten zu können, sind geeignete Maßnahmen zu definieren, in einer Organisation einzuführen und vor dem Hintergrund der Verantwortung beim Betrieb dieser Systeme fortwährend zu beherrschen.

Nach [ISO89] wird der Begriff **Management** wie folgt definiert:

> *„Management is related to activities which control or monitor the use of resources“*

Eine differenziertere Sichtweise in Bezug auf die Tatsache, dass vernetzte Systeme im Kern der Betrachtung stehen, wird in [HA+99] durch die Definition des Begriffes IT-Management wie folgt dargelegt:

> *„… alle Maßnahmen für einen unternehmenszielorientierten effektiven und effizienten Betrieb eines verteilten Systems mit seinen Ressourcen…“*

Hieraus lassen sich sowohl der Zweck sowie der Gegenstand der Disziplin IT-Management identifizieren. Durch die Forderung, den betrieblichen Zweck unternehmenszielorientiert, effektiv und effizient auszurichten, werden grundlegende Prämissen bezüglich des Entwurfes von Softwarelösungen für diesen Bereich gestellt.

Das IT-Management ist zunächst ressourcenzentriert in Bezug auf die IT-Infrastruktur. Zu den Ressourcen einer IT-Infrastruktur werden alle Netzwerkkomponenten, Systemkomponenten sowie logische Anwendungskomponenten gezählt. Ganzheitlich betrachtet kann eine Schichtung von den Netzwerk- über die Systemkomponenten hin zu den logischen Anwendungskomponenten festgelegt

werden. Während in den Anfangszeiten des Einsatzes vernetzter IT-Systeme vom Blickpunkt des Betriebes vor allem technische Rahmenparameter betrachtet wurden, wird in jüngster Zeit verstärkt eine Ausrichtung an den Bedürfnissen der Nutzer der IT-Systeme angestrebt. Daher kann das Management von IT-Diensten als konsequente Fortführung des technisch-orientierten Betriebes der Netz-, System- und Anwendungskomponenten aufgefasst werden [GD+09].

ISO/IEC20000-1:2005

Mit der Definition des einheitlichen und internationalen Standards ISO/IEC20000-1:2005 [ISO05, ISO05b, ISO09] liegt erstmals eine gemeinsame Grundlage vor, um die funktionalen Mindestanforderungen zur Strukturierung und zur Gliederung von Informationssystemen für betriebliche Abläufe im IT-Management festzulegen. Der Standard basiert im Wesentlichen auf den in der *Information Technology Infrastructure Library* (ITIL) [OGC07, OGC07b, OGC07c, OGC08, OGC08b] beschriebenen *Best Practices*, führt aber hinsichtlich verschiedener Aspekte, die den kompletten Lebenszyklus eines IT-Dienstes betreffen, weitergehende Mindestanforderungen ein. Die eigentliche standardisierte Definition der verschiedenen Managementprozesse erfolgt in [ISO05], während die zusätzlichen von der ISO zur Verfügung gestellten Dokumente weitergehende und auf der Standarddefinition aufbauende Vorgehensweisen beschreiben [ISO05b, ISO09].

Die Definition des Standards zielt weniger auf die formale Beschreibung der umzusetzenden betrieblichen Abläufe bei einer Dienstleisterorganisation ab, sondern vielmehr auf die grundsätzlich notwendigen Managementfunktionalitäten, die durch die einzelnen Managementprozesse erbracht werden sollen. Prinzipiell teilt der Standard die Gesamtheit der anfallenden Abläufe in fünf Kategorien (*Service-Delivery*-Prozesse, *Release*-Prozesse, *Control*-Prozesse *Resolution*-Prozesse und *Relationship*-Prozesse) auf, die weiter untergliedert sind und die eigentliche Definition von 13 Managementprozessen umfassen. In Abbildung 6 ist die Gesamtheit der Managementprozesse, die in ISO/IEC20000-1:2005 definiert werden, anhand der zugehörigen Kategorien dargestellt.

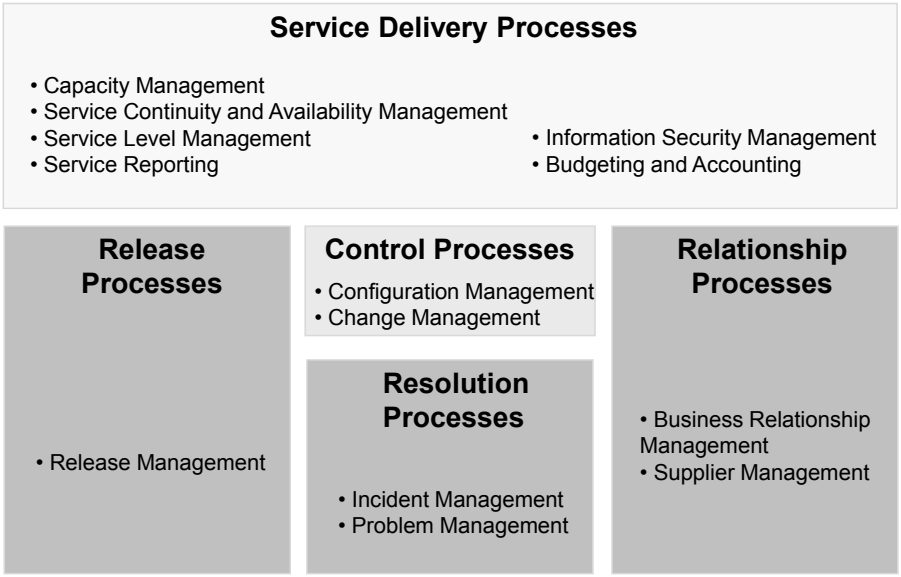

Abbildung 6 Übersicht ISO/IEC20000-1:2005

Insgesamt eignet sich die Definition des Standards dazu, ein einheitliches Begriffsverständnis über die einzelnen Konzepte der Domäne IT-Management zu etablieren. Durch die Beschreibung in natürlich-sprachigem Text erfolgen eine nachvollziehbare Definition der Semantik der einzelnen Managementprozesse und deren Managementaktivitäten. Der Entwurf von wiederverwendbaren Diensten einer dienstorientierten Architektur erfolgt daher auf der Grundlage der in [ISO05] festgelegten funktionalen Mindestanforderungen.

Service Delivery-Prozesse

Die größte Gruppe umfasst insgesamt sechs verschiedene Managementprozesse, die ihrer Zielsetzung folgend hauptsächlich auf organisatorische Aspekte bezüglich der Verwaltung, Aushandlung und Überwachung bestehender oder zu ändernder IT-Dienste angelegt sind. Hierzu zählen *Capacity Management*, *Service Continuity and Availability Management*, *Service Level Management*, *Service Reporting*, *Information Security Management* sowie *Budgeting and Accounting*.

Release-Prozesse

In dieser Kategorie ist in der gegenwärtigen Standarddefinition [ISO05] lediglich ein Managementprozess definiert (*Release Management Process*). Zielsetzung ist, funktionale Mindestanforderungen an ein Informationssystem festzulegen, das bei der Durchführung von geplanten Änderungen an IT-Diensten eine systemische Unterstützung anbietet.

Control-Prozesse

Die beiden Managementprozesse (*Configuration Management* und *Change Management*) in dieser Kategorie stellen zentrale und von allen weiteren Managementprozessen genutzte Funktionalität bereit. Im *Configuration Management* wird mit der *Configuration Management Database* (CMDB) ein zentrales Informationssystem betrieben, das eine ganzheitliche Sicht auf Konfigurationsinformationen liefern soll. Das *Change Management* betrachtet die koordinierte Durchführung von Änderungen, adressiert jedoch den organisatorischen Aspekt.

Resolution-Prozesse

Die beiden in der Kategorie der *Resolution*-Prozesse definierten Managementprozesse zielen auf die im täglichen Betrieb anfallenden Managementaufgaben ab. Aufgrund ihrer organisatorischen Struktur, klaren Gliederung und geringen Komplexität eignen sich sowohl der *Incident-Management*- als auch der *Problem-Management*-Prozess dazu, durch geeignete Informationssysteme (teil-)automatisiert zu werden.

Relationship-Prozesse

Die zur Kommunikation mit externen Dienstleistern oder Drittherstellern notwendigen funktionalen Mindestanforderungen an ein Informationssystem werden in der Kategorie *Relationship*-Prozesse

definiert. Die beiden Managementprozesse in dieser Kategorie weisen einen sehr geringen domänenspezifischen Aspekt auf, sind jedoch in einer ganzheitlichen Standarddefinition erforderlich.

3 Stand der Technik

Vor dem Hintergrund der im ersten Kapitel genannten Problemstellungen werden nun zunächst wesentliche Anforderungen an die eigene Lösung formuliert. Auf der Grundlage dieser Anforderungen werden anschließend ausgewählte bestehende Arbeiten und Ansätze bewertet. Den Abschluss dieses Kapitels bildet eine Zusammenfassung, die auf Basis der Bewertung von bestehenden Arbeiten und Ansätzen den notwendigen Handlungsbedarf formuliert.

3.1 Anforderungen

Die Analyse von geschäftlichen Abläufen, die Erstellung von Geschäftsprozessmodellen sowie die Umsetzung auf konkrete Elemente einer unterstützenden Softwarearchitektur wird wegen der inhärent gegebenen Komplexität dieser Aufgabenstellung in der Regel von unterschiedlichen involvierten Verantwortlichen durchgeführt. Demgegenüber steht ein ebenso komplexer Konstruktionsprozess, um die Elemente einer dienstorientierten Architektur – hier im Speziellen einer dienstorientierten Managementarchitektur – zu entwerfen. Um beide Aspekte pragmatisch ineinander zu überführen und eine ganzheitliche Lösung zu entwickeln, ist ein integriertes Vorgehen notwendig. Die Festlegung auf ein einheitliches und verbindliches Vokabular, um die Bedeutung der einzelnen Modellelemente hinsichtlich ihrer jeweiligen Semantik genau festzulegen, ist daher unerlässlich.

Der Lösungsbeitrag der vorliegenden Arbeit besteht darin, ein Vorgehen für den Entwurf von prozessorientierten und wiederverwendbaren Managementdiensten bereitzustellen, das als Grundlage zur automatisierten Ausführung von Betreiberprozessen genutzt werden kann. Die Anforderungen an ein Softwaresystem, lassen sich daher in die beiden Kategorien **Anforderungen an den Entwurf von Managementdiensten** sowie **Anforderungen an die Abbildung von Managementprozessen** unterteilen.

3.1.1 Anforderungen an den Entwurf von Managementdiensten

Die Grundlage für die Schaffung einer dienstorientierten Managementplattform bildet der an den wesentlichen Eigenschaften eines Softwaredienstes ausgerichtete zielorientierte Entwurf. Dabei werden mit Diensten im Allgemeinen spezielle Eigenschaften – beispielsweise lose Kopplung, Autonomie, Systemunabhängigkeit, Zustandslosigkeit oder Wiederverwendbarkeit – verbunden. Spezielle Anforderungen erhält der Entwurf in der Domäne IT-Management aus der Tatsache, dass die Konstruktion einer dienstorientierten Managementplattform zum Ziel hat, bestehende Managementwerkzeuge in einem integrierten Gesamtkonzept zu betrachten, weshalb eine Abstraktion über die eigentlichen durch die einzelnen Werkzeuge umgesetzten Managementfunktionen erfolgen muss. Zusätzlich muss den Eigenschaften einer prozessorientierten Unterstützung von Managementaktivitäten Rechnung getragen werden. Hieraus lassen sich insgesamt weitere Verfeinerungen an die Anforderungen formulieren, die im Folgenden nun detailliert dargestellt und erläutert werden.

Anforderung 1.1: Referenzcharakter des Dienstentwurfes

Grundlegende Voraussetzung für ein Verfahren zur Spezifikation von Managementdiensten ist die Eigenschaft der Nachvollziehbarkeit der Methode. Somit wird sichergestellt, dass die einzelnen Entwurfsentscheidungen analysiert, diskutiert werden und in der Gesamtheit betrachten die identifizierten Problemstellungen zielgerichtet lösen. Die Forderung nach der Nachvollziehbarkeit bezieht sich auf die Spezifikation des Vorgehens in Form einer oder mehrerer logisch ersichtlicher Schritte. Das Ergebnis dieser konstruktiven Methode ist ein Referenzmodell zur Spezifikation von Managementdiensten. Ein Referenzmodell ist nach [BD+07] eine Abstraktion einer Problemstellung zur Lösung mehrerer gleichartiger Anwendungsfälle und kann unabhängig von konkreten Anwendungsfällen bezüglich gewisser gewünschter Eigenschaften untersucht und bewertet werden.

Darauf aufbauend wird durch den in der vorliegenden Arbeit vorgestellten konstruktiven Ansatz das Ziel verfolgt, in der Praxis Anwendung zu finden. Zwei Aspekte sind besonders relevant: Die zur Spezifikation eines Referenzentwurfes eingesetzten Analyse- und Modellierungssprachen müssen bekannte und durch Entwicklungswerkzeuge unterstütze Ansätze darstellen. Außerdem muss die Methode für den Entwurf der Managementdienste in der Praxis durch bestehende Entwicklungswerkzeuge realisierbar sein. Dies bedeutet, dass vor allem solche Ansätze Beachtung finden, die auf standardisierten Modellierungssprachen beruhen.

Anforderung 1.2: Unabhängigkeit von konkreten Managementwerkzeugen

Mit dem Ziel, eine dienstorientierte Managementplattform zu etablieren, um Betreiberprozesse automatisiert auszuführen, wird neben der Integration von Werkzeugen zur Unterstützung von organisationsbezogenen Aspekten auch eine Integration derjenigen Werkzeuge erforderlich, die die technischen Aspekte beim Betrieb von IT-Infrastrukturen abdecken. Hierzu zählt insbesondere die Überwachung und Steuerung von ressourcennahen Infrastrukturkomponenten, aber auch die Überwachung und Steuerung von logischen Anwendungskomponenten. Eine umfassende Gesamtbetrachtung, um den Entwurf von Managementdiensten integriert zu beschreiben, bezieht daher neben allgemeinen Prozessanforderungen auch die Fähigkeiten ein, die durch bestehende Managementwerkzeuge angeboten werden. Eine Umsetzung von notwendigen Managementfunktionen, die unabhängig von konkreten Werkzeugumsetzungen sind, ist wünschenswert. Als Anforderung an eine dienstorientierte Managementplattform kann daher gestellt werden, dass der hierzu betrachtete Entwurf von Managementdiensten etablierte Konzepte bestehender Managementansätze einbezieht, jedoch von konkreten Werkzeugen unabhängig ist und weitestgehend abstrahiert, um eine praktische Anwendbarkeit sicherzustellen.

Anforderung 1.3: Metamodell der Domäne

Häufig geht dem eigentlichen Entwurf eine detaillierte Analyse des zu lösenden Problems durch einen Experten der Domäne voraus. Dieser Domänenexperte ist mit Regeln, Einschränkungen und Besonderheiten des betrachteten Problembereiches vertraut ist. Eine Überführung dieses Wissens von Analyseartefakten auf Entwurfsartefakte, die dann von den jeweiligen Entwicklungsabteilungen interpretiert werden, geschieht zumeist auf Basis unterschiedlicher Modellierungsansätze (z. B. durch:

Geschäftsprozessdiagramme, Geschäftsentitätsdiagramme, Verteilungsdiagramme, Anwendungsfalldiagramme). Eine Beherrschung aller unterschiedlichen Modellierungsansätze aus der Perspektive eines Softwareentwicklers steht jedoch oftmals nicht im Fokus der Domänenexperten. Dieser Bruch führt zu unvollständigen und teilweise widersprüchlichen Analyse- und Entwurfsartefakten. Erschwerend kommt hinzu, dass der Einsatz von generischen Modellierungssprachen (engl. *General Purpose Languages*, GPL) aufgrund des breiteren Fokus und der nicht klar definierten Semantik der jeweiligen Modellelemente einen Interpretationsspielraum bei der Überführung von Elementen unterschiedlicher Modelle zulässt. Der Einsatz einer domänenspezifischen Sprache (engl. *Domain Specific Language*, DSL) kann dem entgegenwirken. Bei diesem Ansatz werden zunächst die Elemente der Domäne unabhängig von einer konkreten Implementierung beschrieben. Dieses als Domänenmodell bezeichnete Analyseartefakt reflektiert somit die fachfunktionalen Anforderungen einer zu entwickelnden Softwarelösung und stellt die direkte Schnittstelle zwischen dienstorientierter Analyse und dienstorientiertem Entwurf dar. Um den Entwurf von Managementdiensten auf eine durchgängige Art und Weise zu unterstützen ist daher ein Ansatz erforderlich, der die strukturellen Beziehungen der Elemente der Domäne in Beziehung zueinander setzt und die Transformation auf eine dienstorientierte Architektur auf Basis dieses domänenspezifischen Entwurfes ermöglicht. Dieses Metamodell der Domäne bezieht sowohl Aspekte für den automatisierten Ablauf von Betreiberprozessen ein als auch Aspekte zur Integration bestehender Managementwerkzeuge.

Anforderung 1.4: Wiederverwendbarkeit von Managementdiensten

Eines der wesentlichen Ziele, das mit der Einführung einer dienstorientierten Softwarearchitektur verbunden wird, ist die Wiederverwendung bestehender Softwarekomponenten. Durch die Spezifikation einer verbindlichen Schnittstelle wird es dem Nutzer eines Dienstes ermöglicht, Fachfunktionalität auf Basis externer wie auch interner Komponenten zu realisieren. Um dieses Ziel zu erreichen, ist jedoch die Anforderung der Wiederverwendbarkeit als wesentliches Entwurfsziel in den Entwurfsprozess zu integrieren. Gegenstand der Arbeit ist demnach der Entwurf von Managementdiensten vor dem Hintergrund der Kernanforderung der Wiederverwendbarkeit der entstandenen Entwurfsartefakte.

Zur abschließenden Beurteilung eines konkreten Entwurfes von Managementdiensten hinsichtlich der geforderten Eigenschaft der Wiederverwendbarkeit wird ein Ansatz benötigt, mithilfe dessen gesicherte Aussagen über das Maß der Wiederverwendbarkeit getroffen werden können. Ein anerkanntes und gängiges Mittel hierzu ist die Definition einer Metrik zur Bestimmung eines solchen qualitativen Aspektes. Da aus Anforderung 1.3 hervorgeht, dass der Entwurf von Diensten anhand eines strukturierten Modells der Domäne in einem integrierten Entwicklungsvorgehen durchzuführen ist, wird als Ergänzung hierzu gefordert, dass eine Metrik zur Bestimmung der Ausprägung der Wiederverwendbarkeit eines Entwurfes von Managementdiensten sowohl in die Methode eingebettet sein, als auch eine praktische Anwendbarkeit zur Evaluation der entstandenen Modelle sichergestellt werden muss.

3.1.2 Anforderungen an die Abbildung von Managementprozessen

Die Unterstützung definierter Betreiberprozesse durch ein Softwaresystem stellt die Grundlage für einen qualitätsgesicherten Betrieb heutiger komplexer verteilter IT-Infrastrukturen dar [HA+99]. Um die Abbildung dieser Prozesse im Rahmen eines Softwareentwicklungsprozesses zu realisieren, müssen daher dedizierte Anforderungen sowohl an die Spezifikation der zu unterstützenden Prozesse als auch Anforderungen an die Methode zur Abbildung gestellt werden. Als Fortführung der Festlegung von Anforderungen an eine Softwarelösung zu den in Abschnitt 1.3 beschriebenen Problemstellungen werden im Folgenden diese Anforderungen genauer beschrieben.

Anforderung 2.1: Flexible Anwendbarkeit der Ableitungsmethode

Um den sich ändernden Anforderungen von Geschäftsprozessen gerecht zu werden, ist eine flexible Unterstützung durch die den Geschäftsprozessen zugrunde liegenden IT-Dienste notwendig. Dies hat jedoch auch Auswirkungen auf den Betrieb dieser IT-Dienste. Änderungen aufseiten der IT-Dienste implizieren zumeist Änderungen an der logischen Zusammensetzung der hierzu notwendigen Infrastrukturkomponenten. Damit verbunden ist oftmals auch eine notwendige Anpassung von Überwachungs- und Steuerungsprozessen. Wesentliche Anforderung an die Abbildung von Betreiberprozessen auf die dienstorientierte Architektur ist daher eine Unterstützung der Softwareentwickler im Falle von Änderungen im Prozessablauf. Dadurch wird sichergestellt, dass im Falle von Änderungen an den Betreiberprozessen diese Änderungen auf die unterstützende Managementplattform abgebildet werden können.

Anforderung 2.2: Einheitliche Begriffe auf Basis eines Domänenmodells

Um die Abbildung unterschiedlicher Aktivitäten der Managementprozesse auf die durch die dienstorientierte Managementplattform bereitgestellten Managementfunktionen effizient zu gestalten, ist ein gemeinsames Verständnis der Semantik der konkreten Instanzen der Prozessmodellelemente notwendig. Aufgrund fehlender standardisierter Vorgaben zur Schaffung eines einheitlichen Begriffsverständnisses kann bislang keine generische Transformation angegeben werden, die dedizierte Prozesse auf bereitgestellte Dienstoperationen abbildet. Gerade aber die Einbeziehung eines solchen vereinheitlichten Begriffsverständnisses kann die Entwicklung solcher Transformationen maßgeblich beeinflussen. Idealerweise basiert dieses Begriffsverständnis auf denselben Elementen, auf denen auch das Domänenmodell für den Entwurf der Managementdienste beruht. Damit wird ein durchgängiges Begriffsverständnis ermöglicht. Die Nutzung eines auf denselben Elementen beruhenden Modells zur Beschreibung der Aktivitäten innerhalb der Managementprozesse ist demnach wesentliche Anforderung für eine durchgängige und nachvollziehbare Umsetzungsmethodik.

Anforderung 2.3: Formale Modellierung von Managementprozessen

Für die Darstellung des zeitlichen Ablaufes aufeinanderfolgender Tätigkeiten werden im Rahmen des Entwicklungsprozesses Diagramme (Ablaufdiagramme, Flussdiagramme) des zugehörigen Prozesses erstellt. Da diese erstellten Artefakte die Grundlage für eine Abbildung auf die durch die dienstorientierte Managementplattform bereitgestellten Managementfunktionen darstellen, ist die

Auswahl einer konkreten Prozessmodellierungssprache maßgebend für die Methodik und somit entscheidend für die Gesamtqualität des erstellten Softwaresystems. Gewünschte Voraussetzung ist ein zu der eingesetzten Modellierungssprache verfügbares, formal spezifiziertes Metamodell. Anhand dieses Metamodells können zum einen semantische Bezüge der eingesetzten Prozessmodellelemente klar definiert werden, darauf aufbauend können dann aber auch Transformationen spezifiziert werden, die die Abbildung des Prozessmodells auf die zugrunde liegende dienstorientierte Managementplattform erleichtern. Wesentliche Anforderung an die Auswahl der Modellierungssprache ist daher sowohl die Existenz eines zugrunde liegenden formalen Metamodells als auch der semantisch klare Bezug der Modellelemente.

3.1.3 Zusammenfassung des Anforderungskataloges

Mit den vorgestellten Anforderungen lässt sich zusammenfassend ein Anforderungskatalog aufstellen, der als Grundlage zur Bewertung bestehender Arbeiten und Ansätze dient. Auf Basis dieses Anforderungskataloges wird der Handlungsbedarf vor dem Hintergrund teilweiser umgesetzter bzw. vollständig fehlender Anforderungen motiviert. Tabelle 2 fasst die einzelnen Anforderungen zusammen.

Anforderungskatalog	
A1 – Entwurf von Managementdiensten	
A1.1	Referenzcharakter des Dienstentwurfes
A1.2	Unabhängigkeit von konkreten Managementwerkzeugen
A1.3	Integriertes Metamodell der Domäne
A1.4	Nachweis der Wiederverwendbarkeit von Managementdiensten
A2 – Abbildung automatisierbarer Managementprozesse	
A2.1	Flexible Anwendbarkeit der Ableitungsmethode
A2.2	Einheitliche Begriffe auf Basis eines Domänenmodells
A2.3	Modellierung von Managementprozessen

Tabelle 2 Übersicht über den Anforderungskatalog

3.2 Entwurf von Managementdiensten

Die dienstorientierte Architektur als Ansatz zur Lösung der Integrationsaufgaben im IT-Management erfordert die Definition von Managementbasisdiensten und komponierten Managementprozessdiensten. Während Dienste fachlich-motiviert sind und dadurch einen klaren Bezug zu den unterstützenden Betreiberprozessen aufweisen, müssen gleichwohl die bestehenden Werkzeuge betrachtet werden, da eine Integration selbiger in die entwickelte Lösung angestrebt wird. Die bestehenden Ansätze zur Definition von Managementdiensten werden in diesem Abschnitt diskutiert und in Bezug auf den Kriterienkatalog bewertet. Da der Entwurf von wiederverwendbaren Managementdiensten sowohl die Betrachtung von Forschungsarbeiten aus dem Bereich **IT-Management** als auch aus dem Bereich **dienstorientierte Softwareentwicklung** (dienstorientierte SWE) einschließt, wird die Betrachtung verwandter Ansätze anhand dieser beiden Klassifikationskriterien geordnet.

3.2.1 Ansätze aus dem Bereich IT-Management

Zunächst werden Arbeiten aus dem Bereich IT-Management betrachtet und entlang der gestellten Anforderungen bewertet. Da die hier vorgestellten Arbeiten die architektonischen Prinzipien eines auf Diensten basierenden Managementsystems motivieren, sind diese Arbeiten von grundlegender Bedeutung für die Bewertung der eigenen Beiträge.

Anerousis, 1999: An architecture for building scalable, Web-based management services

Bereits im Jahre 1999 beschreibt Anerousis die Idee, zukünftige Managementsysteme auf Basis einfacher Managementagenten zu erschaffen, um somit die Skalierbarkeit zur Begegnung steigender Anforderungen an den Betrieb vernetzter Infrastrukturen sicherzustellen. Anerousis stellt fest, dass eine wesentliche Beobachtung beim Betrieb vernetzter IT-Infrastrukturen dahingehend gemacht werden kann, dass der Schwerpunkt eines Managementsystems nicht mehr auf den eingesetzten Managementtechnologien (Managementprotokolle wie beispielsweise SNMP, CMIP) liegt, sondern vielmehr auf den von einem konkreten Managementsystem eingesetzten Managementfunktionen [An99]. Da IT-Infrastrukturen heutzutage eine kritische Bedeutung in vielen geschäftlichen Abläufen innehaben, stellen Anforderungen hinsichtlich eines qualitätsgesicherten Betriebes, beispielsweise auf der Grundlage vereinbarter *Service Level Agreements*, Betreiberorganisationen vor die Notwendigkeit, diese Anforderungen auf Basis bestehender Systeme als auch zusätzlich zu integrierender Systeme umzusetzen. Dem Autor zufolge ist daher die Frage entscheidend, inwiefern sich konkrete Komponenten eines Managementsystems integrieren lassen oder aber, wie diese Komponenten an neuartige Bedürfnisse angepasst werden können. Weiterhin ändern sich dem Autor nach nicht nur die eingesetzten Managementsysteme, sondern auch die durch Managementsysteme zu betreibenden Infrastrukturen in einem immer schneller werdenden Zyklus. Dies erfordert gerade für den qualitätsgesicherten Betrieb dieser Infrastrukturen die Definition von Managementschnittstellen, die von einem höheren Abstraktionsniveau aus motiviert werden.

Ein Lösungsansatz zur Begegnung dieser Herausforderungen besteht darin, Managementsysteme auf Basis skalierbarer und webbasierter Managementdienste zu erschaffen. Konkret führt der Autor für diesen Ansatz vier Vorteile an:

(1) Die Verantwortung für den Betrieb von Managementdiensten liegt beim Anbieter dieser Dienste. Der Nutzer von Managementdiensten ist nicht länger gezwungen, Managementsoftware zu installieren, was ihn letztlich von den Kosten für den Betrieb dieser Systeme befreit.

(2) Die Nutzung von Managementdiensten kann vollständig auf Basis von standardisierten Softwareanwendungen wie beispielsweise Web-Browsern erfolgen. Diese sind heutzutage für eine Vielzahl unterschiedlicher Plattformen verfügbar, beispielsweise auch auf vielen mobilen Endgeräten.

(3) Durch die Ausrichtung auf die Präsentation von Managementinformation in einem webbasierten Kontext wird diese Information auf einem wesentlich höheren Abstraktionsgrad dargestellt. In der Regel wird von technischen Details wie beispielsweise SNMP-Protokollinformation abstrahiert.

(4) Durch die Nutzung eines verteilten Ansatzes zur Realisierung notwendiger Managementfunktionen wird ein entkoppelter Aufbau der zugrunde liegenden Architektur erforderlich. Somit lässt sich die Darstellung von Managementinformation bezüglich aktueller Infrastrukturzustände wesentlich vereinfachen.

Zusammengefasst kann festgestellt werden, dass der Einsatz verteilter und webbasierter Managementdienste einen echten Mehrwert für den Nutzer dieser Managementdienste darstellt.

Der vom Autor vorgestellte Lösungsansatz präsentiert zum einen ein Informationsmodell zur Beschreibung von Managementinformation in einem verteilten Umfeld sowie ein Architekturmodell zur Ausgestaltung der einzelnen Komponenten dieses verteilten Ansatzes. Das Informationsmodell hat zum Ziel, Managementinformation aggregieren zu können, um auf diese Art und Weise den Gesamtzustand eines verteilten Systems auf Basis einzelner Attribute der verteilten Agenten berechnen zu können. Das präsentierte Berechnungsmodell in Form eines verteilten Manager/Agenten-Ansatzes bietet hierfür spezielle Managementfunktionen an wie beispielsweise *get*, *set* und *action* für den Zugriff auf Attribute von *Managed Objects* oder *getParent*, *getRoot* und *getPath* für die Navigation in einem logischen Baum von *Managed Objects.*

In Abbildung 7 ist der konzeptionelle Lösungsansatz skizziert. Ersichtlich ist eine klare Trennung in drei unabhängige Ebenen (*Management Applications*, *Objects representing computed views*, *Element management information*), sowie die Einordnung des verteilten Berechnungsmodells (engl. *Distributed Computing Environment*).

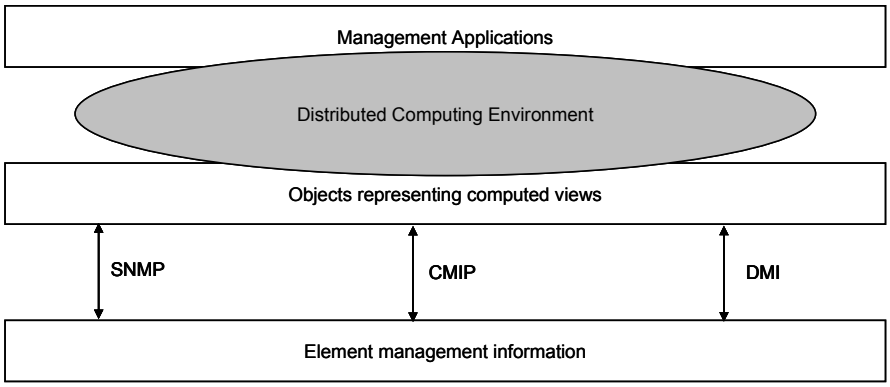

Abbildung 7 Konzeptionelle Architektur nach [An99]

Die in der Arbeit vorgestellte Implementierung des Ansatzes wird auf Basis von *Java RMI* realisiert. Die Implementierung der einzelnen Klienten (*Management Applications*) des Ansatzes sind im Wesentlichen einfache Web Browser zur Darstellung von HTML-Code sowie der Fähigkeit, Java-Code direkt zu interpretieren. Dies ermöglicht den Klienten zum einen, die von den verteilten Objekten aggregierte Managementinformation darzustellen sowie direkt von den einzelnen Objekten bereitgestellte erweiterte Funktionen aufseiten der Klienten auszuführen. Die Implementierung der verteilten Objekte ist komplexer, da neben der einfachen Webserverfunktionalität zur Auslieferung der Webseiten die Logik zur Darstellung eines verteilten Managementinformationsbaumes bereitgestellt werden muss sowie verschiedene Adapter zur Nutzung der unterschiedlichen Managementprotokolle für die Kommunikation mit den einzelnen *Managed Objects*. Es kann festgestellt werden, dass dieser Aufbau im Wesentlichen dem architektonischen Ansatz entspricht, wie er beispielsweise bei WBEM-basierten Systemen verfolgt wird. Hierbei wird mit dem CIMOM (*CIM Object Manager*) eine von der Funktionalität ähnliche Komponente eingeführt.

Bewertung des Ansatzes

Wenngleich die betrachtete Arbeit einige wesentliche Grundprinzipien bezüglich eines verteilten und auf lose gekoppelten, wiederverwendbaren Managementdiensten basierenden Ansatzes motiviert, sind kritische Anforderungen nicht erfüllt. So ist beispielsweise nicht ersichtlich, wie sich der in der in der Arbeit vorgeschlagene Entwicklungsansatz in einen bestehenden Entwicklungsansatz eingliedern lässt. Für die Beschreibung der entstandenen Entwurfsartefakte (Architekturmodell, systematischer Aufbau) werden keine bekannten standardisierten Modellierungssprachen eingesetzt. Die praktische Anwendbarkeit des Ansatzes wird somit zumindest teilweise beeinträchtigt (Anforderung 1.1). Der Entwurf der einzelnen Elemente der verteilten Architektur ist vollständig unabhängig von konkreten Managementwerkzeugen, was eine Integration bestehender Werkzeuge (z. B. bestehende Überwachungssysteme zur Nutzung der vorhandenen Überwachungsinformation) wesentlich vereinfachen würde (Anforderung 1.2). Dem Entwurf liegt kein Analysemodell der zu unterstützenden Domäne zugrunde, weiterhin ist nicht direkt ersichtlich, woher die einzelnen Managementfunktionen der verteilten Objekte motiviert werden. Ein Metamodell der Domäne zur Analyse und zum Entwurf wird nicht eingesetzt (Anforderung 1.3). Die Wiederverwendbarkeit der entstandenen Managementdienste wird zwar motiviert und als wesentliches Entwurfsziel ausgegeben, jedoch ist

nicht ersichtlich, inwiefern die präsentierte Lösung wirklich in weiteren Szenarien eingesetzt werden kann. Die Integration einer Metrik zur Beurteilung der Wiederverwendbarkeit der entstandenen Managementdienste ist nicht Gegenstand der Arbeit (Anforderung 1.4).

Aschemann, Hasselmeyer, 2001: A Loosely Coupled Federation of Distributed Management Services

Eine der ersten Arbeiten mit dem Ansatz, eine auf den Prinzipien der dienstorientierten Architektur basierende Managementplattform auf Basis der Nutzung bestehender Web-Technologien zu erschaffen, wird im Jahre 2001 von Aschemann und Hasselmeyer veröffentlicht [AH01]. Die Motivation für den Ansatz leitet sich aus der Feststellung ab, dass das Management verteilter Systeme bislang hauptsächlich von umfassenden, jedoch monolithischen Managementsystemen bestimmt wird. Der Nachteil, der sich hieraus ergibt, kann in einer schwierigen Integration unterschiedlicher Managementsysteme beim gleichzeitigen Einsatz heterogener Infrastrukturkomponenten gesehen werden. Diese können beispielsweise aus der Anforderung erwachsen, (Teil-)Aktivitäten innerhalb von Managementprozessen zu automatisieren, wodurch die Unterstützung durch Workflow-Systeme notwendig wird. Gerade aber die Automatisierung wird als möglicher Ansatz angesehen, der gestiegenen Komplexität beim Betrieb vernetzter Infrastrukturen zu begegnen und manuelle oder fehleranfällige Arbeitsschritte in Managementprozessen durch die Unterstützung automatischer Ablaufsteuerungen zu realisieren.

Hierbei sind zwei Aspekte von besonderer Bedeutung. Zum einen muss Fachfunktionalität in Form von Managementdiensten bereitgestellt werden, die die eigentlichen Managementaufgaben unterstützt. Diese Managementdienste werden unterstützt von Diensten, die den Betrieb des Managementsystems überhaupt ermöglichen (Managementinfrastrukturdienste). Hierzu gehören beispielsweise zentrale Lokalisationsspeicher, um Managementdienste in einem verteilten Managementsystem auffinden zu können. Wichtiger Bestandteil der Architektur ist daher ein Konfigurationsdienst, der nicht nur Konfigurationsdaten der einzelnen Managementobjekte verwaltet, sondern auch Konfigurationsdaten der Managementdienste. Darüber hinausgehend unterstützt der Konfigurationsdienst aktiv alle Managementaufgaben im Konfigurationsmanagement.

Der von den Autoren vorgeschlagene Entwurfsprozess wird aus der Analyse von zu realisierenden Anwendungsfällen getrieben. In der Arbeit wird anhand der betrieblichen Fragestellungen in Bezug auf einen vernetzten Bürodrucker von typischen Managementaufgaben auf zu unterstützende Managementdienste geschlossen. Die Instrumentierung erfolgt mittels SNMP, weshalb auch ein Entwurf von zwei SNMP-Diensten vorgeschlagen wird (*SNMP Trap Service*, *SNMP Gateway Service*). Weiterhin wird die Abbildung von protokollabhängigen Kommunikationsansätzen mittels eines Protokolldienstes benötigt sowie ein zentraler Dienst zur Speicherung von Konfigurationsinformationen (*Configuration Service*).

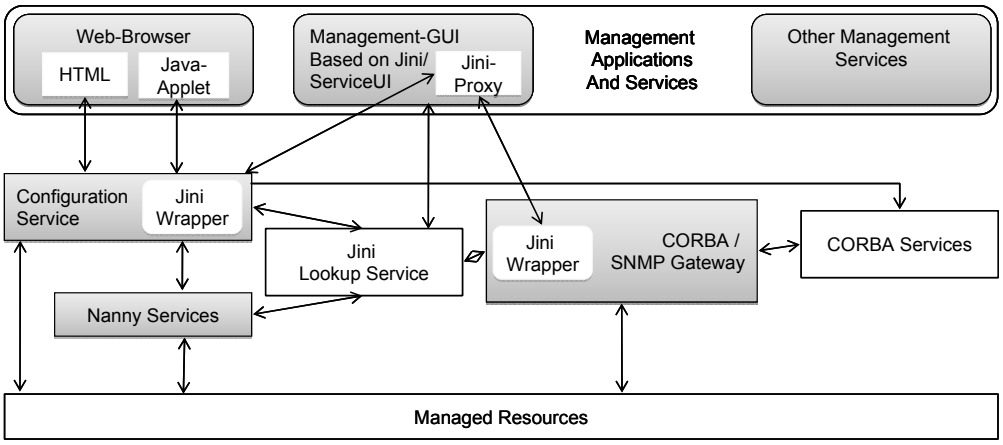

Abbildung 8 Prinzipielle Architektur für verteiltes Management nach [AH+01]

Die vorgeschlagene Architektur (siehe Abbildung 8) wird mittels Jini implementiert. Dieser auf Java basierende Ansatz für die Konstruktion von verteilten Softwarearchitekturen zeichnet sich den Autoren zufolge vor allem dadurch aus, dass die Softwarekomponenten dieser Architektur eine automatische Registrierung an einem zentralen Dienstverzeichnis selbstständig vornehmen. Dieser Aspekt wird für die betrachtete Domäne als wichtig erachtet, da aufgrund der Eigenschaften von Infrastrukturressourcen (eventuell nur zu bestimmten Zeiten betriebsbereit) eine selbstständige Registrierung in einem zentralen Dienstverzeichnis den betrieblichen Aufwand minimiert. Die technische Implementierung dieser Eigenschaft wird im Protokollstapel durch Nutzung von Multicast-Verbindungen auf Internetprotokollebene (Internetprotokollfamilie) bzw. Paketvermittlungsebene (OSI-Protokollstapel) realisiert.

Bewertung des Ansatzes

Der von den Autoren vorgestellte Ansatz beschreibt den durchgehenden Entwurf einer Architektur für Managementdienste. Der Fokus wird auf den Entwurf zentraler Elemente dieser Architektur, wie beispielsweise den Konfigurationsdienst, gesetzt. Zunächst gehen die Autoren von konkreten Szenarien zur Ableitung von spezifischen Anforderungen aus, wobei die Trennung des Vorgehens in Analyse- bzw. Entwurfsphase sowie Implementierungsphase ersichtlich wird, wenngleich das Vorgehen nicht explizit dargelegt und eine abstrahierte Betrachtung der eingesetzten Modellierungsansätze nicht diskutiert wird. Theoretisch scheinen jedoch standardisierte Modellierungssprachen einsetzbar zu sein, weshalb das Vorgehen in der Praxis umsetzbar zu sein scheint (Anforderung 1.1). Der Entwurf der in der Arbeit vorgestellten Managementdienste wird auf Basis der präsentierten Szenarien durchgeführt. Ein Ausgangspunkt auf Basis allgemeiner Anforderungen, die z. B. aus den Standards der ISO motiviert werden (ISO7498 [ISO89], ISO/IEC20000-1:2005 [ISO05]), ist nicht vorgesehen. Insgesamt sind die Grundzüge eines domänenspezifischen Entwurfes von Managementdiensten erkennbar, es fehlt jedoch eine abstrahierende Betrachtung in Form eines integrierten Analyse- und Entwurfsvorgehens (Anforderung 1.3). Die vorgestellten Managementdienste sind aufgrund des szenariogetriebenen Ansatzes zunächst unabhängig von konkreten Managementwerkzeugen (Anforderung 1.2). Durch den Einsatz von Jini zu

Implementierung eines verteilten Managementsystems und dem Szenario-getriebenen Entwurf ist zu erwarten, dass die entworfenen Managementdienste wiederverwendet werden können. Da aber im Umkehrschluss der Entwurf bereits Implementierungsdetails der zugrunde liegenden Technologie aufweist, wird die Wiederverwendbarkeit offensichtlich zunächst auf Erweiterungen auf Basis des gleichen Ansatzes eingeschränkt. Die Evaluation der Wiederverwendbarkeit auf Basis einer dedizierten Metrik ist nicht Gegenstand der Arbeit und fehlt daher gänzlich (Anforderung 1.4).

Xiao et al., 2009: WS-CHMA: A Composite-pattern based Hierarchical WS-Management Architecture

Der Einsatz von Unternehmensanwendungen erfordert die Bereitstellung verteilter Ressourcen, die als Gesamtheit betrieben und verfügbar sein müssen [XL+09]. Um eine Unterstützung bezüglich des Betriebes dieser Ressourcen auf Basis verteilter Managementagenten zu erzielen, erhält die Adaption von *WS-Management*-basierten [DMTF-WS-Management]Ansätzen verstärkt Zuspruch, nicht auch zuletzt aufgrund der flexibleren Ausgestaltung einzelner Managementagenten auf Basis von Webservices. Während die Grundlage einer auf den Prinzipien der dienstorientierten Architektur basierenden Managementlösung mittels des Einsatzes von Ansätzen wie beispielsweise *WS-Management* umgesetzt werden kann, argumentieren die Autoren jedoch, dass die Nutzes dieses Prinzips beim Betrieb vieler tausender Ressourcen eines großen Rechenzentrums eine Aggregation bezüglich einzelner dedizierten Ressourcen notwendig macht.

Der von den Autoren vorgestellte Lösungsansatz zielt auf eine logische Gruppierung von Ressourcen auf Basis des Kompositumentwurfsmusters [GH+08]. Abbildung 9 stellt den strukturellen Aspekt dieses architektonischen Ansatzes in Form eines UML-Klassendiagramms dar.

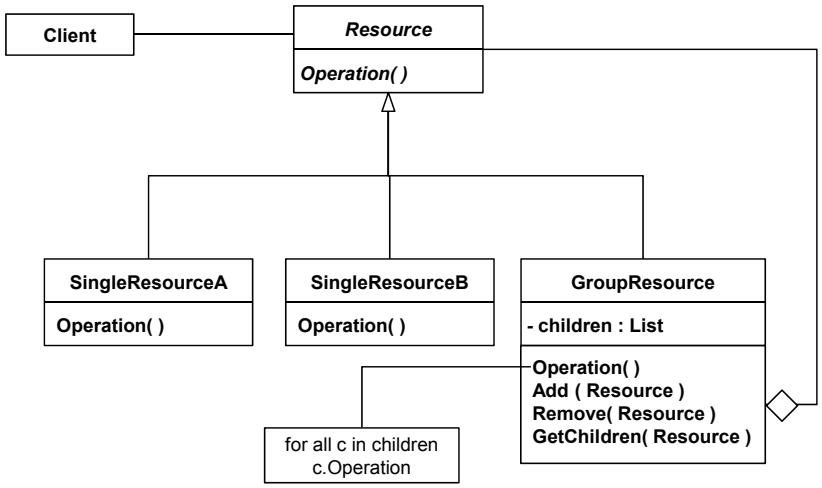

Abbildung 9 Komposition von *Managed Resources* in WS-Management nach [XL+09]

Ersichtlich ist die Trennung des Elementes Ressource in eine abstrakte Oberklasse `Resource` sowie in konkrete Unterklassen `SingleResource` bzw. `GroupResource`. Managementoperationen können sowohl von einzelnen Ressourcen als auch von logischen Ressourcengruppierungen ausgeführt werden, darüber hinausgehend besitzt die Klasse `GroupResource` weiterhin Operation

zum Hinzufügen und Entfernen von einzelnen Ressourcen zur logischen Gruppe sowie eine Operation zur Auflistung aller enthaltenen Ressourcen dieser Ressourcengruppe. Eine Managementoperation einer Ressourcengruppe wird auf alle enthaltenen Ressourcenelemente dieser Ressourcengruppe ausgeführt.

Der vorgestellte Ansatz zielt vor allem auf betriebliche Fragestellungen bezüglich technischer Managementaufgaben wie beispielsweise Überwachung und Steuerung von Ressourcen eines vernetzten IT-Systems ab. Eine Implementierung des architektonischen Ansatzes wird direkt auf Basis der von *WS-Management* vorgegebenen Managementfunktionen realisiert. Wenngleich der vorgestellte Ansatz einen echten Mehrwert bei der Aggregation und Kaskadenschaltung von vielen einzelnen *WS-Management*-basierten Managementagenten darstellt, stellt sich die Frage, wie eine sinnvolle Aggregation auf Basis automatisierender Verfahren durchgeführt werden kann.

Bewertung des Ansatzes

Der Schwerpunkt der Publikation liegt auf der Betrachtung eines architektonischen Ansatzes, um die Aggregation vieler einzelner *WS-Management*-basierter Managementagenten zu gestalten. Eine Einordnung des eigenen Ansatzes in einen übergeordneten Softwareentwicklungsprozess wird von den Autoren nicht diskutiert. Da der vorgestellte Lösungsansatz auf Basis des standardisierten *WS-Managements* umgesetzt wird, kann die praktische Anwendbarkeit hinsichtlich der gestellten Anforderung 1.1 angenommen werden. Der Entwurf der Managementdienste geschieht nicht auf Basis einer Analyse der Domäne (Anforderung 1.3). Die Unabhängigkeit von konkreten Managementwerkezeugen der mit diesem Ansatz erstellten Managementdienste ist nur insofern gegeben, als das die zugrunde liegende Implementierung auf *WS-Management* basiert und somit lediglich technische Aspekte beim Betrieb von IT-Infrastrukturen abdeckt. Eine vollständige Umsetzung von sowohl technischen als auch organisatorischen Aspekten wird nicht diskutiert. Die Bewertung bezüglich Anforderung 1.2 erfolgt daher neutral. Da der Entwurf der Dienste nicht auf Basis eines Domänenmodells stattfindet und der Entwurf nicht in einen übergeordneten Entwicklungsprozess eingebettet ist, ist fraglich, ob die mit diesem Ansatz umgesetzten Managementdienste einen hohen Grad an Wiederverwendbarkeit aufweisen. Eine Diskussion hinsichtlich der Wiederverwendbarkeit der eigenen Lösung findet nicht statt; hierzu kann demnach keine Aussage getroffen werden (Anforderung 1.4).

Lu et al., 2008 und 2009: verschiedene Arbeiten

Die Motivation für den Einsatz einer auf Webservices basierenden dienstorientierten Architektur leitet sich den Autoren von [LW+08] zufolge aus der Beobachtung ab, dass monolithische Managementanwendungen aufgrund der steigenden Heterogenität der zu betreibenden Infrastrukturelemente nicht mehr in dem Maße den notwendigen funktionalen Anforderungen gerecht werden, wie es der steigende Bedarf an Flexibilität beim Einsatz von Informationstechnologie notwendig machen würde. Hieraus leitet sich die Feststellung ab, dass leichtgewichtige und standardisierte Schnittstellen zu Managementagenten entwickelt werden müssen, sodass relativ einfach auf Managementinformation zugegriffen werden kann, aber auch neue Anforderungen bezüglich des Betriebes auf Basis der bestehenden Agenten umgesetzt werden können.

Ein möglicher Ansatz, diese Anforderung umzusetzen, wird von Lu et al. in einer Reihe unterschiedlicher Arbeiten veröffentlicht [LF+08, LL+08, LW+08, LW+09, LW+09b]. Da diese Arbeiten thematisch stark miteinander verwandt sind und letztlich alle den Einsatz einer auf WSDM-[OASIS-MUWS] oder *WS-Management*-basierenden [DMTF-WS-Management] Implementierung betrachten, werden diese Arbeiten im Folgenden gemeinsam betrachtet, diskutiert und bewertet.

In [LF+08, LW+09, LW+09b] wird der Einsatz einer den Prinzipien von *WS-Management*-basierenden zugrunde liegenden verteilten Managementarchitektur untersucht. Die Motivation folgt der Feststellung, dass gerade beim Betrieb von Netzwerk- und Systemressourcen einer IT-Infrastruktur die von den einzelnen Ressourcen angebotenen Managementschnittstellen meist eng an technologische oder systemspezifische Einschränkungen gebunden sind. Eine abstrahierte Zugriffsschicht, die Funktionalität unabhängig von konkreten Produktschnittstellen anbietet, wäre wünschenswert.

Die von den Autoren dargestellte Drei-Schichten-Architektur hat zum Ziel, den Einsatz bestehender Managementschnittstellen wie beispielsweise WMI (*Windows Management Instrumentation*)zu ermöglichen. Darauf aufbauend wird eine einheitliche Zugriffsschicht auf diese einzelnen *Managed Resources* auf Basis einer *WS-Management*-Implementierung (*wiseman*) geschaffen. Managementfunktionen können anschließend von beliebigen Managementanwendungen genutzt werden.

In [LL+08, LW+08] wird eine dreistufige Architektur ähnlich zu der in [LF+08, LW+09, LW+09b] eingeführten Architektur präsentiert. Auf Ebene des Basiszugriffs auf *Managed Resouces* wird die Möglichkeit vorgesehen, durch den Einsatz sogenannter *Management Gateways* einen Webservice-basierten Zugriff auf nicht-Webservice-fähige *Managed Resouces* zu ermöglichen. Die einzelnen Ebenen einer SOA-basierten Managementarchitektur werden auf das von WSDM vorgeschlagene *Middleware*-Modell abgebildet.

Bewertung der Ansätze

Allen Arbeiten von Lu et al. ist gemeinsam, dass der Charakter eines Referenzentwurfes nicht gegeben ist (Anforderung 1.1). Es kommen weder standardisierte Modellierungssprachen zum Einsatz, die eine vergleichende Diskussion zu weiteren Ansätzen auf eine abstrahierte Art und Weise ermöglichen, noch wird der Entwurf der präsentierten Artefakte wie beispielsweise Schnittstellenbeschreibungen, Architekturblaupausen oder Interaktionsabläufen anhand einer nachvollziehbaren Methode beschrieben. Hinsichtlich der Unabhängigkeit von konkreten Managementwerkzeugen kann festgestellt werden, dass eine abstrahierte Beschreibung erfolgt. Dies ist jedoch die logische Konsequenz der Fokussierung der Arbeiten, die dem Einsatz von *WS-Management*- oder WSDM-basierten Ansätzen zugrunde liegt. Es werden jedoch lediglich technische Aspekte betrachtet, vor allem werden Managementfragestellungen bezüglich des Zugriffs auf Infrastrukturressourcen untersucht (Anforderung 1.2). Den Entwürfen liegen keine abstrahierten Domänenmodelle zugrunde. Es ist demnach nicht feststellbar, inwiefern die diskutierten architektonischen Ansätze in einem integrierten Entwicklungsvorgehen genutzt werden können, da ein integriertes Modell für die dienstorientierte Analyse und den dienstorientierten Entwurf fehlt (Anforderung 1.3). Eine Diskussion der erzielten Ergebnisse bezüglich typischer Eigenschaften einer dienstorientierten Softwarelösung, insbesondere der Wiederverwendbarkeit der präsentierten Managementdienste, wird nicht durchgeführt (Anforderung 1.4). Zusammengefasst kann festgestellt werden, dass die von Lu et al.

durchgeführten Arbeiten vor allem bezüglich eines Einsatzes bestehender Standards für den dienstorientierten Zugriff auf technische Managementinformation genutzt werden können. Diese Arbeiten liefern einen Beitrag zur Ausgestaltung einer dienstorientierten Gesamtlösung für die Integration von Managementwerkzeugen.

Schaaf, Brenner, 2008: On Tool Support for Service Level Management: From Requirements to System Specifications

Einer der zentralen Betreiberprozesse zur Wahrung und Sicherstellung einer hohen Qualität der angebotenen IT-Dienste stellt der *Service-Level-Management*-Prozess (SLM-Prozess) dar. Gegenstand und Zielsetzung des SLM-Prozesses ist u. a. die Aushandlung von Dienstleistungsvereinbarungen (DLV, engl. *Service Level Agreement*, SLA), aber auch die Überwachung von IT-Infrastrukturressourcen zur Einhaltung von zugesicherten Qualitätsattributen. Der SLM-Prozess stellt daher eine wichtige Schnittstelle zu Kunden von IT-Dienstleistern dar, weshalb die Beherrschung dieses Prozesses einen unmittelbaren Einfluss auf den Erfolg eines IT-Dienstleisters hat. Eine (Teil-) Automatisierung einzelner Ablaufaktivitäten dieses Prozesses ist daher wünschenswert.

Vom Blickpunkt der den Prozess unterstützenden Managementwerkzeuge wird eine querschnittliche Nutzung durch unterschiedliche Fachabteilungen eines IT-Dienstleisters sichtbar. Da diese einzelnen Werkzeuge in der Regel unabhängig voneinander entwickelt und weitergepflegt wurden, ist eine Integration in eine gemeinsame Plattform zur Unterstützung funktionaler Anforderungen des Prozesses nicht mittelbar zu bewerkstelligen. Erschwerend kommt hinzu, dass der Prozess aufgrund seiner organisatorischen und strukturellen Komplexität schwieriger zu automatisieren ist, als es beispielsweise der *Incident-Management*- oder *Problem-Management*-Prozess ist [Br06].

Zur Begegnung dieser Problemstellungen stellen Schaaf und Brenner in [SB08] eine konstruktive Methode vor, um am Beispiel des SLM-Prozesses die Umsetzung von fachlichen Anforderungen auf eine unterstützende Architektur durchzuführen. Die vorgestellte Methode orientiert sich an dem von der OMG beschriebenen modellgetriebenen Vorgehen, über die Spezifikation von plattformunabhängigen Modellen (engl. *Platform Independent Model*, PIM) und der Transformation auf entsprechende plattformspezifischen Modelle (engl. *Platform Specific Model*, PSM) zu einer lauffähigen Softwarelösung zu kommen. Ausgangspunkt für die Methode ist die analytische Betrachtung von Szenarien sowie informellen Anforderungen an die Softwarelösung.

Die Autoren zerlegen die komplexe Problemstellung in vier Teilbereiche, die an bekannten Prinzipien der Managementarchitektur mit ihren vier Teilmodellen Organisationsmodell, Funktionsmodell, Informationsmodell sowie Kommunikationsmodell ausgerichtet ist. Für jede dieser vier Teilaspekte wird vorgeschlagen, ein durchgehendes MDA-basiertes Entwicklungsvorgehen voranzutreiben.

Die Analyse von Anforderungen an die zu entwerfende Softwarelösung geschieht auf Basis von textuell beschriebenen Szenarien. Darauf aufbauend werden Anwendungsfälle formuliert, die letztlich die Grundlage für die Analyse der vier unterschiedlichen Teilmodelle hinsichtlich der Prozessperspektive bilden. Im Organisationsmodell werden Rollen und organisatorische Grenzen der Domäne identifiziert, im Funktionsmodell wird eine informelle Ablaufbeschreibung einzelner

Ablaufaktivitäten mit der Verknüpfung der unterschiedlichen Anwendungsfälle erzielt, im Informationsmodell werden die Grenzen einzelner Prozessartefakte identifiziert sowie im Kommunikationsmodell letztlich die Schnittstellen zu weiteren Managementbereichen festgelegt.

Der nächste Schritt der Entwurfsmethode fokussiert die Spezifikation von plattformunabhängigen Vorgaben für ein System zur Realisierung der gestellten funktionalen Anforderungen. Die einzelnen Teilmodelle werden verfeinert und teilweise in Pseudo-Code-Schreibweise angegeben. Die Abbildung der plattformunabhängigen Modelle auf plattformspezifische Modelle ist nicht Gegenstand der Arbeit.

Bewertung des Ansatzes

Der on [SB08] vorgestellte Entwicklungsansatz stellt eine integrierte Methode vor, um durch eine Reihe von Modelltransformationen die funktionalen Anforderungen eines Managementprozesses auf eine unterstützende Werkzeuglandschaft abzubilden. Die Arbeit legt den Fokus auf die Beschreibung der Entwurfsartefakte der CIM- und PIM-Ebenen eines modellgetriebenen Entwicklungsansatzes für den Entwurf von Managementwerkzeugen.

Durch die Betrachtung von Szenarien und explizite Formulierung von Anforderungen an die zu entwerfende Softwarelösung kann das Verfahren einfach nachvollzogen und daher insgesamt in der Praxis angewandt werden. Wenngleich die Autoren das Entwicklungsvorgehen, angelehnt an die unterschiedlichen Abstraktionsstufen der MDA, skizzieren, wird auf den Einsatz standardisierter Modellierungssprachen, verzichtet. Insbesondere die Modellierung von Prozessabläufen erfolgt informell, die Modellierung von Schnittstellen der entworfenen Softwarelösung wird auf Ebene der plattformspezifischen Modelle nicht diskutiert. Die Autoren weisen jedoch auf den möglichen Einsatz dieser Modellierungssprachen hin. Insgesamt wird die Anforderung 1.1 daher neutral bewertet. Das Entwicklungsvorgehen wird von funktionalen Prozessanforderungen getrieben und ist daher unabhängig von konkreten Managementwerkzeugen (Anforderung 1.2). Ein der Analyse und dem Entwurf zugrunde liegendes abstrahiertes Domänenmodell ist nicht erkennbar (Anforderung 1.3). Damit wird insbesondere die Integration bestehender Werkzeuge erschwert, da durch den vorwärtsgetriebenen Entwurf die Semantik der einzelnen Entwurfselemente nicht eindeutig definiert werden kann. Erschwerend kommt hinzu, dass nicht klar ersichtlich ist, wie die Transformationen konkret durchgeführt werden. Aufgrund der abstrahierten Betrachtungen innerhalb der Publikation wäre eine Umsetzung auf Webservice-basierte Architekturen durchaus denkbar, wird von den Autoren jedoch nicht explizit diskutiert. Eine abschließende Bewertung der eigenen Lösung vor dem Hintergrund der Wiederverwendbarkeit der entstandenen Entwurfsartefakte wird nicht durchgeführt.

3.2.2 Ansätze aus dem Bereich der dienstorientierten Softwareentwicklung

Aus der Untersuchung von bestehenden Arbeiten bezüglich des Entwurfes von Managementdiensten wird ersichtlich, dass diese Arbeiten in der Regel die grundlegende Anforderung der Wiederverwendbarkeit der entwickelten Lösung nicht explizit betrachten bzw. keine kritische Evaluation der entwickelten Lösung auf Basis formalisierter und nachvollziehbarer Ansätze durchführen kann. Eine Bewertung dieser Arbeiten bezüglich dieser Anforderung ist demnach nicht ohne weiteres möglich. Diese Tatsache kann darauf zurückgeführt werden, dass der Entwurf

dienstorientierter Softwarelösungen für die Domäne IT-Management generell nicht auf Basis von bekannten Arbeiten zur Identifikation und zum Entwurf dienstorientierter Softwarelösungen durchgeführt wird. Daher werden im Folgenden Arbeiten untersucht, die nicht direkt innerhalb der Domäne IT-Management angesiedelt sind, jedoch grundlegend den Entwurf von dienstorientierten Softwarelösungen fokussieren. Die hier vorgestellten Arbeiten diskutieren den Entwurf dienstorientierter Softwarelösungen von einem abstrahierten Standpunkt und sind in der Regel unabhängig von konkreten fachlichen Bezügen motiviert. Die hieraus gewonnen Erkenntnisse stellen die Grundlage dar, um das in Kapitel 4 vorgestellte Metamodell der Domäne IT-Management in einem strukturierten Entwicklungsprozess einsetzen zu können.

Arsanjani et al., 2008: SOMA: A method for developing service-oriented solutions

Der Einsatz dienstorientierter Architekturen stellt mittlerweile einen etablierten Ansatz dar, um eine IT-Unterstützung aus der Perspektive fachlicher Anforderungen zu motivieren, zu gestalten und umzusetzen. Die Implementierung dienstorientierter Architekturen auf Basis von Webservice-Standards kann dabei helfen, die wesentlichen Prinzipien einer dienstorientierten Architektur zu implementieren. Hierzu ist jedoch ein geeigneter Entwicklungsansatz erforderlich, der die durch den Einsatz von Webservices benötigten architektonischen Entwurfsentscheidungen berücksichtigt und in einem integrierten Entwicklungsvorgehen an fachlichen Anforderungen ausrichtet. Die Modellierung von Diensten stellt einen kritischen Erfolgsfaktor dar, um eine erfolgreiche und qualitativ hochwertige dienstorientierte Softwarelösung zu entwickeln. In [AG+08] stellen Arsanjani et al. ein Vorgehen vor, um die Modellierung dienstorientierter Lösungen in den Kontext einer durchgängigen und einheitlichen Methode zu stellen. *Service-Oriented Modeling And Architecture* (abgekürzt: SOMA) ist ein Ansatz, der explizit für den Entwurf von dienstorientierten Architekturen für verschiedene Domänen hin ausgelegt ist. Der Ansatz wurde maßgeblich von der Firma IBM über mehrere Jahre hinweg entwickelt und untergliedert sich insgesamt in sieben verschiedene Phasen.

In der ersten Phase (*Business Modeling and Transformation*) findet eine Modellierung der Geschäftsabläufe statt. Die zweite Phase (*Solution Management*) betrachtet verschiedene mögliche Lösungsansätze für die im jeweils betrachteten Szenario auftretenden Fragestellungen.

Zentrale Idee des Ansatzes ist die Identifikation (*Identification*) möglicher Dienste, in der drei wesentliche Aspekte einer dienstorientierten Architektur fokussiert werden: Dienste, Komponenten (Implementierung der Dienste) und Abläufe (zur späteren Orchestrierung der Dienste). Insgesamt kommen in dieser Phase drei verschiedene Ansätze zum Einsatz:

(1) *Goal-Service Modeling* (GSM) definiert Geschäftsziele und darauf aufbauend eine hierarchische Verfeinerung in Unterziele, die zur Grundlage einer Identifikation von realisierbaren und erforderlichen Diensten genutzt werden.

(2) *Domain Decomposition*: Zentraler Aspekt ist die Eingrenzung, Zerlegung und Modellierung der jeweiligen betrachteten Domäne.

(3) *Asset Analysis* betrachtet die Untersuchung der bestehenden Softwaresysteme (engl. *Asset*), die eine Rolle in den zu unterstützenden Geschäftsprozessen spielen und daher zur Realisierung der identifizierten Dienste genutzt werden können oder müssen.

Im Anschluss an die Identifikationsphase folgt die Entwurfsphase (*Specification*). Basierend auf den identifizierten Dienstkandidaten erfolgt eine vollständige Spezifikation der weiter zu implementierenden Dienste, wobei noch keine technischen Entwurfsentscheidungen getroffen werden. Diese werden in den darauf folgenden Phasen (*Realization* und *Implementation*) ergänzt, um somit letztlich eine in einen gesicherten Betrieb überführbare Softwarelösung zu konstruieren.

Bewertung des Ansatzes

Die von Arsanjani et al. vorgestellte Methode [AG+08] ist ein vollständiger Ansatz, um fachliche Anforderungen auf eine dienstorientierte Architektur abzubilden. Neben der Analyse- und Entwurfsphase werden auch Implementierung sowie Test und Inbetriebnahme von der Methode betrachtet. Die Methode ist insgesamt nachvollziehbar, der Einsatz standardisierter Modellierungssprachen ist möglich, wird in der Veröffentlichung jedoch nicht explizit beschrieben. Insgesamt kann die Methode eingesetzt werden, einen Referenzentwurf von Diensten zu erstellen (Anforderung 1.1). Da die Methode unabhängig von der Domäne IT-Management ist, wird die Anforderung hinsichtlich der Unabhängigkeit von konkreten Managementwerkzeugen nicht bewertet (Anforderung 1.2). Die Methode sieht eine dedizierte Phase zur Dekomposition der zu unterstützenden Domäne vor. Dieser Entwicklungsschritt wird jedoch nicht auf Basis eines einheitlichen Metamodells der Domäne durchgeführt, weshalb eine durchgehende Einhaltung semantischer Bedeutung innerhalb der gesamten Methode schwierig ist, wenn nicht geeignete Maßnahmen durchgeführt werden. Insgesamt wird diese Anforderung daher neutral bewertet (Anforderung 1.3). Die Methode sieht einen speziellen Methodenschritt vor, das Ergebnis der Spezifikationsphase zu überarbeiten und zu refaktorisieren. Wenngleich keine gesonderte Betrachtung innerhalb der Arbeit bezüglich der Anforderung der Wiederverwendbarkeit stattfindet, kann dieser Aspekt in diesem Methodenschritt durchgeführt werden. Da dieser Aspekt in der Arbeit insgesamt keine Beachtung findet, wird diese Anforderung ebenfalls neutral bewertet (Anforderung 1.4).

Erl, 2006 bis 2009: verschiedene Arbeiten

Die von Erl in [Er06, Er08, Er08b, Er09] vorgestellten Arbeiten setzen thematisch allesamt bei der Fragestellung an, welche Änderungen beim Entwurf dienstorientierter Softwaresysteme im Gegensatz zu bekannten Ansätzen, wie beispielsweise der objektorientierten oder komponentenorientierten Softwareentwicklung, zu beachten sind. Aufgrund der Vielfältigkeit und umfassenden Darstellung der einzelnen Aspekte werden insgesamt viele wesentliche und grundlegende Fragestellungen auf eine pragmatische Art und Weise diskutiert. Hinsichtlich eines ganzheitlichen Prozesses zur Entwicklung einer dienstorientierten Softwarelösung führt Erl in [Er06] verschiedene Sichten auf einen Entwurfsprozess ein, die ausgerichtet an der Zielsetzung des Entwurfes eines dienstorientierten Softwaresystems unterschiedliche Aspekte innerhalb des Entwurfsprozesses einordnen. Hervorzuheben sind die Analysephase und die Entwurfsphase, die jeweils spezielle Verfeinerungen bezüglich der Dienstorientierung aufweisen. In der dienstorientierten Analyse (engl. *Service-oriented Analysis*) wird schwerpunktmäßig ein formaler Modellierungsprozess etabliert, der als Ergebnis,

fachfunktionale Anforderungen für den späteren Entwurf von Diensten zu spezifizieren und in Form von Dienstkandidaten darstellt. Im dienstorientierten Entwurf (engl. *Service-oriented Design*) werden diese Dienstkandidaten detaillierter spezifiziert.

Die dienstorientierte Analyse unterteilt sich Erl zufolge in drei Teilschritte (siehe auch Abbildung 10). Im Schritt **Analyseziel festlegen** wird eine Eingrenzung auf das eigentliche Ziel der Analysephase durchgeführt. Hierbei werden Anforderungen, die sich z. B. aus der gewünschten Automatisierung fachlicher Aspekte ergeben, erfasst und festgehalten. Im darauffolgenden Schritt **Automatisierungsziele identifizieren** werden auf Basis der bestehenden Systeme etwaige Unterschiede zwischen den im ersten Schritt festgestellten Analysezielen untersucht, um somit die Menge von umzusetzenden Diensten einzugrenzen. Den Abschluss der dienstorientierten Analyse bildet der Schritt **Dienstkandidaten modellieren**. Da in diesem Schritt das eigentliche Ergebnisartefakt der dienstorientierten Analyse erzeugt wird, wird dieser Schritt detaillierter betrachtet.

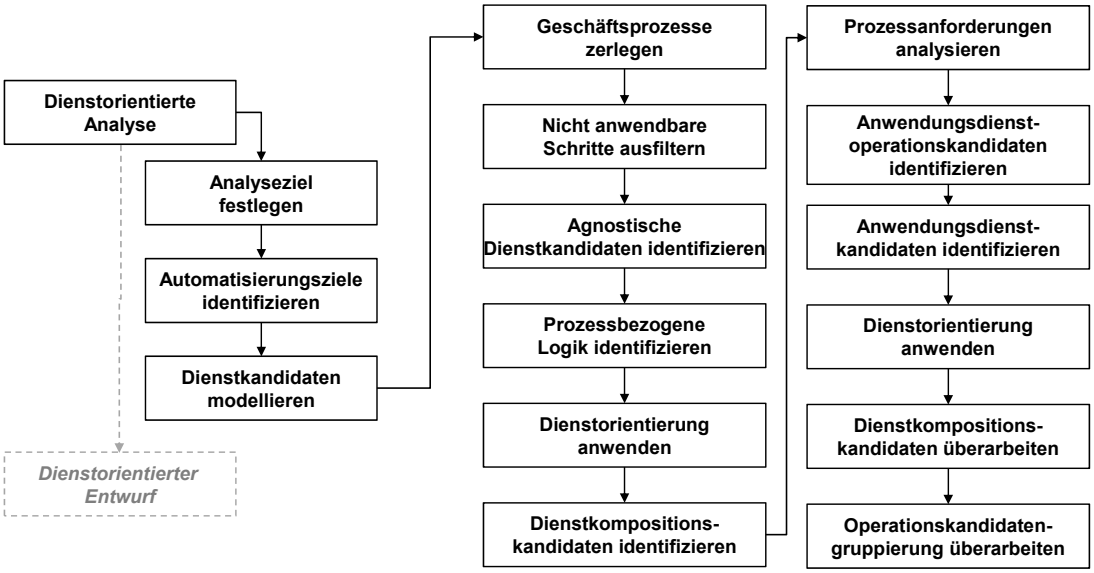

Abbildung 10 Dienstorientierte Analyse nach [Er06]

Insgesamt schlägt Erl zwölf verschiedene Aktivitäten innerhalb des dritten Schrittes in der dienstorientierten Analyse vor, die jedoch nicht alle zwangsläufig durchgeführt werden müssen (**Nicht anwendbare Schritte ausfiltern**). Ausgangspunkt ist eine Dekomposition von Geschäftsprozessen (**Geschäftsprozesse zerlegen**), die auf Basis von dokumentierten Abläufen die Grundlage zur Identifikation von durch Dienste umzusetzende Prozessaktivitäten bildet. Im Schritt **Agnostische Dienstkandidaten identifizieren** wird eine erste grobgranulare Identifikation verschiedener möglicher Dienste ermittelt. Darauf folgend wird **Prozessbezogene Logik identifiziert** und in eigene Dienstkandidaten transferiert. Es folgt eine erste **Anwendung dienstorientierter Prinzipien** auf die bislang entstandenen Artefakte. Der sechste Schritt sieht vor, Kandidaten für eine mögliche **Dienstkomposition zu identifizieren**. Die bis zu diesem Zeitpunkt ermittelten Dienstkandidaten

können in eine erste Klassifikation überführt werden (fachbezogene Dienstkandidaten, anwendungsbezogene Dienstkandidaten), wobei letztere im Schritt **Prozessanforderungen analysieren** weiter verfeinert und analysiert werden. Dies bildet die Grundlage, um in den beiden folgenden Schritten zunächst Kandidaten für mögliche Dienstoperationen von anwendungsbezogenen Diensten zu ermitteln (**Anwendungsdienstoperationskandidaten identifizieren**) und diese dann zu Anwendungsdienstkandidaten zu komponieren (**Anwendungsdienstkandidaten identifizieren**). Es erfolgt eine neuerliche Überarbeitung der bislang erstellten Artefakte hinsichtlich der Prinzipien der **Dienstorientierung**. Anschließend werden die **Dienstkompositionskandidaten überarbeitet** sowie die **Gruppierungen von Operationskandidaten** zu Dienstkandidaten durchgeführt.

Die dienstorientierte Entwurfsphase unterteilt sich Erl zufolge in insgesamt fünf verschiedene Schritte, die analog zur Phase der dienstorientierten Analyse in weitere Teilschritte verfeinert werden kann. In Abbildung 11 sind die Schritte des dienstorientierten Entwurfes dargestellt.

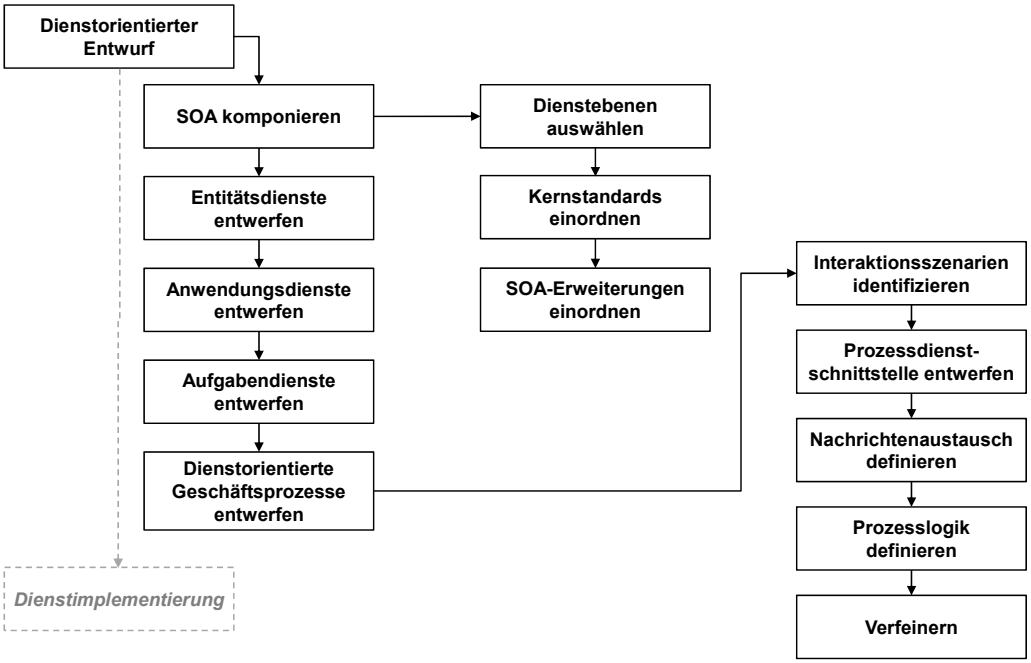

Abbildung 11 Dienstorientierter Entwurf nach [Er06]

Ausgangspunkt in der Entwurfsphase ist die Festlegung auf verbindliche Standards zur späteren Umsetzung der dienstorientierten Architektur. Dieser Schritt (**SOA komponieren**) ist insofern nachvollziehbar und schlüssig, da durch die spätere Umsetzung auf der Grundlage dedizierter Technologien bestimmte Entwurfsentscheidungen eine günstige Auswirkung auf den weiteren Verlauf des Entwicklungsprozesses haben können. Den Abschluss der dienstorientierten Analyse bildet Erl zufolge eine Überarbeitung der Geschäftsprozesse bezüglich der Ergebnisse des dienstorientierten Entwurfes (**Dienstorientierte Geschäftsprozesse entwerfen**). Im Kern des dienstorientierten Entwurfes stehen drei gesonderte Abschnitte für den speziellen Entwurf von Entitäts-, Anwendungs- und Aufgabendiensten.

Der **Entwurf von Entitätsdiensten** beinhaltet zunächst eine Durchsicht eventuell **bestehender Entitätsdienste** zur Umsetzung der gewünschten Anforderung. Werden keine Übereinstimmungen gefunden, wird ein **Entitätsschema definiert** (z. B. ein konkretes Datenmodell) sowie darauf aufbauend eine **abstrakte Dienstschnittstelle**. Auf dieses Ergebnis werden die Prinzipien der **Dienstorientierung angewandt**. Die herausgearbeitete **Dienstschnittstelle wird standardisiert**, der Dienstentwurf wird nun, falls nötig **erweitert**, indem **weitere benötige Dienste identifiziert** werden.

Die einzelnen Schritte für den **Entwurf von Anwendungsdiensten** sehen prinzipiell vergleichbar aus, jedoch wird anstelle der Definition eines Entitätsschemas der **Anwendungskontext** des zu entwerfenden Anwendungsdienstes festgelegt. Weiterhin wird eine verbindliche Aktivität durchgeführt, die zum Ziel hat, **mögliche Funktionen** eines Anwendungsdienstes vorausschauend dem Dienstentwurf hinzuzufügen.

Ebenso vergleichbar sind auch die einzelnen Schritte beim **Entwurf von Anwendungsdiensten**. Hier wird von der Definition der **Workflow-Logik** ausgegangen. Diese bildet den Übergang zur Definition der abstrakten Dienstschnittstelle, woran sich die bekannten Aktivitäten (Dienstorientierung anwenden, Dienstschnittstelle standardisieren, weitere Dienste identifizieren) anschließen.

Die Unterteilung in verschiedene Entwurfsschritte bezüglich des Entwurfes unterschiedlicher Typen von Diensten scheint schlüssig, ist aber beispielsweise in den Arbeiten von Arsanjani et al. [AG+08] nicht zu erkennen.

Als eines der wesentlichen Charakteristika bezüglich des Entwurfes dienstorientierter Softwaresysteme findet sich bei Erl die Anforderung der Wiederverwendbarkeit. In [Er08] wird die Wiederverwendbarkeit als eines von acht verschiedenen Entwurfsparadigmen diskutiert, die dediziert an einer dienstorientierten Lösung erkennbar sein müssen. Erl bezieht den Aspekt der Wiederverwendbarkeit hauptsächlich auf die Untersuchung der einen Dienst implementierenden Logik. Wiederverwendbarkeit nimmt hier einen fundamentalen Stellenwert ein, da von diesem Aspekt weitere grundlegende Paradigmen abgeleitet werden können. Erl zufolge wird ein höheres Maß an Wiederverwendbarkeit per se erreicht, wenn der Entwurf einer Softwarekomponente nicht nur einem dedizierten Zweck dient, sondern von vornherein im Entwurf auf die Nutzung in unterschiedlichen Kontexten ausgerichtet wird. Das Prinzip der Dienstorientierung mit seinen wesentlichen Charakteristika (Dienstverzeichnis, Dienstnehmer- Dienstgeberbeziehung, klar definierte Schnittstelle und Zugangsprotokoll) begünstigen dabei inhärent das Prinzip der Wiederverwendbarkeit.

Abbildung 12 fasst die unterschiedlichen Aktivitäten beim Entwurf der unterschiedlichen Diensttypen zusammen und stellt diese grafisch dar.

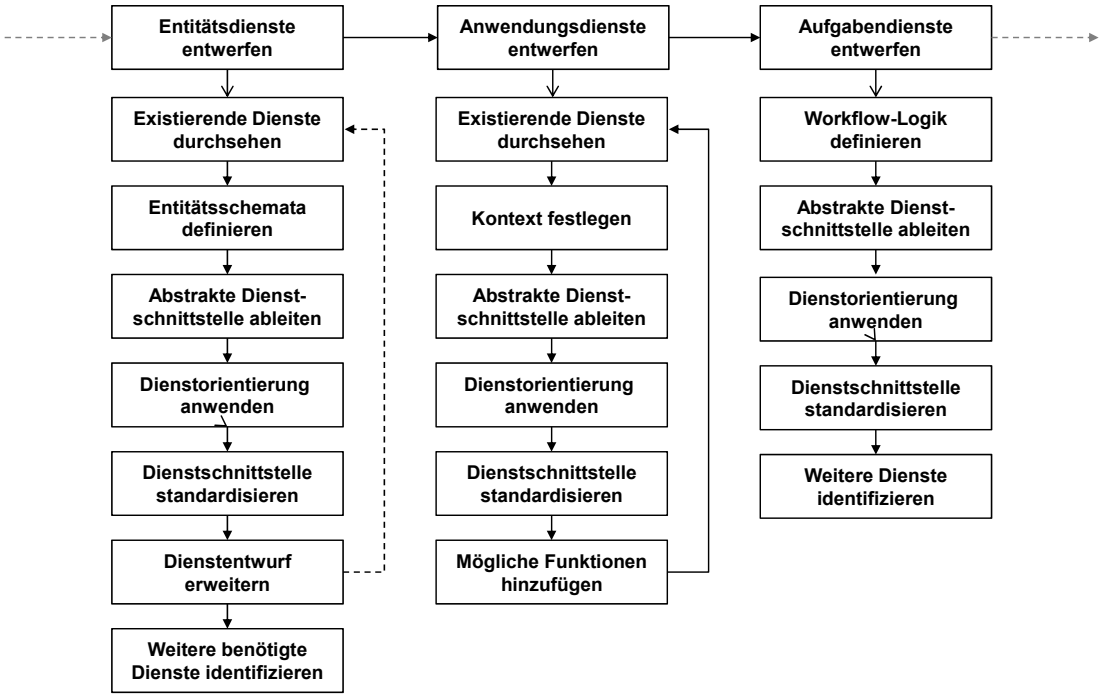

Abbildung 12 Aktivitäten beim Entwurf verschiedener Typen von Diensten [Er06]

Erl formuliert vier wesentliche Entwurfscharakteristika hinsichtlich der Wiederverwendbarkeit. Bei der Forderung nach einem agnostischen Kontext wird angenommen, dass die durch einen Dienst gekapselte Logik hinreichend unabhängig von einem konkreten Kontext ist, damit die Funktionalität des Dienstes in weiteren Szenarien eingesetzt werden kann. Hierbei sind dann jedoch Änderungen an der Zugriffssemantik notwendig. Die Forderung nach einer generischen Dienstlogik ermöglicht ähnlich der Forderung nach einem agnostischen Kontext die Nutzung in unterschiedlichen Szenarien. Der Fokus liegt jedoch auf der Nutzung der durch den Dienst erbrachten Funktionalität. Drittens wird gefordert, dass ein wiederverwendbarer Dienst einen generischen und einfach zu erweiterbaren Dienstvertrag besitzt. Dadurch können Anpassungen einfach durchgeführt werden, wenn der Dienst in weiteren Szenarien genutzt werden soll. Letztlich wird gefordert, dass der Dienst in einem konkurrierenden Zugriffsverfahren genutzt werden kann.

Bewertung der Ansätze

Insgesamt betrachtet ist die von Erl vorgestellte Methode für die Konstruktion dienstorientierter Softwaresysteme umfangreich und vollständig. Neben der eigentlichen Analyse und dem Entwurf von Diensten werden auch die nachfolgenden Phasen des Softwareentwicklungsprozesses betrachtet. Die vorgestellte Methode für die dienstorientierte Analyse und den dienstorientierten Entwurf sind insgesamt nachvollziehbar dargestellt. Wenngleich die von Erl eingesetzte Syntax für die Notation der im Entwicklungsprozess entstehenden Entwurfsartefakte nicht standardisiert ist, ist der Einsatz standardisierter Modellierungssprachen denkbar. Somit kann die Methode für den Referenzentwurf von Managementdiensten eingesetzt werden (Anforderung 1.1). Da die Methode nicht speziell auf die Domäne IT-Management fokussiert ist, wird die Anforderung 1.2 nicht bewertet. Der Entwurf der Dienste führt nach Erl zu einer Menge von Diensten, die inhärent mit fachfunktionalen Anforderungen

der betreffenden Domäne verbunden sind. Der Einsatz oder die Konstruktion eines Metamodells für die Domäne ist jedoch nicht Bestandteil der Methode. Insgesamt wird diese Anforderung daher neutral bewertet (Anforderung 1.3). Als ein wesentliches Entwurfscharakteristikum einer dienstorientierten Softwarelösung identifiziert Erl den Aspekt der Wiederverwendung bestehender Softwareartefakte. Erl zufolge ist ein grundlegendes Ziel beim Entwurf dienstorientiert Lösungen, sowohl bestehende Softwaresysteme wiederzuverwenden als auch die durch einen aktuellen Entwurfsprozess entstehenden Artefakte vor dem Hintergrund einer möglichen späteren Wiederverwendung zu konstruieren. Um eine entstandene Lösung jedoch auf Basis kritisch nachvollziehbarer Ansätze (Modelle, Metriken, Berechnungsvorschriften etc.) zu bewerten, werden keinerlei Hinweise gegeben. Insgesamt wird diese Anforderung daher neutral bewertet.

Engels et al., 2008: Quasar Enterprise

Engels et al. stellen in [EH+08] eine ganzheitliche Methode vor, um den Entwurf von Artefakten einer dienstorientierten Architektur sowohl von der geschäftlichen Ebene als auch von der softwaretechnischen Ebene zu betrachten. Hierzu werden die beiden Begriffe Geschäftsdienst und Anwendungsdienst eingeführt.

Ausgangspunkt der Methode ist zunächst die Definition von Geschäftszielen, die durch den Einsatz von Informationstechnologie realisiert werden sollen. Aus diesen Geschäftszielen kann Engels et al. zufolge eine IT-Strategie sowie eine Geschäftsarchitektur ermittelt werden. Diese Aspekte führen letztlich zu einer IT-Architektur, die in Form einer Struktur der eingesetzten Anwendungen ausgestaltet ist. Bei Engels et al. wird diese Struktur (die eingesetzten Anwendungen sowie deren Beziehungen untereinander) als Anwendungslandschaft bezeichnet. In einem iterativen Entwicklungsvorgehen werden aus diesen genannten Konzepten verschiedene Artefakte erstellt. So steht am Anfang die Definition von Geschäftszielen (Kontext), Geschäftsdiensten und Geschäftsobjekten (Konzept) sowie Geschäftsprozessen (Logik). Geschäftsdienste können Engels et al. aus Teildiensten bestehen und sind demnach nicht atomar. Wichtiger Aspekt für den Entwurf von Diensten ist in der vorgestellten Methode die Definition verschiedener Domänen. Engels et al. zufolge werden Domänen in Subdomänen unterteilt. Somit soll eine schrittweise Eingrenzung und Zusammenfassung verschiedener, zueinander gehörender Aspekte auf Ebene der Datenhaltung möglich werden.

Geschäftsdienste sind Dienste im weiteren Sinne, die einem Dienstnutzer einen geschäftlichen Mehrwert bieten. Durch Verträge ist klar geregelt, welche Leistung durch den Geschäftsdienst erbracht werden soll. Der Geschäftsdienst stellt somit eine Außenansicht auf ein System dar, die nicht zwangsläufig von technischer Natur sein muss. Demgegenüber stehen Anwendungsdienste als Artefakte eines Softwaresystems, das diejenige Funktionalität durch den Einsatz von Softwareartefakten bereitstellt, die erforderlich ist, um Geschäftsdienste zu realisieren. Somit werden komplexe Abläufe in Geschäftsdiensten durch Geschäftsprozesse dargestellt, deren einzelne Aktivitäten durch Funktionalität von Anwendungsdiensten umgesetzt werden. Engels et al. zufolge ist demnach klar von Geschäftsdiensten, die die geschäftliche Architektur betreffen, und Anwendungsdiensten, die den Regeln der Softwarearchitektur genügen, zu unterscheiden.

Bewertung des Ansatzes

Die in [EH+08] vorgestellte Methode "Quasar Enterprise" für den Entwurf einer dienstorientierter Architektur wird am Beispiel eines fiktiven Szenarios beschrieben. Die Methode ist das Ergebnis vielzähliger realer Entwicklungsprojekte und gibt die Erfahrung der Autoren im Bereich der dienstorientierten Softwareentwicklung wieder. Da die Methode auf die Beschreibung der konzeptionellen Elemente im Softwareentwurf fokussiert, ist sie unabhängig von konkreten Modellierungssprachen oder Werkzeugen. Zur Demonstration der einzelnen Methodenschritte wird häufig auf UML-Diagramme zurückgegriffen, was die Unterstützung durch existierende Entwicklungswerkzeuge sicherstellt, jedoch nicht verpflichtend macht. Explizit wird auf die Kenntnis, den Sachverstand oder den Charakter der kreativen Tätigkeit der beteiligten Akteure hingewiesen. Die Methode sieht nicht vor, bereits erstellte Referenzmodelle in den Entwurf zu integrieren oder Referenzmodelle als Zwischenergebnis der Methode zu entwickeln. Insgesamt wird dieser Aspekt daher neutral bewertet (Anforderung 1.1). Die Methode ist nicht dediziert auf Szenarien in der Domäne IT-Management zugeschnitten, aber auch nicht ausschließlich auf das im Buch vorgestellte Szenario beschränkt. Daher wird dieser Aspekt nicht bewertet (Anforderung 1.2). Die Methode sieht explizit vor, dass eine strukturelle Modellierung der jeweils betrachteten Domäne vorgenommen wird. Genauer wird auf Domänenmodelle Bezug genommen wodurch die Modellierung der am Szenario beteiligten Akteure, Geschäftsobjekte etc. bezeichnet wird. Ein abstrahiertes Metamodell wird jedoch nicht vorgestellt, was auch der Tatsache geschuldet ist, dass die vorgestellte Methode nicht dediziert auf die Domäne IT-Management zugeschnitten ist. Daher wird dieser Aspekt neutral bewertet (Anforderung 1.3). Eine fokussierte Betrachtung einzelner Eigenschaften der entworfenen Dienste ist nicht Gegenstand der Arbeit, wenngleich argumentativ versucht wird, gewisse Eigenschaften an die entworfenen Dienste implizit zu betrachten. Insgesamt wird dieser Aspekt daher negativ bewertet (Anforderung 1.4).

Fareghzadeh, 2008: Service Identification Approach to SOA Development

Eine der zentralen Aktivitäten beim Entwurf dienstorientierter Softwarelösungen ist die Identifikation von Diensten auf Basis der zur Verfügung stehenden Analyse- und Entwurfsartefakte. Aufgrund der Komplexität der Zielsetzung sind in jüngster Zeit unterschiedliche Ansätze und Verfahren vorgestellt und diskutiert worden, die sich nicht nur in den zugrunde liegenden Modellierungsansätzen zur Spezifikation von Analyse- und Entwurfsartefakten unterscheiden, sondern auch an den einzelnen Schritten der gesamten Entwurfsmethode. So teilt Fareghzadeh den in [Fa08] vorgestellten Ansatz in insgesamt drei unterschiedliche Phasen auf, die aufeinander aufbauen und gesamtheitlich eine möglichst optimale Lösung für den Entwurf einer dienstorientierten Architektur ergeben sollen. Die drei Phasen werden als *Initial Analysis Phase, In Depth Analysis Phase,* und *Service Taxonomy Phase* bezeichnet.

In der ersten Phase wird eine vollständige Modellierung der Geschäftsdomäne durchgeführt. Neben der Festlegung eines einheitlichen Begriffsverständnisses (z. B. in Form eines Glossars), wird auch bei diesem Ansatz bereits in der Analysephase eine explizite Erfassung der zur Integration möglichen bestehenden Anwendungen vorgenommen. In der zweiten Phase findet eine weitere Modellierung der Geschäftsdomäne statt, wobei hier der Schwerpunkt auf der Definition von Anwendungsfällen und diese Anwendungsfälle realisierenden Geschäftsprozessen gelegt wird. Aus den modellierten

Geschäftsprozessen können durch schrittweise Verfeinerungen die einzelnen Aktivitäten weiter untergliedert werden. In einem zweiten Schritt in dieser Phase wird eine erste Definition des zu konstruierenden Systems erstellt und weiter verfeinert, wobei kritische Abschnitte in der Architektur identifiziert werden können. Die zuvor erstellen Anwendungsfälle bilden die Grundlage, um Integrationspunkte bestehender Systeme in die erstellte architektonische Übersicht zu definieren, wobei im weiteren Verlauf Schnittstellen zu den zu integrierenden Systemen konstruiert werden müssen. Insgesamt findet vergleichbar zu [AG+08] auch hier eine Definition von möglichen Dienstkandidaten statt, ohne diese jedoch vollständig zu spezifizieren. Auf diese Weise kann eine erste Gruppierung logisch zusammengehörender Funktionalität durchgeführt werden.

Nachdem die Dienstkandidaten definiert wurden, findet eine Überprüfung und hiermit einhergehend eventuell eine Umstrukturierung statt. Hierbei werden die in [Er08] beschriebenen Aspekte (Interoperabilität, Modularisierung, Autonomie und Wiederverwendbarkeit) aufgegriffen und vertieft.

Bewertung des Ansatzes

Insgesamt betrachtet ist der von Fareghzadeh vorgestellte Ansatz in [Fa08] eine integrierte Methode, um auf Basis bestehender Softwaresysteme den Entwurf dienstorientierter Architekturen in der Identifikationsphase von Diensten zu unterstützen. Die vorgestellte Methode ist insgesamt nachvollziehbar. Wenngleich in der Arbeit auf den durchgehenden Einsatz standardisierter Modellierungssprachen verzichtet wird, lassen sich unterschiedliche Aspekte der einzelnen Methodenschritte durch den Einsatz unterschiedlicher spezialisierter Modellierungssprachen unterstützen. So wird in dem dargestellten Szenario zur Demonstration der Tragfähigkeit beispielsweise die Modellierung von Geschäftsobjekten auf Basis von UML-Klassendiagrammen durchgeführt. Zusammengefasst kann die Methode als Ansatz für den Referenzentwurf von Diensten eingesetzt werden (Anforderung 1.1). Da die Arbeit originär nicht auf einen Einsatz in der Domäne IT-Management abzielt, wird die grundlegende Anforderung der Unabhängigkeit von konkreten Managementwerkzeugen nicht bewertet. Prinzipiell ist die Methode jedoch unabhängig von konkreten bestehenden Softwaresystemen (Anforderung 1.2). Aufgrund der Ausrichtung der einzelnen Methodenschritte ist der Entwurf von Diensten mit der vorgestellten Methode prinzipiell in Bezug auf eine konkrete Umsetzung von Diensten bezüglich einer klar definierten Domäne anzusehen. Beispielsweise sind dedizierte Methodenschritte vorgesehen, um den Geschäftsbereich festzulegen oder ein Glossar mit den wesentlichen Begrifflichkeiten der zu unterstützenden Domäne zu erarbeiten. Es fehlt jedoch der Einsatz eines integrierten Metamodells der umzusetzenden Domäne, um die einzelnen Aspekte bezüglich eines klaren Domänenbezuges auf Ebene der Modellierungsansätze festzuhalten (Anforderung 1.3). Eine Bewertung der Wiederverwendbarkeit der mit der Methode entworfenen Dienste wird nicht durchgeführt (Anforderung 1.4).

Usländer, 2010: Service-oriented Design of Environmental Information Systems

Usländer untersucht in seiner Dissertation [Us10] den strukturierten Entwurf von verteilten Informationssystemen zur Implementierung fachlicher Anforderungen aus dem Bereich der Umweltbeobachtung. Als Zielplattform zur Gestaltung der verteilten Informationssysteme wird eine Implementierung einer dienstorientierten Architektur gewählt. Die Abbildung von fachlichen

Anforderungen auf die dienstorientierte Architektur wird mittels eines mehrstufigen Abbildungsprozesses durchgeführt. Diese als SERVUS bezeichnete Methode führt über die Analyse des Problembereiches (Domänenanalyse) zu einer Menge spezifizierter Nutzeranforderungen, um diese in Form von Fähigkeiten einer dienstorientierten Architektur in einem iterativen Prozessschritt der Entwurfsmethode abermals zur Verfügung zu stellen. Damit werden in einem mehrstufigen Durchlauf die (abstrakten) Anforderungen seitens der Nutzer über eine abstrahierte Dienstplattform letztlich auf eine (konkrete) Implementierungsplattform abgebildet. Abbildung 13 gibt einen Überblick über die einzelnen Aspekte der Methode.

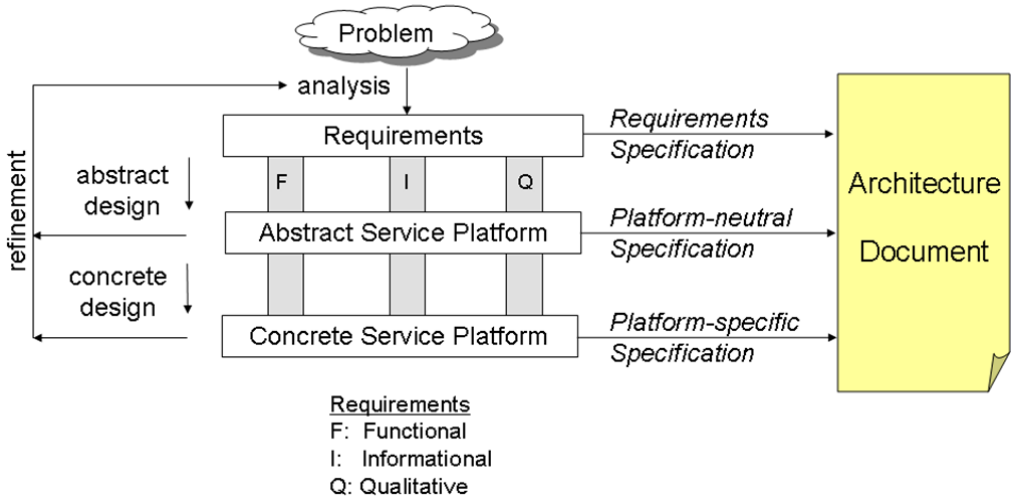

Abbildung 13 Entwurfsmethode nach Usländer [Us10]

Die Spezifikation der einzelnen Aktivitäten für die SERVUS-Methode fokussiert die Anwendung für den Entwurf betrieblicher Informationssysteme in der Domäne der Umweltbeobachtung. Hierbei werden dedizierte Aktivitäten eingeführt, die aufgrund bestehender Standards und Ansätze in diesem Bereich den Entwurfsprozess charakteristisch beeinflussen. Die Integration bestehender Anwendungen wird nicht direkt diskutiert, ist jedoch denkbar durch die mögliche Herangehensweise, die Integrationsaufgabe als umzusetzendes Problem zu motivieren. Prinzipiell jedoch ist die Problemstellung in [Us10] vergleichbar zur der in der vorliegenden Arbeit aufgegriffenen Fragestellung, weshalb einige grundlegende Aspekte in der vorliegenden Arbeit Anwendung finden.

Bewertung des Ansatzes

Der Entwurf von Diensten zur Umweltbeobachtung wird von Usländer in [Us10] nachvollziehbar und von einem abstrahierten Standpunkt aus beschrieben. Die vorgestellte Methode sieht den Einsatz standardisierter Modellierungssprachen, insbesondere UML-basierte Metamodelle, direkt vor und integriert diese in einem ganzheitlichen Entwicklungsvorgehen. Insgesamt hat der durch die SERVUS-Methode durchgeführte Dienstentwurf Referenzcharakter (Anforderung 1.1). Da die Arbeit nicht in der Domäne IT-Management angesiedelt ist, wird keine abstrahierte Betrachtung des Entwicklungsprozesses von konkreten Managementwerkzeugen durchgeführt. Die Anforderung 1.2 wird daher nicht bewertet. Wesentlicher Bestandteil der SERVUS-Methode ist eine formale Erfassung

und Überführung der durch den Nutzer eines Informationssystems angeforderten funktionalen und qualitativen Aspekte, wobei hierzu verschiedene Abstraktionen in Form formal spezifizierter Ontologien behilflich sind. Die hiermit entwickelten Dienste weisen demnach einen hohen Bezug zu der unterstützenden Domäne auf (Anforderung 1.3). Die entstehenden Entwurfsartefakte werden nicht hinsichtlich ihres Grades an Wiederverwendbarkeit untersucht oder bewertet (Anforderung 1.4).

3.3 Abbildung automatisierbarer Managementprozesse

Zur Steigerung der Effizienz und somit auch zur Vermeidung von fehleranfälligen Schritten in Managementprozessen müssen diese durch geeignete Softwaresysteme unterstützt werden. Während der erste Beitrag der vorliegenden Arbeit die Definition von Managementbasisdiensten und Managementprozessdiensten diskutiert, um überhaupt ein geeignetes Managementsystem erschaffen zu können, wird im zweiten Beitrag die Abbildung von Managementprozessen auf diese Softwarelösung betrachtet. Im Folgenden werden bestehende Arbeiten und Ansätze zur Realisierung dieses Beitrages präsentiert und anhand der Kriterienkataloges bezüglich der gestellten Anforderungen bewertet.

Tamm, Zarnekow, 2005: Umsetzung eines ITIL-konformen IT-Service- Support auf der Grundlage von Web-Services

Tamm und Zarnekow beschreiben in [TZ05] den Entwurf von Webservices zur Unterstützung eines typischen *Incident-Management*-Prozesses. Die Motivation des Ansatzes leitet sich aus der Feststellung ab, dass IT-Organisationen einen Wechsel vom reinen Technologie-Lieferanten hin zu einem Dienstleister durchlaufen. Kundenanforderungen und zielgerichtete Lösungen treten in den Vordergrund. Die in letzter Zeit entwickelten Rahmenwerke zur prozessorientierten Gestaltung von Betreiberabläufen helfen zwar bei der Strukturierung von Aktivitäten auf der organisatorischen Ebene, eine darauf aufbauende Abbildung auf Softwarewerkzeuge scheitert jedoch oftmals. Der Einsatz einer dienstorientierten Architektur kann dabei helfen, ausgehend von Prozessdefinitionen notwendige Funktionalität in Form von Softwarediensten abzuleiten. Dies gilt insbesondere auch für Managementprozesse und Managementfunktionen.

Die Autoren beschreiben zunächst die Definition eines typischen *Incident-Management*-Prozesses, wie er gemäß ITIL aufgefasst und in vielen Unternehmen umgesetzt ist. Diese Prozessdefinition wird strukturiert, Teilaktivitäten werden gruppiert und anschließend werden Dienste identifiziert. Hierbei werden neben atomaren Diensten auch komponierte Dienste angegeben. Die Analyse wird durch die Definition von notwendigen Attributen der den Prozess unterstützenden Entität abgeschlossen. Anschließend werden sowohl atomare als auch komponierte Dienste mittels Technologien aus dem Webservice-Umfeld realisiert. Hierbei kommt die *Web Service Description Language* (WSDL) zur Beschreibung der Dienstschnittstellen zum Einsatz, zur Realisierung der komponierten Dienste wird die Webservice-Kompositionssprache BPEL4WS eingesetzt.

Bewertung des Ansatzes

Das von Tamm und Zarnekow vorgeschlagene Vorgehen zur Umsetzung von Managementprozessen auf eine dienstorientierte Architektur ist im Jahre 2005 eine der ersten Arbeiten, die sich mit einer gesamtarchitektonischen Perspektive der Automatisierung von komplexen Betreiberabläufen widmet. Vor diesem Hintergrund ist die Bewertung der praktischen Anwendbarkeit der vorgestellten Methode nur mittelbar möglich, da der Schwerpunkt vor allem auf der Darstellung konzeptioneller Zusammenhänge sowie der praktischen Tragfähigkeit des Einsatzes von Webservice-Technologie beruht (Anforderung 2.1). Für die Modellierung der Prozessaktivitäten und weiter folgend des Entwurfes von Dienstschnittstellen wird kein einheitliches Domänenmodell herangezogen, um eine semantische Übereinstimmung der einzelnen Begrifflichkeiten zu erzielen (Anforderung 2.2). Der beispielhafte Entwurf basiert auf dem Modell eines *Incident-Management*-Prozesses. Dieses Modell ist jedoch rein konzeptionell in Form eines einfachen Ablaufdiagrammes dargestellt. Eine formale Modellierung wird nicht diskutiert, prinzipiell ist aber auch der Einsatz eines formalisierten Prozessmodells möglich (Anforderung 2.3).

Mayerl et al, 2005, 2006: verschiedene Arbeiten

Die Bereitstellung von IT-Diensten mit einer gesicherten Qualität erfordert aufseiten der Betreiber die Definition, Einführung und überwachte Ausführung von fachbezogenen Abläufen. Hinzu kommt die immer stärker zunehmende Abhängigkeit von einer den fachlichen Anforderungen genügende IT-Infrastruktur. So steht bei der Nutzung von IT-Diensten nicht mehr nur der rein technische Aspekt im Vordergrund, sondern vielmehr auch eine fachliche Ausrichtung von Funktionalitäten, die sich aus zu unterstützenden Prozessen ableiten lassen. Mayerl et al. greifen diese Sichtweise in [MV+05, MT+06] auf und schlagen ein aus der Sicht der Managementprozesse getriebenes Entwicklungsvorgehen vor, um Anforderungen an Managementsoftware aus den Managementprozessen abzuleiten. Darüber hinausgehend wird die Integration von bestehenden Managementwerkzeugen in diese neu entwickelte prozessorientierte Managementsoftware mittels der Schaffung von Softwarediensten realisiert. Dadurch können sowohl die Anforderungen der Prozesse als auch bestehende Managementwerkzeuge gleichermaßen betrachtet werden.

Der Beitrag geht von definierten Managementprozessen nach den *Best Practices* der ITIL aus. Ausgehend vom *Service Level Management Process* (SLM) wird zunächst ein Prozessmodell mit der *Business-Process Model and Notation* (BPMN) erstellt. Anhand dieses Prozessmodells können einige grundlegende Aspekte für die Ableitung von Fachfunktionalität betrachtet werden. Beispielsweise werden beteiligte Rollen, Prozessentitäten sowie der Fluss der Prozessentitäten zwischen einzelnen Prozessschritten erkennbar. Eine erste Grobgliederung von zusammenhängenden Teilprozessen innerhalb des SLM-Prozess wird durchgeführt. So kann der Prozess in zwei zusammenhängende Teilprozesse untergliedert werden (SLA Monitoring, SLA Establishment). Diese einzelnen Sub-Prozesse können dann sukzessive weiter verfeinert werden. Die Idee ist, durch die Verfeinerung auf Ebene der Prozessmodellierung möglichst früh weitergehende Automatisierungsmöglichkeiten durch die Verschaltung von Basisdiensten zu erschließen. Hierbei wird vor allem auch der Informationsfluss zwischen den einzelnen Prozessschritten betrachtet. Dies ist notwendig, da durch die Abbildung des SLM-Prozesses auf eine dienstorientierte Architektur vor allem der Nachrichtenaustausch zwischen den einzelnen Diensten die Umsetzung der ablauforganisatorischen Aspekte darstellt. Dies stellt in

gewissem Maße die Grundlage für eine Automatisierung von jeglichen IT-basierten Prozessen dar, da hier vor allem der automatisierte Austausch von Information im Kern der Betrachtungen steht.

Bewertung der Ansätze

Das von Mayerl et al. vorgeschlagene Vorgehen, auf Basis einer formalen Prozessmodellierung den Entwurf von Softwarekomponenten für die Integration von Managementwerkzeugen zu unterstützen, liegt im Wesentlichen ebenfalls dem in der vorliegenden Arbeit vorgestellten Entwicklungsansatz zugrunde. Mayerl et al. modellieren und verfeinern hierzu mit BPMN modellierte Managementprozesse. Dies stellt eine praktische Anwendbarkeit der Ableitungsmethode sicher (Anforderung 2.1), da hier eine standardisierte Modellierungssprache zum Einsatz kommt. Hinsichtlich einer möglichen (teil-)automatisierten Transformation von Prozessmodellen auf Schnittstellenartefakte einer dienstorientierten Architektur kann heute die Version 2 der Modellierungssprache herangezogen werden, die im Gegensatz zur in der Publikation betrachteten Version 1.1 ein formal definiertes Metamodell beinhaltet. Die Modellierung von Prozessabläufen wird nicht auf Basis eines einheitlichen Domänenmodells durchgeführt (Anforderung 2.2), wenngleich Geschäftsobjekte und Entitäten in einem separaten Entwicklungsschritt identifiziert und modelliert werden. Dies erschwert die semantische Vergleichbarkeit unterschiedlicher entwickelter Schnittstellenbeschreibungen untereinander, was bezüglich Anforderung 2.1 die praktische Anwendbarkeit nur auf den eigentlichen Transformationsschritt von Prozessmodell auf Dienstmodell unterstützt. Weiterhin wird eine Integration bestehender Managementwerkzeuge dahingehend erschwert, dass die Abbildung von entworfenen Schnittstellenbeschreibungen auf tatsächlich vorhandene Schnittstellen zu Managementwerkzeugen in einem zusätzlichen manuellen Entwicklungsschritt durchgeführt werden muss. Die in dem vorgestellten Ansatz durchgeführte Modellierung von Managementprozessen wird auf Basis eines (nun mittlerweile verfügbaren) formal definierten Metamodells durchgeführt (Anforderung 2.3).

Brown, Keller, 2006: A Best Practice Approach for Automating IT Management Processes

Die Motivation für die Automatisierung von betrieblichen Abläufen speist sich aus der Feststellung, dass der wirtschaftliche Druck auf IT-Dienstleister wächst. In [BK06] betrachten Brown und Keller daher eine Automatisierung von Managementprozessen hinsichtlich Arbeitskosten von teilnehmenden menschlichen Akteuren innerhalb dieser Prozesse. Beispielhaft wird der *Change-Management*-Prozess untersucht, da dieser Prozess aufgrund seiner organisatorischen Struktur und seines technischen Bezuges vielfältige Herausforderungen an die Automatisierung stellt. In Anlehnung an die zur Umsetzung von Managementprozessen herangezogenen *Best Practices* der *Information Technology Infrastructure Library* (ITIL) wird die in der Arbeit vorgestellte Methode zur Automatisierung von Managementprozessen ebenfalls als *Best Practice* präsentiert. Die Methode umfasst insgesamt sechs aufeinander aufbauende Schritte:

(1) Identifizierung der verfügbaren *Best Practices* für einen zu automatisierenden Prozess.

(2) Einschränkung der zu automatisierenden Teile des Prozesses auf einige wenige, einfach umzusetzende Aspekte.

(3) Identifizierung von Workflows und zusammenhängenden Aktivitäten in den eingeschränkten Prozessteilen bezüglich Delegationsmöglichkeiten für automatisierte Aktivitäten.

(4) Identifizierung von Kontroll- und Datenschnittstellen zur Umsetzung von automatisierten Aktivitäten.

(5) Identifizierung und Entwurf von zusätzlichen Prozessen, um die durch die Automatisierung von Managementprozessen nötige Infrastruktur zu betreiben und weiterzuentwickeln.

(6) Implementierung des Prozessflusses und der delegierten Aktivitäten.

Von besonderem Interesse ist die Aufteilung der Analyse von Automatisierungsmöglichkeiten in mehrere verschiedene Teilschritte. Die Identifizierung von delegierten Aktivitäten bezieht sich auf eine vertiefte Abwägung von Vorteilen, die durch die Automatisierung dedizierter Schritte eines Managementprozesses zu erzielen sind. Die Umsetzung im Rahmen der Implementierung des Prozessflusses und der delegierten Aktivitäten erfolgt am Beispiel des *Change-Management*-Prozesses auf Basis einer Webservice-basierten Architektur. Insgesamt stellt die vorgestellte Methode ein pragmatisches Vorgehen dar, das hauptsächlich auf einer argumentativen Analyse der zu unterstützenden Managementprozesse zielt.

Bewertung des Ansatzes

Die von Brown und Keller vorgestellte Methode zielt auf eine detaillierte Analyse eines zu automatisierenden Managementprozesses, wobei eine genaue Abwägung von Kosten/Nutzen einzelner zu automatisierender Teilschritte erfolgt. Insgesamt ist die vorgestellte Methode pragmatisch und zielorientiert. Wenngleich in der Veröffentlichung auf den Einsatz standardisierter Modellierungssprachen verzichtet wird, ist eine Modellierung und Analyse auf Basis von z. B. mit BPMN modellierten Prozessen denkbar. Die praktische Anwendung der Methode scheint daher prinzipiell möglich (Anforderung 2.1). Der Ansatz sieht keine Modellierung von Prozessen, Prozessaktivitäten, Eintitäten, Teilnehmern oder Rollen auf Basis eines einheitlichen Domänenmodells vor. Diese Beobachtung ist der Tatsache geschuldet, dass dem Entwurf die Analyse von ITIL-basierten *Best Practices* zu Grunde liegt, eine einheitliche Begrifflichkeit nicht genau festgelegt wird (Anforderung 2.2). Eine Formale Modellierung von Managementprozessen wird nicht vorgenommen, wäre jedoch prinzipiell denkbar (Anforderung 2.3).

Ayachitula et al., 2007: IT service management automation – A hybrid methodology to integrate and orchestrate collaborative human centric and automation centric workflows

Im Kern der Betrachtung der Veröffentlichung von Ayachitula et al. aus dem Jahre 2007 [AB+07] steht die Analyse und Bewertung der Komplexität von Managementprozessen mit dem Ziel, automatisierbare Teile zu identifizieren. Grundlegend können vollautomatisierbare Teile von Prozessen von Teilen mit menschlicher Interaktion unterschieden werden. Die Autoren schlagen

hierzu eine einfache Methode vor, die im Wesentlichen ein konzeptionelles Rahmenwerk zur Bestimmung der Komplexität der Domäne IT-Management einbezieht [DK06, DK+07]. Im Kern des Ansatzes steht eine Einteilung in die drei Kategorien (1) Ausführungskomplexität, (2) Koordinationskomplexität und (3) Geschäftsobjektkomplexität. Die Kategorie Ausführungskomplexität kann weiter in die Unterkategorie (1.1) Entscheidungskomplexität unterteilt werden. Der Ansatz zielt auf eine durchgehende Bewertung der Komplexität von Managementprozessen, indem anhand der drei verschiedenen Kategorien zunächst einzelne Managementaktivitäten eines Prozesses bewertet werden, darauf aufbauend dann eine integrierte Bewertung des entsprechenden Prozesses durchgeführt werden kann.

Die erste Kategorie bezieht sich auf Metriken, die die Zusammenhänge aller Aktivitäten eines Managementprozesses betrachtet, bemisst. Hierzu zählen beispielsweise die Anzahl an Kontextwechsel zwischen verschiedenen Aufgaben, die Anzahl unterschiedlicher beteiligter Rollen an einer Aufgabe oder der Grad an Automatisierung einer Aufgabe. Zur Unterkategorie der Entscheidungskomplexität werden alle Metriken gezählt, die den direkten Kontrollfluss betreffen, beispielsweise die Anzahl an Verzweigungen an einer Entscheidungsstelle. Die zweite Kategorie bezieht sich auf Metriken, die den Grad der Komplexität bei der Ablaufsteuerung über verschiedene organisatorische Grenzen hinweg bemessen. Die dritte Kategorie bezieht sich direkt auf diejenigen Prozessartefakte, die in Funktion eines Geschäftsobjektes die zur Ausführung eines Prozesses nötige Information in Form von Daten beinhalten.

Durch die Einteilung in die drei genannten Kategorien wird eine integrierte Betrachtung der Komplexität von Aspekten innerhalb eines Prozesses, prozessübergreifender Austausch von Information sowie der die Prozesse unterstützenden Informationssysteme erzielt. Am Beispiel eines einfachen *Change-Management*-Prozesses wird die Methode demonstriert und kritische Teile innerhalb des Prozesses werden identifiziert. Hierbei werden diejenigen Teile identifiziert, die durchgehend automatisierbar sind, sowie diejenigen Teile, die eine Interaktion mit menschlichen Teilnehmern in einem Managementprozess erfordern.

Bewertung des Ansatzes

Die Ableitung von Entwurfsartefakten zur Spezifikation einer dienstorientierten Architektur ist nicht Gegenstand der Arbeit, weshalb die Anforderung 2.1 nicht bewertet werden kann. Die für die Abschätzung der Komplexität notwendigen Abstraktionen von Managementprozessen in Form definierter Prozessmodelle erfolgt nicht auf Basis eines einheitlichen Domänenmodells. Gerade aber eine einheitliche und durchgehende Klarstellung der Bedeutung wesentlicher Begriffe der Domäne ist wichtig, um eine gesicherte Abschätzung bestimmter Eigenschaften der zu untersuchenden Prozesse zu erreichen (Anforderung 2.2). Beispielweise kann eine Komplexitätsbewertung von organisationsübergreifenden Prozessen auf Basis unterschiedlicher Prozessmodelle nur gesichert erfolgen, wenn semantische Bezüge kooperativer Aktivitäten klar und eindeutig definiert sind. Eine formale Modellierung von Managementprozessen wird in der Arbeit nicht diskutiert. Sie ist prinzipiell jedoch möglich und wird nicht ausgeschlossen (Anforderung 2.3).

3.4 Zusammenfassung und Handlungsbedarf

Zusammenfassend können nun die Ergebnisse der Bewertung der unterschiedlichen Arbeiten hinsichtlich der gestellten Anforderungen dargestellt werden. Aus den beiden folgenden Tabellen ist jeweils ersichtlich, ob eine untersuchte Arbeit die gestellte Anforderung erfüllt (dargestellt mit einem "+"), teilweise erfüllt (dargestellt mit einem "o") oder aber nicht erfüllt (dargestellt mit einem "-"). Da bei einigen der betrachteten Arbeiten eine Bewertung mangels fundierter Bewertungsgrundlage nicht möglich ist, wird in diesem Fall das entsprechende Feld ausgegraut.

Die Ergebnisse der kritischen Betrachtung von bestehenden Arbeiten bezüglich des Entwurfes wiederverwendbarer Managementdienste ist in Tabelle 3 dargestellt. Die getrennte Betrachtung der beiden Teilaspekte „IT-Management“ und „dienstorientierte Softwareentwicklung“ zeigt auf, dass die Arbeiten der ersten Kategorie jeweils keine kritische Bewertung der eigenen Lösungen bezüglich des Aspektes der Wiederverwendbarkeit auf Basis einer nachvollziehbaren Metrik durchführen, während bei den untersuchten Arbeiten bezüglich des dienstorientierten Softwareentwurfes aufgrund des fehlenden Domänenbezugs die Anforderung 1.3 nicht bewertet werden kann.

		IT-Management					Dienstorientierte Softwareentwicklung				
		Anerousis, 1999	Aschemann, Hasselmeyer, 2001	Xiao et al., 2008	Lu et al. 2008, 2009	Schaaf, Brenner, 2008	Arsanjani, 2008	Erl, 2008	Engels et al. 2008	Fareghzadeh, 2008	Usländer, 2010
A1	1.1 Referenzcharakter des Dienstentwurfes	o	o	-	-	o	+	+	o	+	+
	1.2 Unabhängigkeit von konkreten Managementwerkzeugen	+	+	o	o	+					
	1.3 Integriertes Metamodell der Domäne	-	o	-	-	-	-	o	o	-	+
	1.4 Nachweis der Wiederverwendbarkeitder entworfenen Dienste	-	-	-	-	-	o	o	-	-	-

Tabelle 3 Ansätze für den Entwurf von Managementdiensten

Die Arbeiten von Anerousis [An99] bzw. Aschemann und Hasselmeyer [AH01] betrachten zwar den Entwurf von Managementdiensten auf Basis standardisierter Web-Technologien, jedoch sind grundlegende Anforderungen nicht erfüllt. So ist die praktische Anwendbarkeit der vorgestellten Ansätze aufgrund fehlender Modelle oder der nicht vorhandene Einsatz standardisierter Modellierungssprachen nur bedingt gegeben. Der Entwurf der Dienste wird nicht auf Basis eines allgemeinen Verständnisses des zu unterstützenden Problembereiches vorangetrieben. Die Wiederverwendbarkeit der entworfenen Dienste wird nicht bewertet.

Xiao et al. [XL+09] präsentieren ein architektonisches Muster zur Gestaltung eines hierarchisch-strukturierten Aufbaus zur Aggregation von vielen verteilten, auf *WS-Management*-basierten Managementagenten. Wenngleich viele der gestellten Anforderungen durch diese Arbeit nicht erfüllt werden, ist der vorgestellte Ansatz zur Gestaltung einer verteilten Architektur hinsichtlich der technischen Aspekte (Überwachung und Steuerung von *Managed Resources*) von Interesse.

In den Arbeiten von Lu et al. [LF+08, LW+08, LL+08, LW+09, LW+09b] werden architektonische Muster untersucht und diskutiert, die einen dienstorientierten Zugriff auf technische Managementinformation ermöglichen. Somit können abstrahierte und generische Managementschnittstellen zu unterschiedlichen, teilweise proprietären Schnittstellen von Infrastrukturressourcen geschaffen werden. Insgesamt sind die von Lu et al. durchgeführten Arbeiten jedoch sehr stark auf den Einsatz von *WS-Management* bzw. WSDM fokussiert, ein domänenspezifischer und nachvollziehbarer Entwurf von Managementdiensten zur Integration von Werkzeugen für die ganzheitliche Automatisierung von Betreiberprozessen ist nicht ersichtlich. Der Nachweis typischer Diensteigenschaften wie beispielsweise der Wiederverwendbarkeit fehlt in allen Arbeiten.

Schaaf und Brenner zeigen in [SB08] eine durchgehende Methode auf, um angelehnt an das Vorgehen der modellgetriebenen Softwareentwicklung über die analytische Betrachtung von systemunabhängigen, plattformunabhängigen und plattformspezifischen Modellen hin zu einer Softwarelösung zu gelangen, die genau die funktionalen Anforderungen eines betrachteten Betreiberprozesses umsetzt. Insgesamt stellt die von den Autoren vorgestellte Arbeit einen wesentlichen Baustein dar, um die Anforderungen an ein integriertes Softwaresystem umzusetzen. Grundlegende Fragen werden durch diese Arbeit jedoch nicht beantwortet, so ist z. B. nicht ersichtlich, ob dem gesamten Ansatz ein abstrahiertes Modell der Domäne zugrunde liegt. Diese Anforderung stellt einen kritischen Punkt dar, um die Integration bestehender Werkzeuge in automatisierbare Betreiberprozesse auf Basis einer dienstorientierten Architektur bewerkstelligen zu können.

Insgesamt kann festgestellt werden, dass die bestehenden Arbeiten im Bereich des Entwurfes von Managementdiensten hauptsächlich die Unterstützung der technischen Aspekte beim Betrieb von IT-Systemen betrachten. Hierzu gehört vor allem die Überwachung und Steuerung von Ressourcen. Der Entwurf von Managementdiensten zur Unterstützung der organisatorischen Aspekte, also gerade jener Managementfunktionalitäten die zur qualitätsgesicherten Ausführung von Betreiberprozessen wie beispielsweise dem *Incident Management* notwendig sind, werden durch diese Arbeiten bislang nicht betrachtet. Hieraus ergibt sich die Feststellung, dass eine integrierte Gesamtbetrachtung beider Aspekte bislang nicht verfolgt wurde.

Die untersuchten Arbeiten im Bereich der dienstorientierten Softwareentwicklung zeigen ein sehr viel differenzierteres Bild bezüglich des Referenzcharakters des entstehenden Dienstentwurfes. Die Methoden von Erl [Er06, Er08], Arsanjani et al. [AG+08] oder Engels et al [EH+08] stellen umfangreiche Rahmenwerke zur Verfügung, um fachliche Anforderungen in einem integrierten Entwicklungsprozess auf eine dienstorientierte Softwarelösung umzusetzen. Hierbei werden jeweils unterschiedliche Schwerpunkte gesetzt. Die verfeinerte Betrachtung von unterschiedlichen Typen von

zu entwerfenden Diensten bei Erl oder Engels et al. geht auf spezielle Eigenschaften unterschiedlicher Ausprägungen verschiedener möglicher Dienste ein und geht damit über den Ansatz von Arsanjani et al. hinaus. Beide Ansätze betrachten jedoch nicht eine Modellierung auf Basis eines einheitlichen Domänenmodells. Die SERVUS-Methode von Usländer [Us10] fokussiert die Überführung formal spezifizierter Analyse- und Entwurfsartefakte auf eine konkrete Implementierung einer dienstorientierten Architektur. Grundlegender Baustein ist die formale Spezifikation der wesentlichen Domänenelemente. Alle Arbeiten weisen jedoch mindestens eine, zur Begegnung der identifizierten Problemstellung für den Kontext der vorliegenden Arbeit kritische, nicht erfüllte Anforderung auf.

Die Ergebnisse bezüglich der Bewertung von bestehenden Arbeiten mit der Zielsetzung, automatisierbare Managementprozesse auf einen dienstorientierten Lösungsansatz abzubilden, sind in Tabelle 4 dargestellt. Es ist ersichtlich, dass die untersuchten Arbeiten allesamt keine einheitliche Begrifflichkeit auf Basis eines gemeinsamen Domänenmodells verfolgen.

		Tamm, Zarnekow, 2005	Mayerl et al., 2005, 2006	Brown, Keller, 2006	Ayachitula et al., 2007
A2	2.1 Flexible Anwendbarkeit der Methode	-	+	+	
	2.2 Einheitliche Begriffe auf Basis eines Domänenmodells	-	o	-	-
	2.3 Modellierung von Managementprozessen	o	o	-	o

Tabelle 4 Ansätze für die Abbildung automatisierbarer Managementprozesse

Eine der ersten Arbeiten mit der Motivation, Managementprozesse durch die Abbildung auf eine dienstorientierte Architektur zu automatisieren, wird von Tamm und Zarnekow 2005 in [TZ05] vorgestellt. In dieser Arbeit werden hauptsächlich grundlegende Aspekte einer dienstorientierten Softwarelösung diskutiert, eine klar herausgearbeitete Methode ist nicht erkennbar. Die prinzipielle Tragfähigkeit einer dienstorientierten Architektur wird anhand eines einfachen Beispieles zur Automatisierung eines *Incident-Management*-Prozesses demonstriert. Die Arbeiten von Mayerl et al. zeigen grundlegende Aspekte einer durchgehenden Methode auf, um aus formal modellierten Managementprozessen verschiedene Entwurfsartefakte abzuleiten, die den weiteren Verlauf der Softwareentwicklung unterstützen [MV+05, MT+06]. Brown und Keller beziehen in der in [BK06] vorgestellten Methode verschiedene Schritte ein, um eine Automatisierung zielorientiert anhand einfach zu automatisierender Managementaktivitäten zu erreichen. Ayachitula et al. betrachten lediglich die Analyse automatisierbarer Betreiberprozesse, ohne die Modellierung der zugrunde liegenden Domäne zu fokussieren.

Zusammengefasst kann also festgestellt werden, dass einzelne Arbeiten jeweils unterschiedliche Aspekte betrachten. Eine durchgängige Methode für den Entwurf wiederverwendbarer Managementdienste auf Basis eines einheitlichen Metamodells der Domäne IT-Management sowie

darauf aufbauend der Abbildung von Betreiberprozessen auf die entworfene Architektur steht bislang nicht zur Verfügung.

Zielsetzung der vorliegenden Arbeit ist daher, den Entwurf wiederverwendbarer Managementdienste zur Integration von Managementwerkzeugen systematisch und nachvollziehbar zu beschreiben sowie eine konstruktive Methode zur Abbildung von Betreiberprozessen auf diese entworfenen Managementdienste zu formulieren. Im Kern steht ein Metamodell der Domäne IT-Management, das diese unterschiedlichen Aspekte aufgreift und die Möglichkeit eröffnet, in einem integrierten Entwicklungsvorgehen nachvollziehbar eingesetzt zu werden. Das diskutierte Metamodell wird zunächst konzeptionell entwickelt, um anschließend daraus verschiedene Teile der abstrakten Syntax auf verschiedene, teilweise durch standardisierte Modellierungssprachen vorgegebene konkrete Syntaxen umzusetzen. Die Präsentation dieser einzelnen Beiträge erfolgt in den beiden nachfolgenden Kapiteln 4 (**Metamodell zur Spezifikation von Managementdiensten**), und Kapitel 5 (**Domänengetriebener Entwurf wiederverwendbarer Managementdienste**). In Kapitel 6 wird darauf aufbauend die praktische Anwendung in einem durchgehenden Szenario vorgestellt.

4 Metamodell für den Entwurf wiederverwendbarer Managementdienste

Um Betreiberprozesse zu automatisieren, ist die Unterstützung durch eine entsprechende Ausführungsumgebung erforderlich. Die Strukturierung dieser Ausführungsumgebung nach den Prinzipien der dienstorientierten Architektur ermöglicht die Nutzung der von den Managementwerkzeugen angebotenen Managementfunktionen, um eine automatisierte Ausführung von Betreiberprozessen auf Basis von bestehenden Managementwerkzeugen zu realisieren. Ein besonderer Schwerpunkt der Elemente der dienstorientierten Architektur stellt die Orientierung der Dienste an den zu unterstützenden Prozessen dar sowie die Eigenschaft von Diensten, in weiteren Prozessen wiederverwendet werden zu können. Um diese Eigenschaften der entworfenen Managementdienste zu erreichen, erfordert der Entwurfsprozess einer dienstorientierten Architektur eine strukturierte, nachvollziehbare und systematische Entwicklungsmethode, die an wesentlichen Eigenschaften der zu entwickelnden Lösung ausgerichtet ist. Um frühzeitig Aussagen bezüglich Betreiberprozessorientierung und Wiederverwendbarkeit von entstehenden Managementdiensten treffen zu können, ist die Unterstützung des Entwurfsprozesses durch geeignete Modelle und Modellüberführungen erforderlich. Hierzu ist ein gemeinsames Verständnis der betrachteten Domäne notwendig. Damit ein gemeinsames Verständnis über die Konzepte der betrachteten Domäne etabliert werden kann, ist die Definition eines geeigneten Metamodells erforderlich. Managementdienste, die auf der Grundlage dieses Metamodells entworfen werden, sind an einer einheitlichen Darstellung der verschiedenen relevanten Konzepte ausgerichtet, wodurch die Wiederverwendbarkeit der einzelnen Managementdienste über die Grenzen eines einzelnen abgeschlossenen Szenarios hinaus ermöglicht wird. Das Metamodell einer Domäne beschreibt die verschiedenen relevanten Konzepte dieser Domäne, damit auf Basis des Metamodells konforme Instanzen zu diesem Metamodell gebildet werden können. Dies wird von bestehenden Arbeiten bislang nicht betrachtet.

Die Definition des Metamodells ist Gegenstand dieses Abschnittes. Ergänzend wird aufgezeigt, inwiefern dieses Metamodell die Konstruktion von Managementdiensten bezüglich einer wesentlichen Eigenschaft – der Eigenschaft der Wiederverwendbarkeit – zielführend unterstützen kann.

4.1 Einordnung in den Softwareentwicklungsprozess

Das vorgestellte Vorgehen hat zum Ziel, in der Praxis Anwendung zu finden. Um die beschriebenen Beiträge daher klar einordnen zu können, wird aufbauend auf den im ersten Kapitel identifizierten Problemstellungen zunächst der Bezug zu einem durchgehenden und strukturierten Entwicklungsprozess skizziert.

4.1.1 Betrachtetes Szenario

Die Grundlage hierzu liefert das in Abbildung 14 dargestellte abstrahiert zu betrachtende Szenario. Der zu unterstützende Prozess wird durch einen Analysten (Geschäftsprozessanalysten) modelliert und als Grundlage für den weiteren Entwurf von Diensten als Artefakt dem Architekten bereitgestellt [EH+08]. Der Fokus eines Architekten bei der Konstruktion dienstorientierter Schnittstellen zu bestehenden Managementwerkzeugen liegt bei der Orientierung am zu unterstützenden

Betreiberprozess. Dies stellt sicher, dass die betrachteten Betreiberprozesse durch Managementdienste zielgerichtet unterstützt werden können. Das bedeutet, dass eine einfache und direkte Abbildung derjenigen Aktivitäten im Prozess vorgenommen werden kann, die durch den Einsatz entsprechender Managementdienste realisiert werden. Dem gegenüber stehen die technisch-orientierten Schnittstellen, die bestehende Managementwerkzeuge in der Regel aufweisen. Die technischen Schnittstellen werden von den jeweiligen (Fach-)Entwicklern verstanden, weisen jedoch ein geringes Maß an direktem Bezug zu den zu unterstützenden Betreiberprozessen auf. Damit entsteht eine semantische Lücke zwischen den von den Analysten identifizierten erforderlichen und den von den Entwicklern identifizierten bereitgestellten Funktionen. Dies wirkt sich negativ auf die Wiederverwendbarkeit der Werkzeuge aus, da die Konstruktion passender Schnittstellen jeweils Szenario-abhängig durchzuführen ist. Die dienstorientierte Integration bestehender Managementwerkzeuge erfordert daher, dass ein gemeinsames Verständnis über die zu unterstützenden Betreiberprozesse bei der Konstruktion von Managementdiensten vorherrscht und allen beteiligten Akteuren zur Verfügung gestellt wird. Dieses gemeinsame Wissen bildet die Grundlage, einen einheitlichen und durchgehenden Entwurf voranzutreiben. Neben der Orientierung am zu unterstützenden Betreiberprozess wird der Entwurf vor allem hinsichtlich einer möglichen Wiederverwendbarkeit der entstehenden Managementdienste hin ausgelegt. Wiederverwendbare Managementdienste bieten den Vorteil, dass sie sowohl einfacher in weiteren Betreiberprozessen zum Einsatz kommen können, aber auch eine flexible Umsetzung sich ändernder Anforderungen auf Basis der bestehenden Dienste ermöglichen.

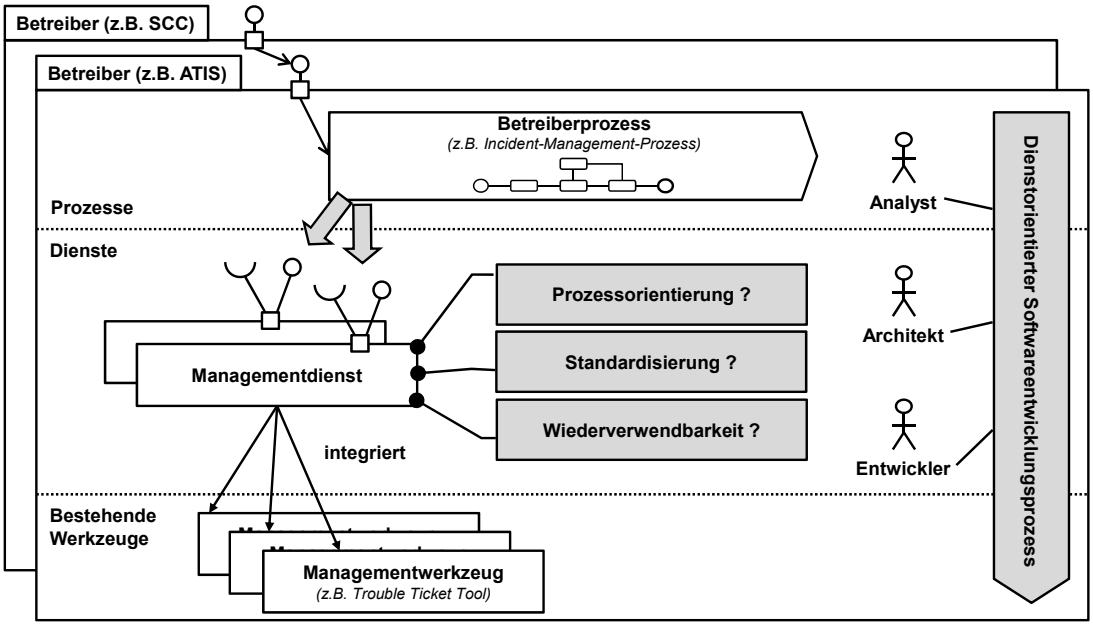

Abbildung 14 Szenario: Dienstorientierte Integration von Managementwerkzeugen

Fokussiert man den Entwurf wiederverwendbarer Managementdienste, ist der Einsatz von Verfahren erforderlich, die die Konzepte der zugrunde liegenden Domäne auf eine einheitliche und formale Art und Weise darstellen [FG+02]. Um ausgehend von den bestehenden Managementwerkzeugen

dienstorientierte Schnittstellen konstruieren zu können, die sowohl ausgerichtet an den zu unterstützenden Betreiberprozessen sind als auch ein definierbares Maß an Wiederverwendbarkeit aufweisen, wird ein strukturiertes und auf bestehenden Ansätzen im Bereich des dienstorientierten Entwurfes von Softwaresystemen aufbauendes Vorgehens verfolgt.

4.1.2 Dienstorientierter Entwicklungsprozess

Das betrachtete Entwicklungsvorgehen orientiert sich zunächst an den anerkannten und in der Praxis vielfach bewährten Vorgehensmodellen für den objektorientierten Entwurf verteilter Softwaresysteme [Ba00, BD09] und wird um die beim Entwurf dienstorientierter Softwaresysteme zusätzlichen Aspekte durch die in [Er06, EH+08, Fa08, AG+08] vorgestellten Verfeinerungen ergänzt. Dies betrifft insbesondere die konkrete Ausgestaltung der Analyse- und Entwurfsphasen, wobei explizit auf die in allen Ansätzen beschriebene Grundlage der Domänenmodellierung und Definition von Dienstkandidaten als Ergebnis der dienstorientierten Analyse eingegangen wird. Insgesamt kann das zugrunde gelegte Entwicklungsvorgehen daher als Abstraktion von aus der wissenschaftlichen Literatur anerkannten Methoden und in der Praxis erprobten Entwicklungsansätzen angesehen werden. Dadurch können die in der vorliegenden Arbeit gelieferten Beiträge in jeweils konkreten Ausgestaltungen eines speziellen Entwicklungsvorgehens zum Einsatz kommen.

Der dieser Arbeit zugrunde liegende Entwicklungsprozess gliedert sich wie in Abbildung 15 ersichtlich in vier verschiedene Phasen (**Dienstorientierten Analysephase**, **Dienstorientierte Entwurfesphase**, **Implementierungsphase** sowie **Verteilung, Test und Inbetriebnahme**). Die letztgenannte Phase (Verteilung, Abnahmetest und Inbetriebnahme) sind nicht Gegenstand der Betrachtungen und werden daher im weiteren Verlauf auch nicht weiter untersucht.

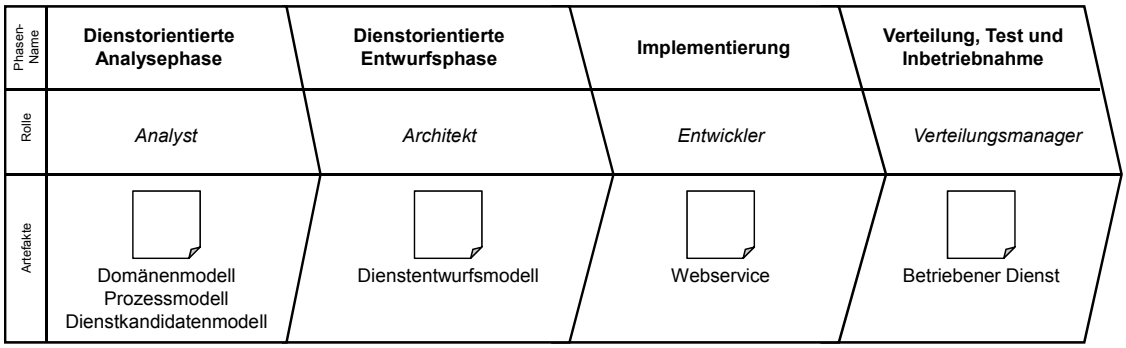

Abbildung 15 Zugrunde gelegter Softwareentwicklungsprozess

Ziel der **dienstorientierten Analyse** ist die Überführung der vorgegebenen fachlichen Anforderungen in ein einheitliches und klar definiertes Format, das die Grundlage für die weitere Spezifikation zu realisierender Dienste bildet. Diese Phase entspricht der Definitionsphase in [Ba00] oder der Analysephase in [BD09]. Im zentralen Ergebnisartefakt der dienstorientierten Analyse werden Dienstkandidaten identifiziert [Er06], die mögliche Dienste darstellen, um fachlich benötigte Funktionalitäten auf Basis der Prinzipien einer dienstorientierten Architektur abzubilden. Bei den Dienstkandidaten handelt es sich jedoch noch nicht um konkrete Vorgaben, welche Dienste genau im weiteren Verlauf zu spezifizieren sind, sondern vielmehr zunächst um Vorgaben, welche

Funktionalität grundlegend gebündelt von welchen Diensten benötigt wird [Er06, Fa08]. Auf Basis der identifizierten Dienstkandidaten können spätere Entwurfsentscheidungen getroffen werden, benötige Dienste beispielsweise von externen Anbietern einzukaufen oder durch die Integration bestehender Softwaresysteme zu implementieren. Die Definition von Dienstkandidaten liefert einen Gesamtblick auf die zu entwerfende Architektur [Er06] und entspricht daher im Wesentlichen der bei objektorientierten Entwicklungsansätzen durchgeführten Systemspezifikation [BD09]. Die Definition von Dienstkandidaten umfasst daher vor allem auch die Darstellung von verschiedenen Abhängigkeiten, die die identifizierten Dienstkandidaten untereinander besitzen. Somit wird neben der Darstellung der einzelnen Elemente der dienstorientierten Architektur auch eine erste Sicht auf die weiter zu spezifizierende Gesamtarchitektur erstellt. Dies entspricht der Phase der Systemspezifikation in [BD09]. Die Grundlage für den Entwurf von Diensten bildet ein gemeinsames Verständnis der betrachteten Domäne [AG+08, Fa08, Er06], weshalb die Modellierung der Domäne einen zentralen Stellenwert einnimmt.

Im **dienstorientierten Entwurf** wird auf Basis der identifizierten Dienstkandidaten eine Spezifikation von konkreten Dienstschnittstellen erarbeitet, die im weiteren Verlauf implementiert werden können. Durch die Trennung von Schnittstellenspezifikation und Implementierungslogik und die klare Ausrichtung der dienstorientierten Entwurfsphase auf die Konstruktion der Schnittstellenspezifikationen wird eine Abstraktion der einem Dienst zugrunde liegenden Implementierung erreicht. Diese abstrakte Dienstschnittstellenspezifikation bezieht wesentliche Konzepte einer dienstorientierten Umsetzung ein (nachrichtenorientierte Kommunikation, formal spezifizierte Dienstoperationen), ist jedoch unabhängig von konkreten Schnittstellenbeschreibungssprachen.

Die **Implementierungsphase** des dienstorientierten Entwicklungsprozesses zielt auf eine Realisierung der erstellten abstrakten Dienstschnittstellen und Dienstkomponentenmodelle. Die Spezifikation von konkreten Dienstschnittstellen konkretisiert die abstrakten Dienstschnittstellen, indem Ansätze zur genauen Beschreibung der weiter zu implementierenden Dienstentwürfe wie beispielsweise die *Web Service Description Language* (WSDL) [W3C-WSDL] oder *die Business Process Execution Language* (BPEL) [OASIS-WSBPEL] zum Tragen kommen. In dieser Phase wird die technische Integration eines bestehenden Werkzeuges bewältigt, oder es wird eine vollständige Neuimplementierung durchgeführt.

Die Phase **Verteilung**, **Test** und **Inbetriebnahme** liefert das fertige und erstellte Produkt aus, wobei Abnahmetests durchgeführt und spezielle Inbetriebnahme-Prozesse durchlaufen werden. Falls nachträgliche Änderungen auftreten, werden diese dokumentiert und entweder einer neuen Instanz eines Entwicklungsvorgehens zugeführt oder im Rahmen vorher vereinbarte Änderungsprozesse umgesetzt.

4.1.3 Modelle im Entwicklungsprozess

Die beteiligten Akteure in der dienstorientierten Analyse (Analyst) und im dienstorientierten Entwurf (Architekt) arbeiten eng auf der Grundlage der in der Analyse erstellten und im Entwurf weiter spezifizierten Artefakte zusammen [Er08]. Der Entwurf wiederverwendbarer Softwareartefakte

erfordert daher die explizite Betrachtung der zugrunde liegenden Domäne [FG+02, WY+08]. Die Verfügbarkeit eines einheitlichen Begriffsverständnisses in Form von Domänenmodellen stellt demnach einen ersten Ansatzpunkt dar, den Entwurf der Managementdienste bezüglich Wiederverwendbarkeit zu unterstützen und für alle beteiligten Akteure nachvollziehbar zu gestalten.

Dabei ist es zunächst unerheblich, ob die erstellten Domänenmodelle konform zu einem einheitlichen Metamodell der Domäne sind oder nicht. Die Forderung nach Nachvollziehbarkeit im Sinne eines Referenzentwurfes (Anforderung 1.1) bedingt jedoch die abstrahierte Betrachtung der wesentlichen Elemente der Domäne. Damit wird sichergestellt, dass sowohl der Analyst, der Architekt und der Entwickler über das geforderte gemeinsame Verständnis verfügen, um betreiberprozessorientierte und wiederverwendbare Managementdienste entwerfen zu können. In dem von der vorliegenden Arbeit verfolgten Vorgehen wird daher angenommen, dass die erstellten Domänenmodelle konform zu einem Metamodell der Domäne sein müssen, in dem die wesentlichen Aspekte für den Entwurf wiederverwendbarer und betreiberprozessorientierter Managementdienste integriert sind. Dies wird durch bestehende Arbeiten bislang nicht betrachtet.

Im Folgenden werden die im weiteren Verlauf grundsätzlichen Begriffe (Domänenmodell und Domänenmetamodell, Betreiberprozessmodell und Managementdienstmodell) erläutert und in den Kontext des zugrunde gelegten dienstorientierten Entwicklungsprozesses eingeordnet.

Domänenmetamodell und Domänenmodelle

Die Grundlage für den Entwurf wiederverwendbarer Softwareartefakte ist ein einheitliches Verständnis über die zugrunde gelegte Domäne [Ne80, Ar89, WY+08]. Die Grundlage für diese strukturelle Modellierung bilden Domänenmodelle der betrachteten Domäne IT-Management, in denen die fachlichen Aspekte des betrachteten Bereiches dargestellt werden. Um sicherzustellen, dass eine einheitliche Auffassung über die Bedeutung der verschiedenen dargestellten Konzepte in den Domänenmodellen über die Grenzen einzelner Entwicklungsprojekte hinweg vorherrscht, ist ein Metamodell erforderlich, das gewisse Einschränkungen hinsichtlich der Konformität von zu diesem Metamodell konkreten Modellinstanzen vorgibt. Ein Metamodell definiert eine abstrakte Syntax sowie eine statische Semantik. Aufgrund des beschreibenden Charakters der Domänenmodelle und des Domänenmetamodells bietet sich die Definition einer Ontologie an [KK+06, De02, Be04, SW+10, FG+02, CJ+99, AZ+06].

Betreiberprozessmodelle

Während in den Domänenmodellen die strukturellen Aspekte dargestellt sind und hiermit vor allem die statischen Konzepte modelliert werden, ist zur Erfassung von dynamischen Abläufen die Konstruktion von Modellen der zu unterstützenden Betreiberprozesse erforderlich. Damit in den Modellen der Betreiberprozesse diejenigen Konzepte dargestellt werden, die durch die Modellierung der relevanten Aspekte der betrachteten Domäne erfasst wurden, wird die Modellierung der Betreiberprozesse auf Basis der zuvor erstellten Domänenmodelle durchgeführt. Hierbei können verschiedene Elemente der Domänenmodelle direkt übernommen werden (beispielsweise Teilnehmer, Handlungen, Datenobjekte), verschiedene Elemente bedürfen jedoch der weiteren Überarbeitung und

Anpassung. Insbesondere zählt hierzu die Modellierung der eigentlichen Ablaufhandlungen (temporale Aspekte, konditionale Aspekte), die nicht durch die Domänenmodelle erfasst wurden.

Managementdienstmodelle

Die Grundlage für die eigentliche Implementierung wiederverwendbarer Managementdienste bilden die in der Entwurfsphase erstellten Modelle der zu implementierenden Managementdienste. Hierbei wird von konkreten Technologien und Plattformen zunächst abstrahiert und der Fokus auf die Spezifikation abstrakter Dienstschnittstellen gelegt. In den Managementdienstmodellen wird eine genaue Festlegung auf die Signaturen der Dienstoperationen der einzelnen Dienstschnittstellen vorgenommen sowie die Nachrichten definiert, die beim Aufruf oder als Rückgabe einer Dienstoperation erforderlich sind.

4.2 Metamodell der Domäne IT-Management

Zur Überführung der fachfunktionalen Anforderung von informell spezifizierten Anforderungsartefakten in Analyseartfakte, die vom nachfolgenden Phasenschritt des dienstorientierten Entwurfes aufgegriffen werden können, wird ein formales Modell der betrachteten Domäne konstruiert. Ein Domänenmodell ist die Sammlung einheitlicher Abstraktionen zur Beschreibung der Umgebung, Anforderungen, Funktionen, Architektur und betrieblichen Randbedingungen eines Rechensystems für einen speziellen Anwendungsbereich [Ar89]. Von einem abstrahierten Standpunkt aus betrachtet stellt ein Domänenmodell daher die wesentlichen Konzepte dieser Domäne dar, blendet dabei aber diejenigen Details aus, die aus fachlicher Sicht irrelevant sind.

Ein Metamodell einer Domäne ist Teil einer Ontologie dieser Domäne, fokussiert jedoch lediglich die Bedeutung der einzelnen Elemente sowie deren strukturellen Beziehungen untereinander. Der Einsatz der Elemente des Metamodells ist nicht mehr originär Ziel des Metamodells, wird jedoch durch ergänzende Betrachtungen (z. B. durch die Spezifikation von Transformationen) zu einer Ontologie ergänzt. Die Unterstützung des Entwurfes von Managementdiensten auf Basis eines Modells der Domäne IT-Management hilft demnach, die für den Entwurf der Managementdienste notwendigen Eigenschaften zu erfassen und in einen Softwareentwurf zu überführen. Um die Qualität des fertigen Softwareproduktes sicherzustellen, wird ein integriertes Metamodell der Domäne zur Unterstützeng der dienstorientierten Analyse und des dienstorientierten Entwurfes benötigt. Hierdurch können die fachfunktionalen Anforderungen in der Analysephase ermittelt werden und der Entwurf einer dienstorientierten Lösung auf Basis dieser ermittelten Anforderungen zielgerichtet umgesetzt werden. Der durchgehende Einsatz von Modellen ermöglicht die Abstraktion von wichtigen Sachverhalten und bildet die Grundlage, ein ingenieursmäßiges Vorgehen zu formulieren. Somit wird es erst möglich, schon während des Entwurfes des zu erstellenden Softwaresystems Aussagen über bestimmte Eigenschaften zu treffen. Hierzu gehört beispielweise, eine Aussage über die Prozessorientierung oder die Wiederverwendbarkeit der entstehenden Managementdienste zu treffen.

Abbildung 16 zeigt diese Zusammenhänge auf und ordnet sie in den von der vorliegenden Arbeit fokussierten Entwicklungsprozess ein.

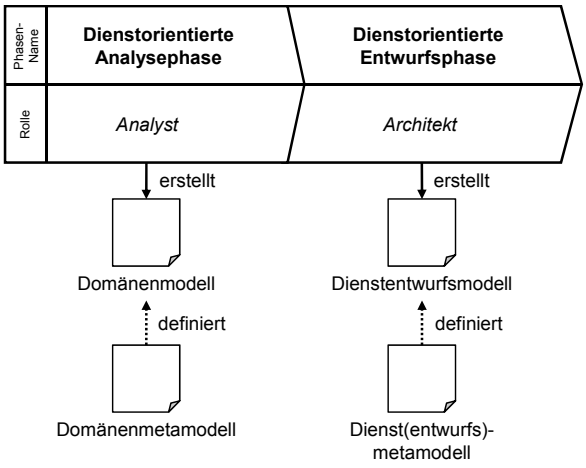

Abbildung 16 In der vorliegenden Arbeit fokussierter Aspekt im Entwicklungsprozess

Wesentliches Charakteristikum des betrachteten Ansatzes ist, dass durchz die Definition von Metamodellen die Komplexität der betrachteten Problemstellung auch teilweise auf den Konstruktionsschritt entsprechender notwendiger Modelle und Metamodelle gelenkt wird. Die Vorteile, beispielsweise durch eine einheitliche Beschreibung der Begriffe in der betrachteten Domäne zur Steigerung der Standardisierung abgeleiteter Softwarekomponenten, überwiegt heirbei jedoch die angesprochenen Nachteile. Der möglichst breite und umfassende Einsatz standardisierter Modellierungssprachen und -ansätze ist daher eine wesentliche Grundlage, um einen zielgerichteten Entwurf nachvollziehbar und somit in der Praxis anwendbar sicherzustellen.

4.2.1 Konzepte der Fachdomäne IT-Management

An einem einfachen Beispiel sollen die wesentlichen Konzepte der fachlichen Domäne IT-Management verdeutlicht werden. Hierzu wird, losgelöst von einer konkreten Betrachtung oder eines konkreten Szenarios, die Anforderung betrachtet, dass Mitarbeiter eines IT-Dienstleisters bei der Bearbeitung von bei Kunden des IT-Dienstleisters auftretenden Störungen behilflich sind. Hierzu nimmt der Mitarbeiter des IT-Dienstleisters die gemeldete Störung auf. Diese Information wird im Störungsbearbeitungswerkzeug (engl. *Trouble Ticket Tool,* auch: *Trouble-Ticket*-Werkzeug) abgelegt, damit jeder zukünftige, an der Entstörung beteiligte weitere Mitarbeiter des IT-Dienstleisters auf diese Information zurückgreifen kann. Da für die Bearbeitung der Störung im weiteren Verlauf verschiedene Managementwerkzeuge eingesetzt werden, soll der gesamte Prozess der Störungsbearbeitung auf Basis von zu konstruierenden Schnittstellen zu den einzelnen Werkzeugen automatisiert werden. Als zusätzliche Anforderung soll der Entwurf von vornherein die Möglichkeit eröffnen, in zukünftigen Erweiterungsprojekten weitere Managementfunktionen durch Nutzung der entwickelten Werkzeugschnittstellen umzusetzen.

Zur Verdeutlichung der Konzepte der Domäne sowie konkreter Instanzen hiervon ist auf Basis der genannten fiktiven Anforderung folgende, in Abbildung 17 dargestellte grafische Abstraktion dargestellt.

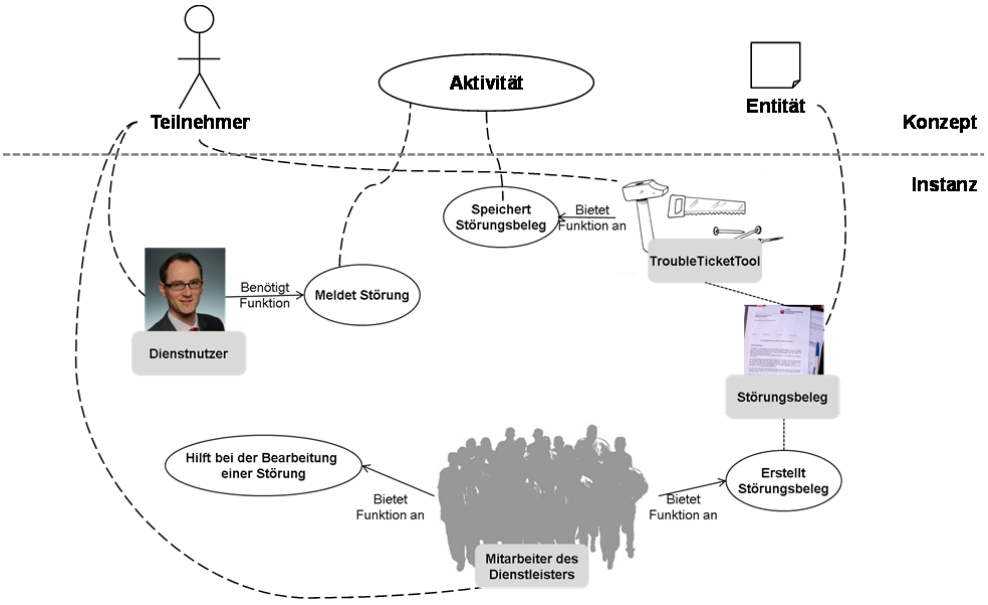

Abbildung 17 Abstrakte Darstellung grundlegender Konzepte der Domäne IT-Management

Die einzelnen Handlungstragenden (Dienstkunde, Mitarbeiter des IT-Dienstleisters, *Trouble Ticket Tool*) führen verschiedene Handlungen durch (Störung melden, Störung bearbeiten, Störungsbeleg erstellen, Störungsbeleg speichern). Die Speicherung der anfallenden Daten erfolgt in dem Datenblatt (Störungsbeleg). Die zur Darstellung dieser Konzepte notwendigen Modellelemente sind demnach der Teilnehmer, die Aktivität sowie die Entität. Diese wesentliche Abstraktion einer fachfunktionalen Anforderung bildet die Grundlage, um auf Basis eines integrierten Metamodells zunächst die Analyse dieser fachfunktionalen Anforderungen und darauf aufbauend den Entwurf von Diensten für die Domäne IT-Management (Managementdienste) zu beschreiben. Hierzu werden weitere Verfeinerungen eingeführt, die auf Basis der Analyse bestehender Arbeiten eine zielgerichtete Konstruktion dienstorientierter Architekturen ermöglicht.

4.2.2 Taxonomie der Domäne

Zunächst werden die einzelnen Elemente des Metamodells der Domäne konzeptionell, rein analytisch und auf Basis eines natürlich-sprachlichen Textes beschrieben. Hierdurch lassen sich die einzelnen Elemente einführen und erläutern. Grundlage für das Metamodell ist eine Analyse der Standardspezifikation von ISO/IEC20000-1:2005 [ISO05]. Die Entscheidung, eine Abstraktion auf Basis dieses Standards vorzunehmen, ist mit der Tatsache begründet, dass durch die Spezifikation ISO/IEC20000-1:2005 ein international anerkannter Entwurf für die Festlegung von funktionalen Mindestanforderungen an ein Managementsystem für den prozessorientierten Betrieb von IT-Diensten vorliegt [ISO05]. Diese Spezifikation eines Standards für diese Domäne kann demnach als ein wesentliches generisches Anforderungsartefakt aufgefasst werden, das die wesentlichen Charakteristika von fachfunktionalen Anforderungen vorgibt. Da zu erwarten ist, dass sowohl die Hersteller von Managementwerkzeugen als auch komplette Betreiberorganisationen sich an die Vorgaben dieses Standards orientieren werden, kann die Abstraktion der wesentlichen darin

enthaltenen Elemente die Grundlage zur Modellierung der Domäne bilden. Eine formale Darstellung dieser Elemente unterstützt den Softwareentwicklungsprozess daher zielführend, wenn auf Basis dieser formalen Abstraktion konkrete Modellinstanzen erstellt werden. Diese Modellinstanzen reflektieren die durch die Domäne vorgegebene Semantik der Beziehungen der einzelnen Elemente untereinander [PW+10, PP+10].

Die Erschaffung eines Metamodells für eine Domäne ist die Grundlage, um eine Wissensbasis (engl. *Knowledge Base*) für die Domäne zu erschaffen [GD+09]. Der Einsatz der Elemente dieser Wissensbasis durch die Festlegung von Regeln, die die Anwendung verschiedener Aspekte der betreffenden Elemente zum Ziel haben, ist die Grundlage für eine nachvollziehbare Methode, da somit wesentliche Einschränkungen bezüglich der Semantik der einzelnen Elemente vorgegeben werden.

Die Konstruktion des Metamodells der Domäne erfolgt durch mehrere Schritte. Zunächst wird eine textuelle Beschreibung der Elemente dieses Metamodells gegeben, um die Semantik der einzelnen Elemente zu verdeutlichen. Im weiteren Verlauf werden diejenigen Teile des Metamodells durch eine konkrete Syntax definiert, die nicht auf Basis bereits verfügbarer standardisierter Modellierungssprachen umgesetzt werden können. Hierzu gehört vor allem der Einsatz der SoaML, um die durch die dienstorientierte Analyse identifizierten Dienstkandidaten in den Phasenschritt des dienstorientierten Entwurfes zu überführen.

Die Reihenfolge der Beschreibung der einzelnen Elemente spielt keine Reihenfolge. Zur Wahrung eines Mindestmaßes an syntaktischem Verständnis folgt der Aufbau einer einzelnen Beschreibung jedoch einer einfachen und klar definierten Konvention. Die Festlegung auf eine Konvention entspricht weitestgehend der Festlegung auf eine Art gemeinsames Verständnis zur Definition des Vokabulars. Dadurch erhält das Vokabular den Charakter einer Taxonomie und stellt die Grundlage für die Formulierung einer Ontologie dar:

- **Deutscher Begriff** (in Klammern der **englische Begriff**): Die Festlegung des deutschen Begriffes stellt eine verbindliche Syntaxdefinition dar, da anhand des deutschen Begriffes die Klassifizierung bzw. die Relation zu weiteren Elementen definiert wird. Bei der Überführung der Konzepte des Metamodells in ein formal definiertes Metamodell wird der englische Begriff zur Identifikation des Elementes verwendet.

- **Beschreibung**: Die Beschreibung ist erklärender Natur und verdeutlicht den Nutzen bzw. den Einsatzzweck des Modellelementes. Weiterhin werden in der Beschreibung eventuell angegebene Relationen oder Klassifizierungen erläutert.

- Relation **Bezieht sich auf**: Aus der Beziehung von Elementen innerhalb eines Modells lassen sich bestimmte Eigenschaften ableiten. Ist hinreichend detaillierte Information vorhanden, so kann diese gleich im Metamodell festgehalten werden, um somit spätere Entwurfsentscheidungen zu unterstützen.

- Klassifizierung **Setzt sich zusammen aus** und Klassifizierung **Gehört zu**: Eine besondere Form der Relation stellt die Zusammensetzung eines Sachverhaltes dar, sagt sie doch etwas über eine enge Beziehung zweier Modellelemente aus. Ist über die alleinige Tatsache der Relation **Bezieht sich auf** hinaus hinreichend detaillierte Information vorhanden, wird die Relation klassifiziert, indem das Ganze der Zusammensetzung die einzelnen Elemente mittels der Klassifizierung **Setzt sich**

zusammen aus definiert wird bzw. das einzelne Element eines Ganzen auf das Ganze mittels der Klassifizierung **Gehört zu** beschrieben wird.

- Klassifizierung **Abstrahiert** und Klassifizierung **Verfeinert**: Sind bei mindestens zwei Modellelementen gemeinsame Eigenschaften feststellbar, können diese Eigenschaften in einem gemeinsamen Modellelement zusammengeführt werden. Hierdurch reduziert sich der spätere Aufwand bei der Umsetzung, da die Möglichkeit, Operationen wiederzuverwenden, erhöht wird. Die Klassifizierung **Abstrahiert** bezieht sich auf das Modellelement, das gemeinsame Eigenschaften zweier oder mehr Elemente zusammenfasst; die inverse Klassifizierung **Verfeinert** beschreibt die Beziehung eines konkreten Elementes zum abstrahierten Element.

Grundlegende Elemente

Im Folgenden werden nun auf Basis der vorangegangenen Konvention die einzelnen Elemente des integrierten Metamodells für die dienstorientierte Analyse und den dienstorientierten Entwurf beschrieben. Diese vorgestellten Modellelemente bilden die Grundlage, um mittels weiterer Ergänzungen ein formales Metamodell der Domäne aufstellen zu können. Dieses formale Metamodell definiert letztlich die für den Entwurf von Managementdiensten zugrunde liegende Syntax und Semantik. Damit kann zum einen der Entwurf von Diensten anhand von abstrahierten Modellen diskutiert werden, zum anderen können aber auch Metriken angewandt werden, um die Übereinstimmung von Dienstentwurf und fachlichem Anforderungsmodell zu bestimmen.

Managementbereich (*Management Area*)

> **Beschreibung**: Ein Managementbereich ist innerhalb der Domäne IT-Management ein abgeschlossener Bereich, wobei klare Grenzen zu anderen Bereichen gezogen werden können. Der Managementbereich steht demnach stellvertretend für einen einzelnen Managementprozess und beinhaltet daher alle weiteren Modellelemente. Durch die Standardspezifikation ISO/IEC20000-1:2005 [ISO05] werden insgesamt dreizehn verschiedene Managementbereiche vorgegeben: *Incident Management, Problem Management, Release Management, Change Management, Configuration Management, Service Level Management, Capacity Management, Continuity Management, Information Security Management, Budgeting and Accounting, Supplier Management, Business Relationship Management.*
>
> **Setzt sich zusammen aus**: Managementeilnehmer, Managementaktivität, Managemententität, Managemententitätszugriffsrichtlinie, Managemententitätsstrukturrichtlinie, Management-fähigkeit

Managementaktivität (*Management Activity*)

> **Beschreibung**: Eine Managementaktivität ist eine konkrete Aufgabe, die durchgeführt wird, um ein bestimmtes, definiertes Managementziel zu erreichen. Während der Begriff Managementfähigkeit also zunächst bewusst von der Eigenschaft des Aktivhandelns

abstrahiert, wird dies durch das Modellelement Managementaktivität festgelegt. Typische Managementaktivitäten umfassen die Überwachung von Infrastrukturkomponenten, aber auch die Behandlung von Fehlern, die Erstellung von Berichten, die Konfiguration von Elementen oder die Abarbeitung von Nutzeranfragen. Managementaktivitäten werden von Managementteilnehmern durchgeführt und operieren auf Managemententitäten. Wird eine Managementaktivität selbstständig von einem Managementwerkzeug durchgeführt (beispielsweise die Überwachung einer Netzkomponente durch einen aktiven Netzwerkerreichbarkeitstest), wird das entsprechende Managementwerkzeug somit ebenfalls durch einen Managementteilnehmer dargestellt.

Gehört zu: Managementbereich

Bezieht sich auf: Managementteilnehmer, Managemententität

Managementfähigkeit (*Management Capability*)

Beschreibung: Für eine erste textuelle Analyse von Anforderungen an wiederverwendbare Managementdienste wird mit dem von konkreten Managementaktivitäten abstrahierenden Konzept der Managementfähigkeit ein dediziertes Element definiert. Managementfähigkeiten werden dabei im weiteren Sinne von unterschiedlichen Managementbereichen angeboten und entsprechen somit einer definierten Zielsetzung eines bestimmten Managementbereiches. Weiterhin abstrahiert eine Managementfähigkeit vom Konzept der Managementaktivität dahingehend, dass hierdurch eine Einschränkung auf den genauen Handlungsbereich einer Managementaktivität im weiteren Verlauf des Entwicklungsvorgehens festgelegt werden kann. Dieses Konzept ist grundlegend, um von den eigentlichen Aktivitäten zu abstrahieren und hiermit die Möglichkeit zu eröffnen, bestehende Managementwerkzeuge im weiteren Verlauf des Entwicklungsprozesses als Implementierung identifizierter Dienste aufzufassen und in einem ganzheitlichen Ansatz zu integrieren. Dieser Sachverhalt wird genutzt, um die Integration bestehender Werkzeuge zu ermöglichen, indem die verschiedenen Konzepte hierdurch auf Basis eines einheitlichen Ansatzes zur Beschreibung der Semantik zueinander in Bezug gebracht werden. Das Modellelement der Managementfähigkeit eröffnet demnach die Möglichkeit, semantische Konzepte auf Basis des Metamodells der Domäne zu nutzen und in einem integrierten Gesamtvorgehen einzusetzen.

Bezieht sich auf: Managementaktivität

Gehört zu: Managementbereich

Managemententität (Management Entity)

Beschreibung: Die Abstraktion von Objekten innerhalb der Domäne IT-Management, die als Informationsspeicher den Zustand einer Prozessausführung speichern, wird als Managemententität definiert. Eine Managemententität ist demnach ein Objekt, das den Zustand der Ausführung einer Managementaktivität widerspiegelt. Entitäten unterliegen gewissen Vorgaben bezüglich ihres Aufbaus und bezüglich des Zugriffes auf die Entität. Diese Vorgaben sind nötig, um Sicherheitsanforderungen bezüglich des Zugriffs auf die durch die

Entität repräsentierte Zustandsinformation umzusetzen. Beispielsweise soll nicht jedem Mitarbeiter einer Betreiberorganisation Zugriff auf kritische Information von rechtlich bindenden Verträgen ermöglicht werden. Managemententitäten sind Ziele von Managementaktivitäten. Eine Managemententität kann als Abstraktion eines *Managed Object* aufgefasst werden.

Gehört zu: Managementbereich

Bezieht sich auf: Managementaktivität, Managementzugriffsrichtlinie, Managementstrukturrichtlinie

Managementteilnehmer (Management Participant)

Beschreibung: Ein Managementteilnehmer ist ein Akteur innerhalb eines Managementbereiches, der durch die Ausführung von Managementaktivitäten den Zustand von Managemententitäten verändern kann. Hierbei regeln Richtlinien den Zugriff auf die entsprechenden Managemententitäten. Managementteilnehmer können stellvertretend für verschiedene Typen von Akteuren stehen (menschliche Akteure, technische Komponenten von Informationssystemen), was für den Entwurf von Managementdiensten von einem prinzipiellen Standpunkt aus betrachtet jedoch nicht relevant ist. So kann die Integration bestehender Managementwerkzeuge durch Modellierungselemente des Metamodells unterstützt werden, indem ein bestehendes Managementwerkzeug als Managementteilnehmer, die von dem Managementwerkzeug angebotenen Funktionalitäten als Managementfähigkeiten und, falls nötig, die durch das Managementwerkzeug manipulierten Daten als Managemententität modelliert wird.

Ferner können durch eine Analyse verschiedener *Best Practices* die drei grundlegende Rollen der Teilnehmer in einem Managementprozess identifiziert und vorgegeben werden: *Process Owner, Process Manager, Process Participant.* Hierdurch können grundlegende Aspekte für eventuell notwendige Eskalationsschritte innerhalb eines Managementprozesses definiert und erfasst werden.

Gehört zu: Managementbereich

Bezieht sich auf: Managementaktivität, Managemententitätszugriffrichtlinie

Managementrichtlinie (Management Policy)

Beschreibung: Managementrichtlinien stellen ein wesentliches Anforderungsartefakt für den Entwurf eines Managementsystems dar. Richtlinien können in verschiedenen Ausprägungen identifiziert werden und ermöglichen, strukturelle Sachverhalte durch nicht-invasive Erweiterungen an den betrachteten Subjekten zu realisieren. Der Zweck der Einführung eines eigenständigen Metamodellelementes liegt darin begründet, einen einfach erweiterbaren Mechanismus im Metamodell zu integrieren, um spätere erweiterte Anforderungen umsetzen zu können. Wird beispielsweise auf Basis des hier vorgestellten Metamodells die Umsetzung

von qualitativen Anforderungen (z. B. Sicherheitsanforderungen) betrachtet, kann dies durch eine einfache Erweiterung der hier vorgestellten Konzepte realisiert werden, ohne jedoch grundlegend das vollständige Metamodell neu zu konstruieren. Am Beispiel einer durch den Standard ISO/IEC20000-1:2005 vorgegeben Richtlinie (bezüglich des Aufbaus von Managemententitäten) wird demonstriert, wie dieses Konzept für den fachlich-motivierten Entwurf eingesetzt werden kann.

Abstrahiert: Managemententitätsstrukturrichtlinie

Ergänzung zur Identifikation von Managementdienstkandidaten

Da die Zielsetzung des Metamodells dem Entwurf wiederverwendbarer Managementdienste folgt, werden bei den Überlegungen zu den wesentlichen Konzepten im Folgenden nun die Elemente des Modells eingeführt, die die Grundlage für den späteren Entwurf von Managementdiensten bilden. Diese Grundlage ist die Identifikation von möglichen Dienstkandidaten. Die Integration beider Aspekte (Modellierung der Domäne und Identifikation von Dienstkandidaten) in ein gemeinsames Modell bringt den Vorteil, dass sowohl die Semantik der strukturellen Beziehungen der Domänenelemente erfasst werden können, als auch gleichzeitig eine Beziehung zu den die fachlichen Anforderungen umsetzenden Managementdiensten hergestellt werden kann. Hierbei wird die von Erl eingeführte Notation, den ersten Entwurf möglicher Dienste als Dienstkandidat zu bezeichnen, übernommen. Ein Dienstkandidat ist demnach ein in der dienstorientierten Analyse identifizierter Dienst, der jedoch noch nicht durch die Schritte des dienstorientierten Entwurfes spezifiziert ist.

Angebotene Managementfähigkeit (Provided Management Capability)

Beschreibung: Die angebotene Managementfähigkeit stellt eine von einem Managementdienst und damit auch eine von einem Managementbereich zur Verfügung gestellte Fähigkeit bereit, ein dediziertes Ziel zu erreichen. Hierzu wird die zugrunde liegende Logik abstrahiert, damit das Ergebnis der Ausführung der zugrunde liegenden Logik durch die angebotene Managementfähigkeit abstrahiert wird.

Verfeinert: Managementfähigkeit

Benötige Managementfähigkeit (Required Management Capability)

Beschreibung: Um die fachlichen Anforderungen zu bündeln und unabhängig von konkreten durchzuführenden Managementaktivitäten darstellen zu können, wird als inverses Element zur angebotenen Managementfähigkeit das Konzept der benötigten Managementfähigkeit eingeführt.

Verfeinert: Managementfähigkeit

Managemententitätsstrukturrichtlinie (Management Entity Structure Policy)

Beschreibung: Eine Managemententitätsstrukturrichtlinie beschreibt den gewöhnlichen und zweckmäßigen Aufbau einer Managemententität. Die Erfassung dieser Anforderung ist grundlegende Voraussetzung, um den späteren nachrichtenorientierten Zugriff auf die den Zugriff auf die Entitäten regelnde Managementbasisdienste realisieren und eine Integration bestehender Werkzeuge in diesen integrierten Gesamtansatz von einer fachlichen Sicht her motivieren zu können.

Gehört zu: Managementbereich

Bezieht sich auf: Managemententität

Verfeinert: Managementrichtlinie

Basismanagementaktivität (Basic Management Activity)

Beschreibung: Basismanagementaktivitäten bilden die Grundlage, um Managementhandlungen durchzuführen. Eine Basismanagementaktivität charakterisiert sich dadurch, dass nicht offensichtlich ist, ob sich diese Aktivität aus weiteren identifizierten Aktivitäten zusammensetzt oder ob diese Aktivität bezüglich ihres betreffenden Kontextes vollständig atomar durchgeführt werden kann. Als Basisaktivitäten werden in der Regel diejenigen Managementaktivitäten identifiziert, die keinen direkten Bezug zu den konkreten Prozesskontexten besitzen. Dies sind beispielsweise Managementaktivitäten, die direkt die Manipulation von Managemententitäten betreffen.

Gehört zu: Zusammengesetzte Managementaktivität (nicht zwangsläufig)

Verfeinert: Managementaktivität

Zusammengesetzte Managementaktivität (Composed Management Activity)

Beschreibung: Zur Erfüllung von bestimmten Zielen sind in der Regel mehrere Aktivitäten in Form von aufeinanderfolgenden oder zeitgleichen Schritten notwendig. Dieser Sachverhalt muss, bei bekannter Tatsache, auch in entsprechenden Instanzen des Domänenmetamodells zum Ausdruck gebracht werden können. Demnach ist die Existenz eines entsprechenden Metamodellelementes erforderlich. Zusammengesetzte Managementaktivitäten stellen demnach eine Komposition von Basismanagementaktivitäten dar.

Setzt sich zusammen aus: Basismanagementaktivität

Verfeinert: Managementaktivität

Managementdienstkandidat (Management Service Candidate)

Beschreibung: Ein Managementdienst stellt Funktionalitäten bereit, um die durch das Modellelement Managementfähigkeit erfasste Anforderungen umzusetzen. Das Element Managementdienst ist ein generisches Element, das eine genaue Bedeutung durch die Festlegung eines konkreten Typs von Managementdienst erhält.

Bezieht sich auf: Managementaktivität

Abstrahiert: Managementprozessdienstkandidat, Managementbasisdienstkandidat

Managementprozessdienstkandidat (Management Process Service Candidate)

Beschreibung: Ein Managementprozessdienst realisiert fachfunktionale Anforderungen durch Komposition verschiedener Managementbasisdienste oder weitere Managementprozessdienste. Da bei der Umsetzung von geschäftlichen Abläufen neben entsprechenden Referenzmodellen immer auch Anpassungen an konkrete Situationen vorzunehmen sind, zeichnet sich diese Klasse von Managementdiensten durch ein im Vergleich zur Klasse der Managementbasisdienste geringeres Maß an Wiederverwendbarkeit aus.

Verfeinert: Managementdienstkandidat

Managementbasisdienstkandidat (Management Basic Service Candidate)

Beschreibung: Ein Managementbasisdienst stellt grundlegende Managementfunktionalitäten bereit und ist unabhängig von konkreten Managementprozessen in unterschiedlichen Szenarien einsetzbar. Ein Managementbasisdienst stellt beispielsweise Zugriffsfunktionen auf Managemententitäten bereit, um den Status der Ausführung von Managementprozessen zu dokumentieren oder Funktionen für den Zugriff auf Konzepte des technischen Betriebes von IT-Infrastrukturen (Überwachung von *ManagedObjects*). Diese Klasse von Diensten zeichnet sich durch ein im Vergleich zu Managementprozessdiensten höheres Maß an Wiederverwendbarkeit aus.

Verfeinert: Managementdienstkandidat

Die hier vorgestellte Taxonomie bildet die Grundlage, um ein formales Metamodell der Domäne für den Entwurf von betreiberprozessorientierten und wiederverwendbaren Managementdiensten zu definieren.

4.2.3 Abstrakte Syntax des Domänenmetamodells

Die Zusammenfassung der durch die erstellte Taxonomie betreffenden Elemente eines ersten Entwurfes für ein Metamodell der Domäne IT-Management ist in Abbildung 18 dargestellt. Das Modell beschreibt gemäß [SV+07] die relevanten Konzepte der Domäne und kann daher als abstrakte Syntax eines weiter zu spezifizierenden Metamodells betrachtet werden. Auf Basis dieses ersten

Entwurfes werden im weiteren Verlauf zusätzliche Verfeinerungen eingeführt, damit ein ausdrucksstarkes Metamodell zur Unterstützung einer zielführenden dienstorientierten Analyse die Grundlage für ein durchgehendes Entwicklungsvorgehen gebildet werden kann.

Im Wesentlichen entspricht die hier vorgestellte Klassifizierung von Managementdienstkandidaten in die Typen Managementbasisdienstkandidat (`Management Basic Service Candidate`), Managementprozessdienstkandidat (`Management Process Service Candidate`) den in bekannten Klassifizierungsschemata (z. B. [Co07, Er08, Fa08]) vorkommenden Einordnungen. Um einen einheitlichen Bezug zwischen den in einer dienstorientierten Softwarelösung für das IT-Management vorkommenden Diensten und den zum Entwurfszeitpunkt bekannten Typen von Diensten herstellen zu können, werden die bekannten Konzepte daher im Wesentlichen übernommen und vor dem Hintergrund der genannten Problemstellungen auf die eigene Lösung angepasst.

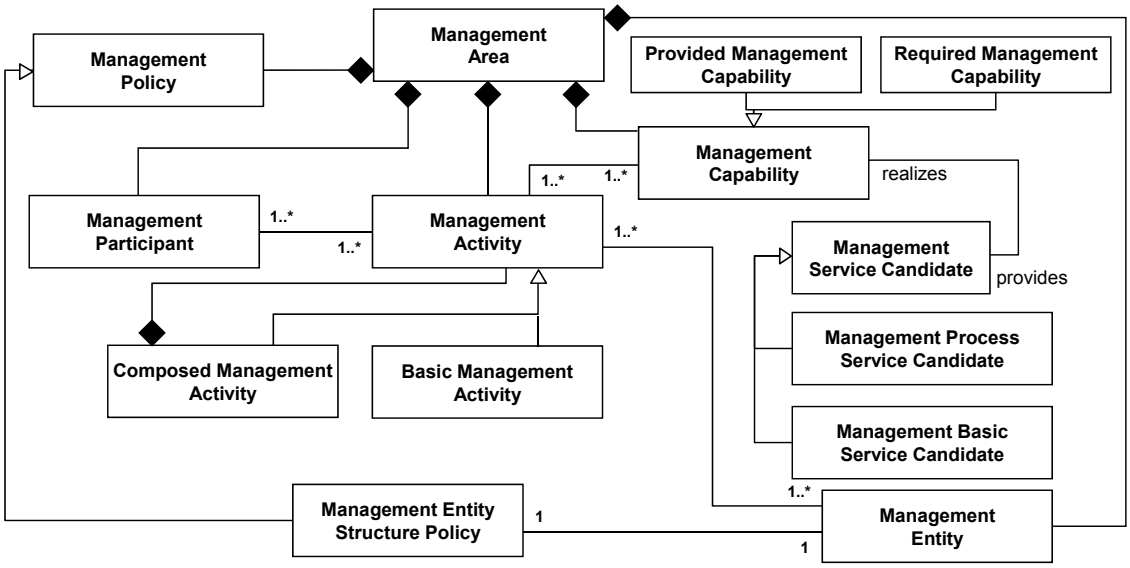

Abbildung 18 Abstrakte Syntax des Modells zur Spezifikation von Managementdiensten

Um die fachlichen Anforderungen bezüglich des Datenformates derjenigen Elemente spezifizieren zu können, die die eigentliche Managementinformation (Störungsbeleg, Ressourcenzustandsobjekt etc.) speichern, kann das Element `Management Entity Structure Policy` genutzt werden. Das Element `Management Policy` stellt eine Abstraktion dieser beiden Richtlinienelemente dar. Eine Gruppierung der verschiedenen Elemente eines Domänenmodells kann durch die Eingrenzung auf eine von dreizehn verschiedenen Managementbereichen durch das Element `Management Area` durchgeführt werden. Benötigte und durchgeführte Managementaktivitäten werden durch das Konzept der Managementfähigkeit (`Management Capability`) dargestellt. Dieses Element stellt den Anknüpfungspunkt zwischen strukturellem Domänenmetamodell und dem Metamodell zur Identifizierung von Managementdienstkandidaten dar.

Das Modellelement `Management Service Candidate` erhält eine Assoziation zum Modellelement `Management Capability`. Dadurch wird ein klarer Bezug zwischen

umzusetzenden Managementfähigkeiten, und den eine oder mehrere Managementfähigkeiten umsetzenden Managementdienstkandidaten hergestellt. Diese Managementdienstkandidaten werden konkretisiert, indem die unterschiedlichen Aspekte bei der Identifizierung in einer Unterscheidung zwischen den Basisdienstkandidaten (`Management Basic Service Candidate`), den auf der Grundlage der identifizierten Basisdienstkandidaten komponierten Prozessdienstkandidaten (`Management Process Service Candidate`) und den Dienstkandidaten für die Umsetzung von weiteren, nicht aus dem fachfunktionalen Entwurf identifizierbaren Dienstkandidaten münden.

4.2.4 Definition der Metamodelle

Die Zielsetzung des verfolgten Ansatzes besteht in der praktischen Anwendbarkeit in einem integrierten Entwicklungsvorgehen. Hierzu ist eine Spezifikation der erarbeiteten integrierten konzeptionellen Modelle mit den Mitteln eines formal definierten Metamodells notwendig, um eine konkrete Syntax für die vorgestellte abstrakte Syntax angeben zu können. Erst durch die Festlegung auf eine konkrete Syntax können die vorgestellten Konzepte in realen Entwicklungsprojekten eingesetzt und durch Entwicklungsumgebungen unterstützt werden.

Prinzipiell existieren vielfältige Möglichkeiten, abstrakte Konzepte formal zu spezifizieren. Vor allem die UML-basierten Ansätze (Definition eines formalen MOF-basierten Metamodells zur Nutzung in UML-Werkzeugen [OMG-MOF], Definition eines UML-Profils zur Nutzung in UML-Werkzeugen [OMG-UML-Infra]) erfreuen sich in der Vergangenheit wachsender Beliebtheit. Beiden Ansätzen ist jedoch gemein, dass eine formale Definition der den umzusetzenden Konzepten zugrunde liegenden Metamodellelementen unumgänglich ist, wenn der falsche Einsatz der jeweiligen Metamodelle verhindert werden soll. Daher muss neben der Definition der abstrakten Syntax der darzustellenden Metamodellelemente die Definition der statischen und dynamischen Semantik erfolgen. Verschiedene Arbeiten in der Forschungsgruppe ([Me06], [Em08], [Li09], [Mo09]) haben gezeigt, dass dies zwar prinzipiell möglich ist, jedoch einen erheblichen Aufwand nach sich zieht und durch entsprechende Entwicklungswerkzeuge unterstützt werden muss [HT06].

Da durch die Zielsetzung der Arbeit, durch die Integration bestehender Managementwerkzeuge eine rückwärts-orientierte Herangehensweise bei der Schaffung einer dienstorientierten Architektur im IT-Management zu unterstützen vor allem der Aspekt der vielfältigen verschiedenartigen Managementwerkzeuge und deren unterschiedlich beschriebene Schnittstellen zu beachten ist, wird die Spezifikation eines Metamodells zur Unterstützung der Domänenmodellierung in Form einer Ontologie vorangetrieben [Be04]. Hierbei kommt vor allem der beschreibende Charakter von Ontologien zum tragen, da durch die Konkretiesierung durch den Einsatz von OWL im weiteren Verlauf der vorliegenden Arbeit eine formalisierte Grundlage auf der Basis einer Prädikatenlogik eingeführt wird [AZ+06].

In nachfolgender Abbildung 19 ist dieser Sachverhalt dargestellt, wobei eine Einordnung in die von der OMG vorgestellte Metapyramide vorgenommen wird [OMG-MDA].

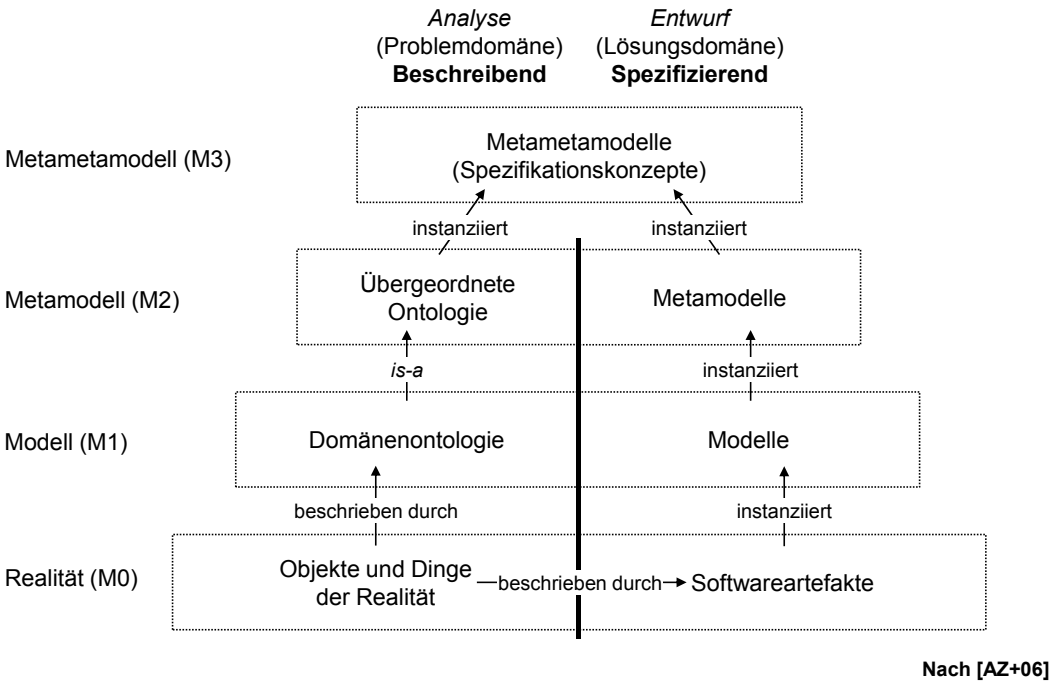

Abbildung 19 Einsatz von Ontologien in der Analysephase

Die aus der von der OMG vorgestellte Metapyramide [OMG-MDA] mit deren bekannten Konzepte wie beispielsweise Modell, Metamodell oder Metametamodell kann nach [AZ+06] als Grundlage genommen werden, die verschiedenen Konzepte bei der Ontologie-basierten Analyse in einem Softwareentwicklungsprozess einzuordnen. Vergleichbar zu der bekannten Einordnung in die verschiedenen Abstraktionsebenen M0 bis M3 existieren beim Einsatz von Ontologien verschiedene Ausprägungen, wobei besonders hervorzuheben ist, dass keine klare Trennung in der konkreten Ausprägung zwischen Instanzen von Metamodell- und Modellelementen bei der Ontologie-basierten Vorgehensweise herrscht. Dies ermöglicht vor allem die Bearbeitung von Modell und Metamodell gleicherzeitig, was bespielsweise in den verschiedenen Entwicklungswerkzeugen für Ontologien (z. B. in Protégé [Protege]) unterstützt wird.

Hieraus lässt sich vor dem Hintergrund der eingesetzen Modellierungsansätze eine klare Festlegung treffen, wie die verschiedenen Modellierungssprachen über Analyse- und Entwurfsphase hinweg in das betrachtete Entwicklungsvorgehen eingeordnet werden können. Da durch die Anforderungen der vorliegenden Arbeit gefordert wird, dass der vorgestellte Ansatz in der Praxis Anwendung finden kann, ist die durchgehende Überführung der verschiedenen, im Kontext des Entwicklungsvorgehens erstellten Artefakte sicher zu stellen. Dies wird in Abbildung 20 verdeutlicht, indem die Beziehungen der einzelnen Artefakte untereinader durch die Angabe entweder durch eine Instanz-Beziehung oder durch eine Modellüberführung-Beziehung dargestellt sind.

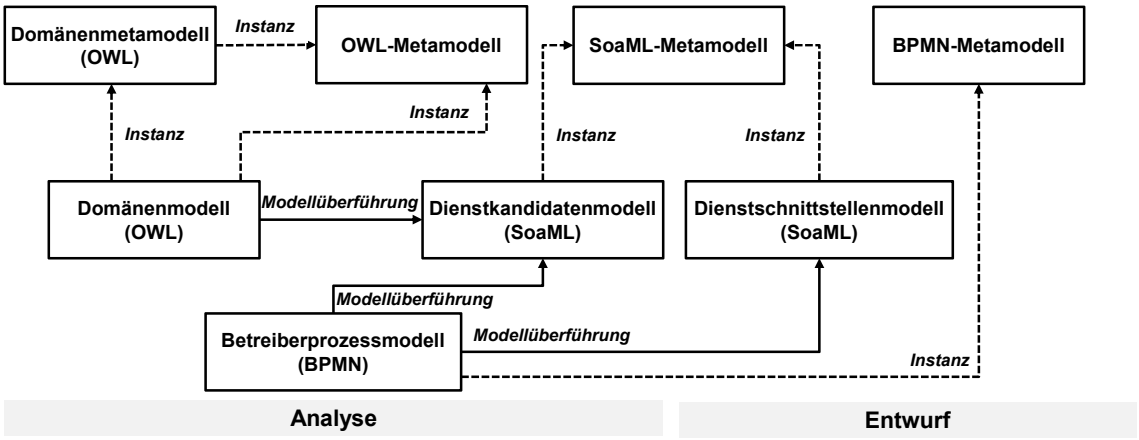

Abbildung 20 Instanzen und Modellüberführungen im Entwicklungsvorgehen

Diese Ontologie wird im Folgenden durch den Einsatz der Sprache OWL spezifiziert, sodass die Semantik der einzelnen Modellelemente klar definiert ist. Der Vorteil bei der Nutzung einer OWL-Ontologie liegt in der Möglichkeit, durch den Einsatz von Klassifikationsalgorithmen eine automatisierte Generierung von Wissen durchzuführen. Dies kann genutzt werden, um beispielsweise semantisch äquivalente Definitionen von Informationsmodellen in verschiedenen Managementwerkzeugen automatisiert zu generieren.

Modellierung der Domäne

Da aus Anforderung 1.1 (Seite 39) folgt, dass der Entwurf wiederverwendbarer Managementdienste auf Basis eines nachvollziehbaren und durch den Einsatz standardisierter Modellierungssprachen durchführbaren Vorgehens erfolgt, wird das im vorigen Abschnitt erarbeitete konzeptionelle Metamodell der Domäne im weiteren Verlauf konkretisiert. Hierzu wird, wie aus Abbildung 20 ersichtlich, die Definition einer OWL-Ontologie [W3C-OWL] angestrebt. Somit können verschiedene Vorteile dieses Ansatzes genutzt werden, wie beispielsweise die Verifikation der Konformität zum Metamodell von erstellten Domänenmodellen oder die automatisierte Generierung von Wissen, wenn semantische Bezüge in den erstellten Domänenmodellen nicht vollständig definiert sind.

Definition des Domänenmetamodells

Die Spezifikation des Metamodells zur strukturellen Analyse der Domäne IT-Management verfeinert und konkretisiert die eingeführten Konzepte. Durch die Festlegung von OWL-Restriktionen können zusätzliche Einschränkungen an die Modellelemente formuliert werden, um im Einsatz des Metamodells zu verhindern, dass semantisch falsche Modelle erstellt werden. Durch die Festlegung der statischen Semantik kann überhaupt erst eine Verifikation der Konformität der erstellten Modellinstanzen bezüglich des Metamodells erfolgen.

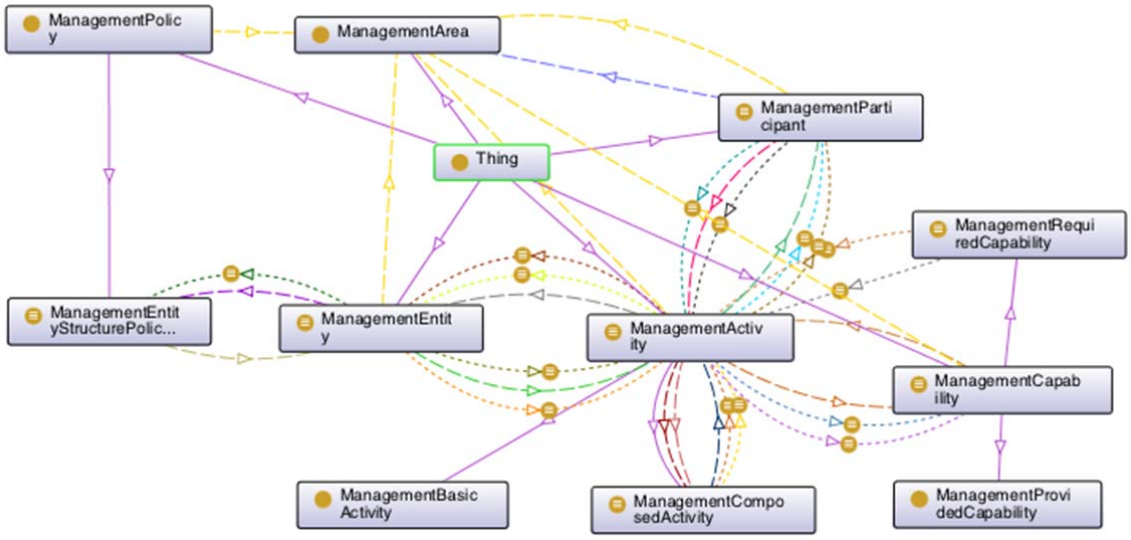

Abbildung 21 Spezifiziertes Metamodell der Domäne als OWL-Ontologie

Abbildung 21 stellt das vollständig spezifizierte Metamodell dar. Auf Basis dieses einheitlichen Metamodells können Referenzmodelle für einzelne Managementbereiche erstellt werden. Diese Referenzmodelle vereinfachen die Integration bestehender Werkzeuge, da dadurch definierte Integrationspunkte vorgegeben werden. Dies hilft, Mehrdeutigkeiten zu reduzieren. Am Beispiel typischer, in der Störungsbearbeitung beteiligter Managementbereiche wird bei der Vorstellung einer systematischen Methode zur Anwendung des Metamodells der Entwurf spezifischer Ontologien in Abschnitt 5.2.1 (Seite 126 ff.) präsentiert.

Im Folgenden werden die OWL-Restriktionen der einzelnen Modellelemente definiert, die notwendig sind, um auf Basis der vorgestellten Ontologie konforme Modellinstanzen bilden zu können. Die dargestellten Einschränkungen liefern einen Beitrag, um die verschiedenen Konzepte der Domäne formal zu beschreiben, und wurden in [PR+11] publiziert sowie als Mitwirkung in dem Beitrag [Re11] veröffentlicht.

Klassen

Die Definition der einzelnen Modellelemente des Metamodells erfolgt in einer OWL-Ontologie durch die Definition einzelner OWL-Klassen. Im Folgenden werden daher zunächst die einzelnen OWL-Klassen definiert sowie Einschränkungen festgelegt, bezüglich derer gültige Instanzen identifiziert werden können.

Management Area

Das Modellelement `ManagementArea` ist ein zentrales Element, um verschiedene fachliche Bereiche innerhalb einer Betreiberorganisation untergliedern zu können.

```
Equivalent classes:
(contains some ManagementActivity)
 and (contains some ManagementParticipant)
 and (contains only
   (ManagementActivity
    or ManagementCapability
    or ManagementEntity
    or ManagementParticipant
    or ManagementPolicy))
```

Quelltext 1 OWL-Einschränkungen des Managementbereiches

Die Einschränkungen, die sich aus der Taxonomie ergeben, legen fest, dass ein Managementbereich mindestens eine Managementaktivität und einen Managementteilnehmer beinhalten muss, damit ein Modellelement als Managementbereich klassifiziert werden kann. Die restlichen Einschränkungen legen die Zugehörigkeit der im Domänenmetamodell identifizierten strukturellen Modellierungselemente bezüglich eines betrachteten Managementbereiches fest.

Management Activity

Managementaktivitäten sind die zentralen Modellelemente innerhalb der dienstorientierten Analyse, um die einzelnen Handlungen innerhalb eines Managementszenarios zu beschreiben.

```
Equivalent classes:
(hasParticipant some ManagementParticipant)
 and (operatesOn some ManagementEntity)
 and (requires some ManagementCapability)
 and (hasParticipant only ManagementParticipant)
 and (operatesOn only ManagementEntity)
 and (requires only ManagementCapability)

Super classes:
(isPartOf only ManagementArea)
 and (isUsedBy only ManagementComposedActivity)
 and (isPartOf exactly 1 ManagementArea)
 and (hasDescription some string)
 and (hasDescription only string)
 and (ManagementBasicActivity or ManagementComposedActivity)
```

Quelltext 2 OWL-Einschränkungen der Managementaktivität

Die verschiedenen Einschränkungen, die die Semantik des Elementes genau festlegen, ergeben sich aus der Taxonomie der Domäne. Hinreichende Bedingung ist die die Existenz einer Assoziation zu Managementteilnehmern, die Manipulation einer Managemententität oder das Benötigten einer Managementfähigkeit.

Management Basic Activity

Basismanagementaktivitäten weisen keine weiteren – außer die von der Oberklasse Managementaktivität geerbten – Einschränkungen auf.

Management Composed Activity

Neben den ohnehin von der Oberklasse geerbten Einschränkungen weist die zusammengesetzte Managementaktivität eine zusätzliche Einschränkung auf. Diese zusätzliche Einschränkung legt fest, dass eine Managementaktivität eine zusammengesetzte Managementaktivität ist, wenn sie eine `isComposedOf`-Assoziation zu einer weiteren Managementaktivität aufweist.

Equivalent classes:
(isComposedOf some ManagementActivity)
and (isComposedOf only ManagementActivity)

Quelltext 3 Einschränkungen bei den zusammengesetzten Managementaktivitäten

Management Capability

Die Einschränkungen bei der Klasse *Management Capability* legen im Wesentlichen fest, dass eine Managementfähigkeit genau einem Managementbereich zugeordnet wird sowie eine Beschreibung in Freitextform aufweist.

Equivalent classes:
ManagementProvidedCapability or ManagmentRequiredCapability

Super classes:
(isPartOf only ManagementArea)
and (isPartOf exactly 1 ManagementArea)
and (hasDescription some string)
and (hasDescription only string)

Quelltext 4 Einschränkungen der Managementfähigkeit (Disjunktheit der Unterklassen)

Provided Management Capability

Angebotene Managementfähigkeiten weisen keine weiteren – außer die von der Oberklasse Managementfähigkeit geerbten – Einschränkungen auf.

Required Management Capability

Neben den von der Oberklasse Managementfähigkeit geerbten Einschränkungen weist die Klasse `Required Management Capability` die zusätzliche Einschränkung auf, dass sie von einer Managementaktivität benötigt wird.

Equivalent classes:
(isRequiredBy some ManagementActivity)
and (isRequiredBy only ManagementActivity)

Quelltext 5 Einschränkungen bei den benötigen Managementfähigkeiten

Management Entity

Managemententitäten sind genau einem einzigen Managementbereich zugeordnet. Hierdurch wird es überhaupt erst möglich, verschiedene Managementdienste bezüglich der Trennung verschiedener funktionaler Kontexte zu untersuchen.

Equivalent classes:
(isOperatedBy some ManagementActivity)
and (isDefinedBy only ManagementEntityStructurePolicy)
and (isOperatedBy only ManagementActivity)
and (isDefinedBy exactly 1 ManagementEntityStructurePolicy)

Super classes:
(isPartOf only ManagementArea)
and (isPartOf exactly 1 ManagementArea)
and (hasIdentifier some string)
and (hasIdentifier only string)

Quelltext 6 Einschränkungen bei der Managemententität

Als weitere hinreichende Einschränkung wird festgelegt, dass eine Managemententität von einer Managementaktivität manipuliert sowie durch eine Managemententitätsstrukturrichtlinie definiert wird.

Management Participant

Managementteilnehmer zeichnen sich dadurch aus, dass sie Managementaktivitäten durchführen. Dies wird durch die Definition der `participatesIn`-Assoziation festgelegt. Weiterhin sind Managementteilnehmer genau einem einzigen Managementbereich zugeordnet und besitzen einen eindeutigen Identifikator.

Equivalent classes:
(participatesIn some ManagementActivity)
and (participatesIn only ManagementActivity)

Super classes:
(isPartOf some ManagementArea)
and (isPartOf only ManagementArea)
and (hasIdentifier some string)
and (hasIdentifier only string)

Quelltext 7 Einschränkungen beim Managementteilnehmer

Management Policy

Die nachfolgenden Einschränkungen legen fest, dass zur Definition von Managemententitäts-Strukturrichtlinien lediglich textbasierte Beschreibungen zum Einsatz kommen dürfen. Somit wird verhindert, dass unabhängig vom später genau zu definierenden Attributtyp Entscheidungen im Voraus festgelegt werden.

Super classes:
(isPartOf only ManagementArea)
and (hasDescription some string)
and (hasDescription only string)

Quelltext 8 Einschränkungen bei der Managementrichtlinie

Management Entity Structure Policy

Die Richtlinie für eine Managemententität legt den Aufbau einer Managemententität fest. Dies ist insofern sinnvollerweise direkt in der Analysephase zu unterstützen, da dadurch direkt während der Erfassung der Anforderungen bestimmte erforderliche Informationen festgelegt werden können wie z. B. welche Information bei der Meldung einer Störung angeben muss. Die Managemententitäts-strukturrichtlinie legt demnach den fachlich-orientierten Aufbau von Managemententitäten fest. Die Einschränkung im Auszug aus dem Quelltext 9 legt fest, dass eine Managemententitätsstrukturrichtlinie exakt eine Managemententität definiert.

Equivalent classes:
(defines only ManagementEntity)
and (defines exactly 1 ManagementEntity)

Quelltext 9 Einschränkungen bei der Managemententitätsstrukturrichtlinie

Assoziationen

Neben der Festlegung der Semantik der einzelnen Modellelemente in der OWL-Ontologie muss auch eine Definition der Assoziationen der einzelnen Elemente untereinander erfolgen, damit die Modellierung konkreter Szenarien konform zu der Semantik des Metamodells vorgenommen werden kann. Hierzu werden in OWL die Assoziationen zwischen den einzelnen Modellelementen durch spezielle Definitionen – den Objektbeziehungen (engl. *Object Properties*) – festgelegt. Eine *Object-Property*-Definition legt fest, in welchem Fall eine Beziehung zwischen zwei ausgewählten Elementen gültig ist. Im Gegensatz zur Definition in z. B. einem UML-basierten Metamodell wird durch eine *Object-Property*-Definition in OWL demnach nicht spezifiziert, welche Assoziationen zwischen Modellelementen definiert werden müssen, sondern vielmehr, was für eine Assoziation vorliegt, wenn eine Beziehung zwischen zwei Elementen in einem konkreten Domänenmodell identifiziert wurde. Im Umkehrschluss gibt die Definition einer *Object-Property*-Definition dem beteiligten Akteuren im betrachteten Entwicklungsprozess aber auch Hinweise, welche Assoziationen zwischen Modellelementen überhaupt vorkommen können, um konforme Domänenmodelle konstruieren zu können. Die Definition der Assoziationen wurde ausschnittsweise in [PR+11, Re11] veröffentlicht.

contains

Die Assoziation `contains` legt die Beziehung der zu einem Managementbereich gehörenden Elemente fest. Die inverse Assoziation hierzu wird ebenfalls festgelegt und als `isPartOf` definiert.

Domains:
ManagementArea

Ranges:
ManagementActivity
or ManagementCapability
or ManagementEntity
or ManagementParticipant
or ManagementPolicy

Inverse properties:
isPartOf

Quelltext 10 Object property contains

defines

Die zur `isDefinedBy` inverse Assoziation legt fest, dass eine Managemententitätsstrukturrichtlinie den syntaktischen Aufbau einer spezifischen Managemententität definiert.

Domains:
ManagementEntityStructurePolicy

Ranges:
ManagementEntity

Inverse properties:
isDefinedBy

Quelltext 11 Object property defines

hasParticipant

Die Assoziation zwischen einem Managementteilnehmer und den von diesem Teilmnehmer durchgeführten Managementaktivitäten wird als `hasParticipant` definiert. Die inverse Relation wird als `participatesIn` festgelegt.

Domains:
ManagementParticipant

Ranges:
ManagementActivity

Inverse properties:

participatesIn

Quelltext 12 Object property hasParticipant

isComposedOf

Um mehrere Managementaktivitäten zu einer zusammengesetzten Managementaktivität komponieren zu können, wird die Assoziation `isComposedOf` definiert. Die inverse Relation hierzu wird ebenfalls definiert und lautet `isUsedBy`.

Domains:
ManagementComposedActivity

Ranges:
ManagementActivity

Inverse properties:
isUsedBy

Quelltext 13 Object property isComposedOf

isDefinedBy

Die Assoziation zwischen einer Entitätsstrukturrichtlinie und der zugehörigen Managemententität wird durch die Assoziation `isDefinedBy` definiert. Die hierzu inverse Assoziation wird ebenfalls definiert und lautet `defines`.

Domains:
ManagementEntity

Ranges:
ManagementEntityStructurePolicy

Inverse properties:
defines

Quelltext 14 Object property isDefinedBy

isOperatedBy

Die Beziehung zwischen einer Managemententität und der auf der Entität operierenden Managementaktivitäten wird definiert durch die Assoziation `isOperatedBy`.

Domains:
ManagementEntity

Ranges:
ManagementActivity

Inverse properties:
operatesOn

Quelltext 15 Object property isOperatedBy

isPartOf

Die Assoziation `isPartOf` legt die Beziehung der zu einem Managementbereich gehörenden Elemente fest. Die inverse Assoziation hierzu wird ebenfalls festgelegt und als `contains` definiert.

Domains:
ManagementActivity
or ManagementCapability
or ManagementEntity
or ManagementParticipant
or ManagementPolicy

Ranges:
ManagementArea

Inverse properties:
contains

Quelltext 16 Object property isPartOf

isRequiredBy

Die von einer spezifischen Managementaktivität erforderliche Managementfähigkeit wird in einer Ontologie durch die Assoziation `isRequiredBy` festgelegt. Die hierzu inverse Relation lautet `requires`.

Domains:
ManagementCapability

Ranges:
ManagementActivity

Inverse properties:
requires

Quelltext 17 Object property isRequiredBy

isUsedBy

Während die Assoziation `isComposedOf` die zu einer zusammengesetzten Managementfähigkeit gehörenden einzelnen Managementaktivitäten verbindet, definiert die Assoziation `isUsedBy` die entgegengesetzte Richtung.

Domains:
ManagementActivity

Ranges:
ManagementComposedActivity

Inverse properties:
isComposedOf

Quelltext 18 Object property isUsedBy

operatesOn

Die Assoziation `operatesOn` legt die Beziehung zwischen einer Managementaktivität und einer eventuell hierzu benötigten Managemententität fest.

Domains:
ManagementActivity

Ranges:
ManagementEntity

Inverse properties:
isOperatedBy

Quelltext 19 Object property operatesOn

participatesIn

Die Assoziation `participatesIn` legt fest, dass ein Managementteilnehmer eine spezifische Managementaktivität durchführt. Die hierzu inverse Assoziation `hasParticipant` wird ebenfalls definiert.

Domains:
ManagementParticipant

Ranges:
ManagementActivity

Inverse properties:
hasParticipant

Quelltext 20 Object property participatesIn

requires

Die zur `isRequiredBy` inverse Assoziation legt fest, dass eine bestimmte Managementaktivität eine Managementfähigkeit benötigt.

Domains:
ManagementActivity

Ranges:
ManagementCapability

Inverse properties:
isRequiredBy

Quelltext 21 Object property requires

Modellierung von Dienstkandidaten

Für den weiteren Verlauf des Entwicklungsprozesses wird im dienstorientierten Entwurf die Modellierungssprache SoaML eingesetzt, die aufgrund der bereitgestellten Modellierungsmöglichkeiten unterschiedliche Modellierungselemente enthält, um bereits in der dienstorientierten Analyse die Identifikation von möglichen Managementdiensten in Form von Managementdienstkandidaten zu unterstützen. Der Vorteil der SoaML liegt demnach in der klar definierten Semantik der einzelnen Modellelemente begründet. Daher wird die durch das konzeptionelle Modell zur Identifikation von Managementdienstkandidaten vorgegebene abstrakte Syntax auf die durch die SoaML zur Verfügung gestellte konkrete Syntax umgesetzt.

Semantischer Bezug zum Domänenmodell im Dienstkandidatenmodell

Zur Modellierung von Dienstkandidaten kann in der SoaML das Modellelement `Capability` genutzt werden [PW+10, PP+10, OMG-SoaML, Am10, De10, Ge11]. Dieses Modellelement entspricht semantisch dem Modellelement `Management Service Candidate` und kann bei entsprechenden Modellüberführungen als solches übernommen werden.

Da in einem SoaML-basierten Dienstkandidatenmodell zunächst kein weiterer semantischer Bezug zu den zugrunde liegenden Domänenmodellen besteht, muss die Information im vom Domänenmodell abgeleiteten SoaML-Dienstkandidatenmodell erweitert werden. Vor allem zur Bewertung der Wiederverwendbarkeit der entworfenen Managementdienste in den SoaML-Dienstmodellen ist es erforderlich, dass die relevanten Konzepte aus den Domänenmodellen enthalten sind. Dies kann prinzipiell durch drei, teilweise direkt von der SoaML unterstützt, verschiedene Ansätze erreicht werden:

(1) Eine Verfeinerung des durch die OMG definierten UML-Profils für die Spezifikation des Metamodells der SoaML erlaubt die Definition und Einführung weiterer Untertypen des Modellelementes `Capability`. Da durch dieses Modellelement jedoch ursprünglich nicht die eigentliche Modellierung des Konzeptes eines Dienstkandidaten verfolgt wird, muss eine Verfeinerung in Form einer erweiterten Profildefinition erfolgen. Diese erweiterte Profildefinition kann zwar in konkreten Entwicklungswerkzeugen genutzt werden (sofern diese die Möglichkeit bieten, zusätzliche auf UML-basierte Profildefinitionen zu nutzen), jedoch steht mit den durch das SoaML-Metamodell bereitgestellten Modellelemente `Category` und `Catalogue` insgesamt ein wesentlich zielführenderer Ansatz zur Verfügung.

(2) Die Information bezüglich der Klassifizierung verschiedener Dienstkandidaten kann in einem konkreten SoaML-Modell durch die Möglichkeit erfasst werden, eine Instanz eines konkreten Modellelementes zu benennen. Um eine Erfassung der Klassifizierung verschiedener Typen von Dienstkandidaten zu ermöglichen, muss entweder eine klar definierte Konvention bei der Benennung von Dienstkandidaten eingehalten werden oder eine automatisierte Transformation von Instanzen des Domänenmodells auf Instanzen eines SoaML-basierten Dienstkandidatenmodell durchgeführt werden. Im Rahmen dieser automatisierten Transformation kann die Benennung der Instanzen des `Capability`-Elementes durch den ausgeführten Transformationsalgorithmus erfolgen.

(3) Die bereits angesprochenen SoaML-Metamodellelemente `Category` und `Catalog` bieten die Möglichkeit, verschiedene Elemente eines SoaML-Modells zu gruppieren, um hierdurch verschiedene Blickpunkte auf ein Modell auf Basis der gleichen definierten Modellelemente umzusetzen. Dadurch können, ähnlich dem Ansatz der aspektorientierten Programmierung, verschiedene Konzepte durch nicht-invasive Änderungen an den zugrunde liegenden Modellen eingefügt werden. Beispielsweise können auf Basis ein und desselben Modells verschiedene geschäftsbezogene und technologiebezogene Aspekte dargestellt und erfasst werden. Insgesamt sieht das Metamodell verschiedene Elemente vor, um unterschiedliche Charakteristika verschiedener Blickpunkte umzusetzen. In Abbildung 22 sind die vom Metamodell vorgegebenen Elemente samt der erweiterten Metaklassen dargestellt.

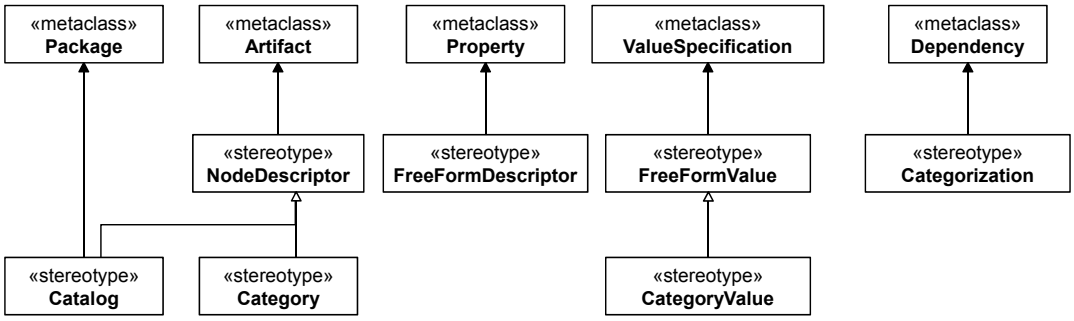

Abbildung 22 SoaML-Metamodellelemente zur Klassifikation und Kategorisierung

Die Metamodellelemente Katalog (`Catalog`) und Kategorie (`Categoy`), um das Konzept zur Gruppierung verschiedener Aspekte umzusetzen, erweitern hierzu die UML-Metamodellelemente Paket (`Package`) und Artefakt (`Artifact`). Konkrete Werte verschiedener Kategorien können durch das Element `CategoryValue` ausgedrückt werden. Die Metamodellelemente `NodeDescriptor`, `FreeFormDescriptor`, und `FreeFormValue` werden aus Kompatibilitätsgründen mit der von der OMG spezifizierten *Reusable Asset Specification* [OMG-RAS] veröffentlichten Standardspezifikation eingeführt und stellen Platzhalter für eine Integration dieser Spezifikation dar. Der Einsatz in einem SoaML-Modell ist zunächst direkt nicht vorgesehen.

Da durch den von der SoaML vorgegeben metamodellbasierten Ansatz zunächst lediglich eine konkrete Syntax sowie die statische Semantik eines Konzeptes zur Modellierung verschiedener Aspekte in einem SoaML-Modell definiert werden, ist zur Überführung der Klassifikationsinformation

ähnlich der zuvor betrachteten Variante zu verfahren. Die Definition von Katalogen und Kategorien kann direkt im Entwicklungswerkzeug erfolgen. Eine Sammlung verschiedener `Category`-Elemente in einem gemeinsamen `Catalog`-Element erleichtert die spätere Auffindbarkeit der jeweiligen Typinstanzen. Weiterhin fordert die Definition der Semantik des Modellelementes `Category` in [OMG-SoaML] dass jedes `Category`-Element mindestens einem `Catalog`-Element zugeordnet ist. Die Zuordnung eines `Category`-Elementes zu einem `Catalog`-Element erfolgt mit der Definition einer *OwnedElement*-Beziehung.

Element des konzeptionellen Dienstkandidatenmodells	**Element von SoaML**
`Management Capability`	`Capability Operation`
`Management Basic Service Candidate`	`Capability, Category, CategoryValue`
`Management Process Service Candidate`	`Capability, Category, CategoryValue`
`Management Entity`	`MessageType, Category, CategoryValue`

Tabelle 5 SoaML-Dienstkandidatenmodellierung

In Tabelle 5 ist eine Gegenüberstellung der Modellelemente aus der Taxonomie der Domäne und der Modellelemente der SoaML, die hierzu eine semantische Entsprechung aufweisen, dargestellt. Es ist ersichtlich, dass die unterschiedlichen Typen verschiedener Managementdienstkandidaten jeweils alle auf (verschiedene) Instanzen des SoaML-Modellelementes `Capability` abgebildet werden können. Somit geht der semantische Bezug der einzelnen Dienstkandidaten bezüglich ihrer Einordnung zunächst verloren.

Eine beispielhafte Modellierung eines Dienstkandidaten auf Basis eines Ausschnitts eines Domänenmodells ist in Abbildung 23 dargestellt. Dargestellt sind drei Dienstkandidaten (die beiden Managementbasisdienstkandidaten `WDBEntryService` und `IncidentRecordService` sowie der Managementprozessdienstkandidat `WDBCheckingService`). Ebenso beispielhaft sind verschiedene Kategorien abgebildet, die eine Klassifizierung der jeweiligen Dienstkandidaten ermöglichen.

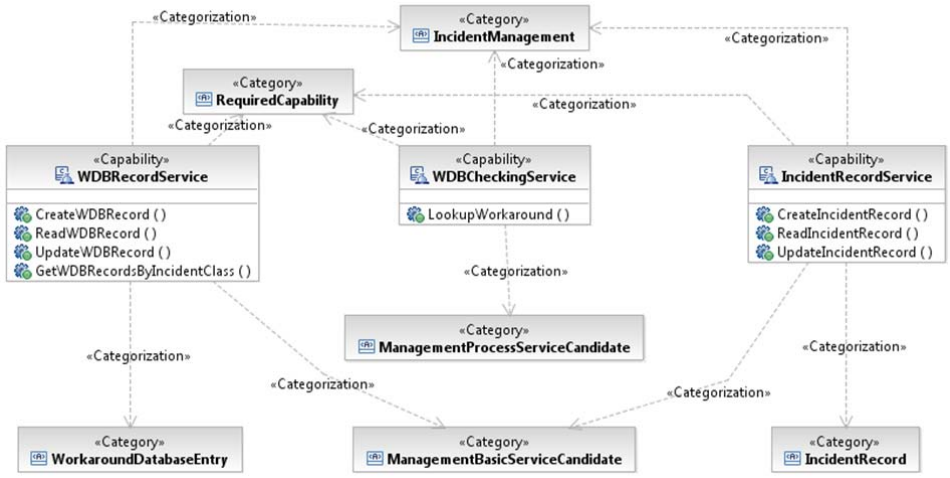

Abbildung 23 Semantische Erweiterung eines Dienstkandidaten im SoaML-Modell

Hieraus ist ersichtlich, dass die Information bezüglich der Klassifizierung eines Dienstkandidatentypen durch die Erweiterung des Dienstkandidatenmodells mit dem `Category` und dem `CategoryValue`-Element umgesetzt werden kann.

Zur Steigerung der Nachvollziehbarkeit wird im Folgenden ein Vorschlag grundlegender zur Verfügung zu stellender Kataloge vorgestellt. Dieser Vorschlag von Katalogen bildet die Grundlage, um im weiteren Verlauf die Bewertung der Wiederverwendbarkeit der vorgestellten Managementdienste durch einen Bezug zu den für einen konkreten Entwurf erstellten Domänenmodellen herstellen zu können.

Kataloge und Kategorien

Aus der analytischen Betrachtung der Elemente des Domänenmetamodells ergeben sich verschiedene Klassifizierungen verschiedener Aspekte, die durch die Definition von `Catalog`-Elementen spezifiziert werden. Der Unterschied zum Modellelement `Category` besteht in der Klasse/Instanz-Beziehung zwischen `Catalog`/`Category`.

Der Katalog ManagementArea

Der Katalog zur Klassifizierung eines Dienstkandidaten- oder Dienstmodellelementes bezüglich des zugehörigen Managementbereiches umfasst die aus einer Vorabanalyse identifizierbaren Aufzählungselemente des entsprechenden Domänenmetamodellelementes. Abbildung 24 zeigt ein Modell des Kataloges mit verschiedenen dreizehn identifizierten Managementbereichen gemäß ISO/IEC20000-1:2005 [ISO05].

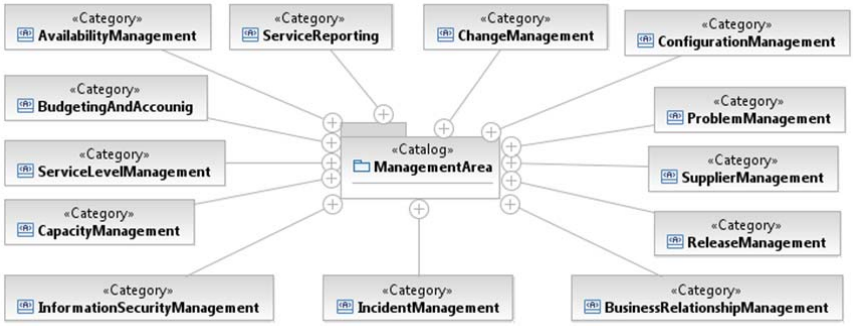

Abbildung 24 Der Katalog ManagementArea

Die Modellierung erfolgt in SoaML durch eine Stereotypisierung der UML-Metaklassen `Package` (für `Catalog`) bzw. `Artifact` (für `Category`) [OMG-UML-Super]. Die Beziehung zwischen `Catalog`- und `Category`-Element ist gemäß der Vorgabe im Standard über eine `ownedElement`-Assoziation dargestellt.

Der Katalog ManagementServiceType

Ein grundlegendes Klassifizierungsmerkmal einer dienstorientierten Architektur zur Unterstützung der Ausführung automatisierbarer Betreiberprozesse ist die Einteilung der Dienste der Architektur in die beiden Typen Managementbasisdienst und Managementprozessdienst. Da die vorgestellte Methode den fachfunktionalen Entwurf zur Überführung fachlicher Anforderungen betrachtet, wird der dritte grundlegend erforderliche Diensttyp – Managementplattinfrastrukturdienst – an dieser Stelle nicht weiter betrachtet.

Abbildung 25 Der Katalog ManagementServiceType

In Abbildung 25 ist der Katalog zur Klassifizierung unter Typen von Managementdiensten dargestellt. Die Kategorisierungselemente leiten sich aus einer Vorabanalyse des entsprechenden Metamodellelementes ab und können daher statisch spezifiziert werden.

Der Katalog ManagementCapabilityType

Zielsetzung der vorliegenden Arbeit ist die Konstruktion fachlich-motivierter Schnittstellen zu bestehenden Managementwerkzeugen. In realen Entwicklungsprojekten wird eine Abbildung von benötigter Funktionalität auf angebotene Funktionalität durchgeführt. Durch das Domänenmetamodell wird hierzu mit dem Element Managementfähigkeit (engl. *Management Capability*) ein dediziertes

Metamodellelement eingeführt, das die Grundlage bildet, diese Abbildung syntaktisch überhaupt beschreiben zu können. Eine Klassifizierung einer identifizierten Managementfähigkeit umschließt daher die beiden Aspekte der angebotenen Managementfähigkeit (engl. *Provided Capability*) und benötigten Managementfähigkeit (engl. *Required Capability*).

Abbildung 26 stellt den Katalog zur Klassifizierung der unterschiedlichen Typen von Managementfähigkeiten dar.

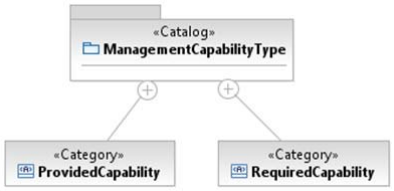

Abbildung 26 Der Katalog ManagementCapabilityType

Ein zur Managementfähigkeit vergleichbares Konzept findet sich in der SoaML mit dem Metamodellelement der SoaML-`Capability` wieder. Während die Semantik im Domänenmetamodell auf die Abstraktion von einzelnen Managementaktivitäten bzw. Werkzeugfunktionalität abzielt und somit mit einem Operationskandidaten eines Dienstkandidaten vergleichbar ist, bezieht sich die Semantik der SoaML-`Capability` auf vollständige Dienstkandidaten.

Die Kataloge ManagementEntity, ManagementPolicy

Wie bereits im einleitenden Abschnitt herausgearbeitet, existieren zwei grundlegend unterschiedliche Typen von Klassifizierungsaspekten. Während die von konkreten Projekten unabhängigen Klassifizierungsaspekte wie zugeordneter Managementbereich, Typ von Managementfähigkeit oder Managementdienst direkt aus dem Domänenmetamodell abgeleitet werden können, kann für weitere Klassifizierungsaspekte erst bei Vorlage von konkreten Domänenmetamodellinstanzen entschieden werden, welche Klassifizierungsmerkmale sich anbieten.

Abbildung 27 Die Kataloge ManagementEntity, ManagementPolicy sind zunächst leer

Hierzu gehören insbesondere die Kataloge für Managemententitäten, Managementrichtlinien und Managementaktivitäten welche zunächst als leere UML-Packages stereotypisiert definiert werden.

4.2.5 Zusammenfassung Metamodell

Das in diesem Abschnitt vorgestellte Metamodell zur Modellierung von identifizierbaren Dienstkandidaten stellt die Grundlage dar, um einen durchgehenden Entwurf von Managementdiensten auf Basis der in der dienstorientierten Analyse erfassten fachfunktionalen Anforderungen zu ermöglichen. Der Ansatz, ein eigenständiges Metamodell der Domäne zu erarbeiten, eröffnet die Möglichkeit, unabhängig von konkreten Vorgehensmodellen zur Etablierung dienstorientierter Architekturen zu sein. Im nachfolgenden Abschnitt werden diese Aspekte vertiefend betrachtet und es wird auf den Unterschied bei der Spezifikation der unterschiedlichen Typen von Managementdiensten – den Managementbasisdiensten und Managementprozessdiensten – eingegangen.

Mit der formalen Spezifikation der unterschiedlichen Teilaspekte eines Metamodells für die Domäne IT-Management in Form einer Ontologie ist die Grundlage geschaffen, um auf Basis dieses Metamodells den Entwurf von Managementdiensten im Rahmen eines dienstorientierten Softwareentwicklungsprozesses zu betrachten.

4.3 Wiederverwendbarkeit von Managementdiensten

Da mit der Ausrichtung am Paradigma der dienstorientierten Architektur die Wiederverwendung von Softwarekomponenten durch einfache Nutzung von fachlich-motivierten Schnittstellen angestrebt wird, nimmt das Prinzip der Wiederverwendung zur Umsetzung von fachlichen Anforderungen innerhalb einer dienstorientierten Architektur einen zentralen Stellenwert ein. Während mit der Integration bestehender Managementwerkzeuge zur automatisierten Ausführung von Betreiberprozessen die Wiederverwendung der bestehenden Softwareartefakte bereits in der Motivation der vorliegenden Arbeit auftaucht, werden in diesem Abschnitt verschiedene Aspekte und Kriterien von Wiederverwendbarkeit untersucht, die sich auf den Entwurf von Managementdiensten beziehen, um den genannten Problemstellungen zu begegnen.

Wiederverwendung von Software bezeichnet den Prozess zur Konstruktion von Softwaresystemen auf Basis bestehender Softwarekomponenten [Kr92]. Davon unterschieden werden muss die Bewertung der Wiederverwendbarkeit, die im Gegensatz zur Bestimmung der Wiederverwendung ein in die Zukunft gerichtetes Maß definiert. Ein hohes Maß an Wiederverwendbarkeit entwickelter Komponenten ist die Grundlage, um durch Wiederverwendung die beispielsweise in Abschnitt 2.2.2 (Seite 16 ff.) genannten Vorteile erzielen zu können. Vereinfacht gesprochen kann ein bewusster Entwurf bezüglich einer hohen Wiederverwendbarkeit eine günstige Auswirkung auf die Wiederverwendung besitzen. Umgekehrt kann gefolgert werden, dass spätere Wiederverwendung erfordert, dass in einem Entwicklungsvorgehen der Aspekt der Wiederverwendbarkeit beachtet werden muss [FG+02]. Wiederverwendbarkeit im Kontext der vorliegenden Arbeit bezieht sich hierbei auf die fachliche Ausrichtung der entworfenen Managementdienste, da diese letztlich genutzt werden, um automatisierte Betreiberprozesse zu unterstützen.

Wiederverwendung bezieht sich auf fertige, implementierte Dienste. Wiederverwendung impliziert demnach, dass Eigenschaften einer fertigen Implementierung betrachtet werden. Da spätere Änderungen an fertigen, verteilten und in Betrieb genommenen Artefakten jedoch in der Regel

kostenintensiv sind [Po97], ist die Untersuchung verschiedener Aspekte von Wiederverwendbarkeit bereits im Entwurf wünschenswert. Der Einsatz einer Ontologie zur Modellierung der relevanten Konzepte einer Domäne unterstützt dieses Prinzip wesentlich [FG+02]. Hierzu sind jedoch verschiedene, teilweise komplexe Sachverhalte zu berücksichtigen [Pr93]. Zur Bewertung der Wiederverwendbarkeit muss eine Bezugsskala angegeben werden, auf die sich das zu bestimmende Maß an Wiederverwendbarkeit bezieht. Aus dieser grundlegenden Überlegung lässt sich bereits ableiten, dass die Bewertung der Wiederverwendbarkeit entworfener Managementdienste an konkreten vorliegenden Modellen erfolgen muss. In diesem Abschnitt werden daher die wesentlichen strukturellen Grundlagen vorgestellt, um die Bewertung der Wiederverwendbarkeit in den folgenden Kapiteln weiter konkretisieren und an praktischen Beispielen demonstrieren zu können. Insbesondere werden in den Tragfähigkeitsnachweisen konkrete Beispiele zur Anwendung der im Folgenden vorgestellten Aspekte vermittelt.

Um die Möglichkeit zu verdeutlichen, durch die Angabe von konkreten Berechnungsvorschriften die vorgestellten Aspekte formal zu erfassen, wird an drei ausgesuchten und grundlegenden Aspekten aufgezeigt, wie eine Definition von Berechnungsvorschriften aufgebaut werden kann. Diese hier vorgestellte Möglichkeit kann einen Ausgangspunkt für weiterführende Arbeiten bieten, den bewussten, auf Wiederverwendbarkeit hin ausgelegten Entwurf durch geeignete formale Ansätze zu unterstützen. Bestehende Arbeiten aus dem Bereich objektorientierter Ansätze oder komponentenbasierter Ansätze (beispielsweise in [WY+03, RD05]) mit der Zielsetzung, Wiederverwendung zu untersuchen, bilden hierbei die Grundlage.

4.3.1 Wiederverwendbarkeit in Managementdienstmodellen

Um einen vorliegenden Entwurf von Managementdiensten bezüglich der verschiedenen Aspekte von Wiederverwendbarkeit zu bewerten, ist eine Fokussierung auf einige zielführende Aspekte sinnvoll, da nicht alle vorgestellten Aspekte relevant zur Analyse- oder Entwurfszeit von Managementdiensten sind oder durch die Angabe von Berechnungsvorschriften gemessen werden können. Daher wird im Folgenden eine Auswahl von relevanten Aspekten vertieft betrachtet, die den beteiligten Analysten oder Entwicklern des betrachteten Entwicklungsprozesses eine direkte Unterstützung anbieten. Eine vergleichbare Betrachtung wird durch bestehende Arbeiten zwar in einigen Fällen vertieft (z. B. in [RD05, CK08, HL+08]), bei diesen Arbeiten fehlt jedoch der direkte Bezug zu einer möglichen Instrumentierung in den einem Entwurf zugrunde liegenden Modellen.

Klassifikation

Die Identifikation von Dienstkandidaten auf der Grundlage der erstellten Domänenmodelle hat letztlich zum Ziel, den späteren Entwurf von Managementdiensten bezüglich der betrachteten Eigenschaften von Betreiberprozessorientierung und Wiederverwendbarkeit zu unterstützen. Durch die Definition von Dienstkandidaten wird bereits in der Analysephase des Entwicklungsvorgehens versucht, eine Gruppierung von zueinander gehörenden logischen Anwendungskomponenten zu definieren, die die weitere Grundlage für einen dienstorientierten Entwurf bilden. Diese Gruppierung wird zunächst rein unter dem Gesichtspunkt fachlicher Erwägungen getroffen. Die Aufteilung der verschiedenen logischen Komponenten ergibt in der Gesamtheit eine erste Übersicht über die gesamte

dienstorientierte Architektur. Um die hierbei in Erscheinung tretenden Abhängigkeiten der einzelnen Elemente dieser Gesamtarchitektur sichtbar zu machen, ist die Einordnung in eine möglichst eindeutige Kategorie sinnvoll [RH06]. Reussner und Hasselbring zufolge sind (Software-) Kategorien...

> *„... eine Richtschnur beim Entwurf von Komponenten und Schnittstellen und dienen als Kontrollinstanz bei späteren Änderungen." [RH06]*

Die Auszeichnung aller verschiedenen Kategorien einer Komponente definiert die Klassifikation dieser Komponente. Die Klassifikation gibt insgesamt Rückschluss auf die verschiedenen Aspekte, die durch diese Komponente – im konkreten Falle einem vorliegenden Managementdienstkandidaten oder einer spezifizierten Managementdienstschnittstelle – abgedeckt werden.

Damit implementierte und betriebene Managementdienste wiederverwendet werden können, ist es grundsätzlich erforderlich, diese in einem konkreten Entwicklungsprojekt auffinden zu können. Damit ein Artefakt aufgefunden werden kann, ist die Unterstützung einer Infrastruktur erforderlich (z. B. ein Dienstverzeichnis, ein Dienstkatalog, ein sucheunterstützendes System usw.). Diesem technischen Aspekt bei der Auswahl steht die Anforderung gegenüber, dass die benötigten Managementdienste anhand der Konzepte aus dem einem Entwurf zugrunde liegenden Domänenmodell identifiziert und verstanden werden können. Dieser fachliche Aspekt ist prinzipiell unabhängig vom technischen Aspekt, ergänzt diesen aber, indem z. B. nicht nur die Suche anhand fest vorgegebener Namen ermöglicht wird, sondern auch an zuvor festgelegten Kategorien. Eine geeignete Klassifikation eines betriebenen und implementierten Dienstes vergrößert demnach die Wiederverwendbarkeit, da durch die Einordnung in eine Klassifikation eine Zuordnung zu den verschiedenen Konzepten im zugrunde liegenden Domänenmodell möglich wird. Diese Klassifikation entspricht den vorgestellten Katalogen und Kategorien. Die Klassifikation zur Analyse- bzw. Entwurfszeit ermöglicht eine durchgehende Erweiterung der entstehenden Entwurfsmodelle im Entwicklungsvorgehen, wodurch die Annotation von implementierten Diensten auf Basis der eingesetzten Entwurfsmodelle möglich wird. Die Betrachtung von Entwurfsmodellen bezüglich deren Klassifikation ist daher Grundlage, um eine Bewertung der Wiederverwendbarkeit der entworfenen Managementdienste durchzuführen.

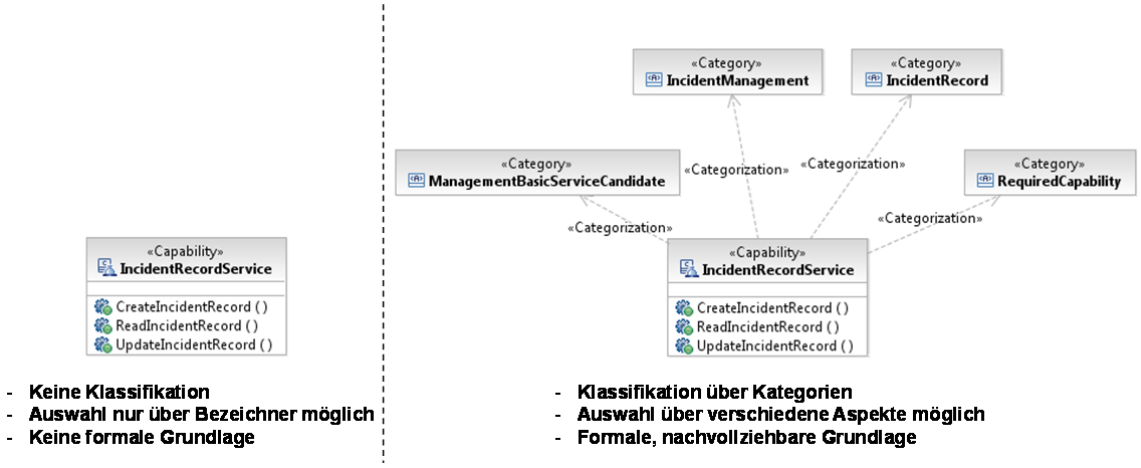

Abbildung 28 Klassifikation eines Dienstkandidaten

Der Aspekt der Klassifikation ist in Abbildung 28 beispielhaft an einem Dienstkandidaten für einen Managementdienst zum Zugriff auf Störungsbelege dargestellt. Wenngleich die Wiederverwendung von Managementdiensten sich auf bestehende, in einem Dienstverzeichnis eingetragene Dienste bezieht, wird hier vereinfachend angenommen, dass der zu diesem Dienstkandidaten ableitbare abstrakte Managementdienst die von seinem Dienstkandidaten zugeordnete Klassifikation übernimmt und als Erweiterung in der Schnittstellenbeschreibung bei der konkreten Implementierung aufweist. Dies kann beispielsweise durch die semantische Annotation einer Webservice-Schnittstelle mit SAWSDL (*Semantic Annotations for WSDL and XML Schema*) [W3C-SAWSDL] erfolgen. Da im linken Teil von Abbildung 28 keine weiteren Informationen beim modellierten Dienstkandidaten angegeben werden, setzt eine Auswahl eines zu diesem Dienstkandidaten gehörenden konkreten implementierten Managementdienstes voraus, dass die gesuchten Bezeichner von Dienstoperationen genau bekannt sind. Eventuelle Erweiterungen in Form einer unscharfen Suche ermöglichen lediglich die Auswahl, wenn die genauen Bezeichnungen nicht bekannt sind. Wie im rechten Teil der Abbildung deutlich wird, unterstützt die Zuordnung verschiedener Kategorien die spätere Auswahl eines Managementdienstes, wenn die genauen Bezeichner nicht bekannt sind. So kann beispielweise die Auswahl eines Dienstkandidaten anhand der Suche nach den verschiedenen zugehörigen Klassifikationen erfolgen.

Die Wiederverwendbarkeit der entworfenen Managementdienste soll vor dem Hintergrund der betrachteten Domäne erfolgen. Da in der dienstorientierten Analyse zum vorgestellten Metamodell konforme Instanzen eingesetzt werden sollen, kann eine Maßzahl ermittelt werden, die den Überdeckungsgrad aller möglichen Klassifikationselemente mit den tatsächlich vorhandenen Klassifikationselementen angibt. Aus dem Quotienten zwischen tatsächlichem und möglichem Überdeckungsgrad kann schließlich das Maß an Wiederverwendbarkeit bezüglich des Aspektes der Auswahl vor dem Hintergrund der fachlichen Klassifikation definiert werden. Hierbei wird ersichtlich, dass die Betrachtung einer vollständigen Klassifikation noch keinen direkten Einfluss auf die Wiederverwendbarkeit der entstehenden Artefakte ausübt, sondern einen indirekten, da lediglich sichergestellt wird, dass alle relevanten Konzepte der einem Dienstkandidaten- oder Dienstschnittstellenmodell zugrunde liegenden Domänenmodelle überführt worden sind. Damit wird die systematische Betrachtung der weiteren Aspekte mit Bezug auf die Domänenmodelle überhaupt erst möglich.

Hierbei ist es zunächst irrelevant, ob ein konzipierter Dienstkandidat oder eine spezifizierte Dienstschnittstelle betrachtet wird. Da jedoch nicht zwangsläufig aus jedem Dienstkandidat genau eine Dienstschnittstelle entworfen wird, ist eine Überarbeitung des bei einer spezifizierten Dienstschnittstelle angegebenen Klassifikationselementes erforderlich. Weiterhin kann eine Klassifikation mehrdeutig sein (Angabe von sowohl Managementprozessdienst als auch Managementbasisdienst) oder mehrfach vorkommen (Angabe mehrerer Managemententitäten). Auch hier wird angenommen, dass diese Aspekte in der Überarbeitung der entstehenden Dienstkandidaten- oder Dienstmodelle beachtet und geeignete Schritte im Entwicklungsvorgehen eingeführt werden. Dies wird in Kapitel 5 (Seite 152 ff.) detailliert betrachtet.

Die Wertigkeit der angegeben Klassifikationsdimensionen $RoCD_{MSC}$ bei einem Dienstkandidaten MSC_x ergibt sich, wie in nachfolgender Formel dargestellt, demnach als Quotient aus der Anzahl

tatsächlichen Klassifikationsdimensionen $AoCD_{MSC}$ und der Anzahl der insgesamt möglichen Klassifikationsdimensionen $ToCD_{MSC}$.

$$RoCD_{MSC}(MSC_x) = \frac{AoCD_{MSC}(MSC_{Xx})}{ToCD_{MSC}}$$

Die Anzahl $ToCD_{MSC}$ aller insgesamt möglichen Klassifikationsdimensionen bei Dienstkandidaten wird als Menge der einzelnen Elemente der in 4.2.4 definierten Kategorien sich wie folgt definiert:

$$ToCD_{MSC} = |\{CD_i|\forall i\}|$$

Somit folgt für die Klassifikationsdimensionen bei einem Dienstkandidaten der vorgestellten Kategorien *ManagementAreaType*, *ManagementCapabilityType*, *ManagementServiceType* und *ManagementEntity*:

$$ToCD_{MSC} = |\{MCT, MST, MAT, ME\}| = 4$$

Die Bestimmung der tatsächlich vorhandenen Klassifikastionsdimensionen an einem Dienstkandidaten gestaltet sich etwas aufwändiger, da erst bei der Vorlage eines konkreten Dienstkaniddtane entschieden werden kann, welche Klassifikationsdimensionen angegeben sind. Die Anzahl $AoCD_{MSC}(MSC_x)$ der tatsächlich vorhandenen Klassifikationsdimensionen ergibt sich wie folgt als Summe zur Überprüfung der Existenz einer Klassifikationsdimension im vorliegenden Modell:

$$AoCD_{MSC}(MSC_x) = \sum_{\forall i} EX(CD_i, MSC_x)$$

Die Auswahlfunktion $EX(CD_i, MSC_x)$ liefert 1 im Falle der Existenz der Klassifikationsdimension CD_i am Dienstkandidaten MSC_x zurück, andernfalls 0:

$$EX(CD_i, MSC_x) = \begin{cases} 1, falls\ \exists c \in CD_i\ auch\ an\ MSC_x \\ 0, sonst \end{cases}$$

Die Auswahlfunktion wird genutzt, um die Existenz eines der definierten Kataloge zu überprüfen.

Die Wertigkeit der angegebenen Klassifikationsdimensionen bei einer Managementdienstschnittstelle ergibt sich analog zur Wertigkeit der angegebenen Klassifikationsdimensionen der Managementdienstkandidaten als Quotient aus allen möglichen und tatsächlich vorhandenen Klassifikationsdimensionen:

$$RoCD_{MSI}(MSI_x) = \frac{AoCD_{MSI}(MSI_x)}{ToCD_{MSI}}$$

Die Anzahl $ToCD_{SI}$ aller möglichen Klassifikationselemente bei einer Dienstschnittstelle ist, im Gegensatz zur Definition aller möglichen Klassifikationselemente der Dienstkandidaten, wie folgt reduziert definiert, da in der Definition der Schnittstelle der bezug zum Typ der Fähigkeit fehlt:

$$ToCD_{MSI} = |\{MST, MAT, ME\}|$$

Die Anzahl $AoCD_{SI}(MSI_x)$ ist als Funktion äquivalent zu $AoCD_{SC}(MSC_x)$ definiert

$$AoCD_{MSI}(MSI_x) = \sum_{\forall i} EX(CD_i, MSI_x)$$

Die Bedeutung der einzelnen Bezeichner ist in nachfolgender Tabelle zusammengefasst dargestellt.

Bezeichner	Bedeutung
$RoCD_{MSC}(MSC_x)$	*Ratio of Classification Dimensions of a Service Candidate* Bezeichnet den Quotienten aus tatsächlicher und möglicher Anzahl von Klassifikationsdimensionen bei einem Managementdienstkandidaten MSC_x
$RoCD_{MSI}(MSI_x)$	*Ratio of Classification Dimensions of a Service Interface* Bezeichnet den Quotienten aus tatsächlicher und möglicher Anzahl von Klassifikationsdimensionen bei einer Managementdienstschnittstelle MSI_x
$AoCD_{MSC}(MSC_x)$	*Amount of Classification Dimensions of a Service Candidate* Bezeichnet die Anzahl angegebener Klassifikationsdimensionen bei einem Managementdienstkandidaten MSC_x
$AoCD_{MSI}(MSI_x)$	*Amount of Classification Dimensions of a Service Interface* Bezeichnet die Anzahl angegebener Klassifikationsdimensionen bei einer Managementdienstschnittstelle MSI_x
$ToCD_{MSC}$	*Total of Classification Dimensions of Service Candidates* Bezeichnet die Anzahl aller möglicher Klassifikationsdimensionen bei Managementdienstkandidaten
$ToCD_{MSI}$	*Total of Classification Dimensions of Service Interfaces* Bezeichnet die Anzahl aller möglicher Klassifikationsdimensionen bei einer Managementdienstschnittstelle
MSC	*Management Service Candidate* Bezeichnet den betrachteten Managementdienstkandidaten

MSI	*Management Service Interface* Bezeichnet die betrachtete Managementdienstschnittstelle
CD_i	*Classification Dimension* Bezeichner für eine vorhandene Klassifikationsdimension aus der Menge aller möglichen Klassifikationsdimensionen
MST	*Management Service Type* Bezeichnet den Typ von Managementdienst
MCT	*Management Capability Type* Bezeichnet den Typ von umgesetzter Managementfähigkeit
MAT	*Management Area Type* Bezeichnet den Managementbereich
ME	*Management Entity* Bezeichnet eine Managemententität

Tabelle 6 Bezeichner und deren Bedeutung bei der Bewertung der Klassifikation

Durch die Struktur der Definitionen mit dem Ansatz, nicht direkt jedes Element der Klassifikation zu zählen sondern in Form von Dimensionen in die Bestimmung einfliessen zu lassen, können sich nur Werte zwischen 0 und 1 ergeben. Ein Wert von 0 deutet hierbei auf einen Dienstkandidaten (oder eine Dienstschnittstelle) hin, der keinerlei Klassifikation aufweisst. Ein Wert von 1 steht für eine optimale Klassifikation vor dem Hintergrund der festgelegten Kataloge.

Mit der Betrachtung des Aspektes der Klassifikation als grundlegende Voraussetzung, die Auffindbarkeit von implementierten und betriebenen Diensten zu ermöglichen und den Elementen des betrachteten Entwurfes einen klaren Bezug zu den jeweiligen betrachteten Kategorien zu geben, ist die Basis geschaffen, einen vorliegenden Entwurf von Managementdiensten bezüglich einer wichtigen Eigenschaft von Wiederverwendbarkeit zu untersuchen. Hierbei ist es zunächst unerheblich, ob ein identifizierter Dienstkandidat oder aber ein bereits spezifizierter Managementdienst betrachtet wird.

Vollständigkeit

Um zukünftige Erweiterungen vorwegzunehmen und dadurch die Wahrscheinlichkeit auf eine direkte Wiederverwendbarkeit unmittelbar zu vergrößern, wird ein vorliegender Entwurf bezüglich verschiedener Muster auf Vollständigkeit überprüft. Die zu untersuchenden Muster sind fachlich-

motiviert, damit eine Umsetzung zukünftiger fachlicher Anforderungen vereinfacht wird. Der Aspekt der Vollständigkeit kann demnach vor allem bei vorliegenden Entwürfen von Managementbasisdiensten (bzw. Managementbasisdienstkandidaten) eingeordnet werden. Vor allem die Vervollständigung von datenzentrierten Managementdiensten bezüglich des *Create*-, *Read*-, *Update*-Musters (CRU-Muster) steht im Fokus dieses Aspektes.

Die explizite Beachtung dieses Aspektes ist insofern von Bedeutung, da deswegen verhindert werden kann, dass verschiedene Dienstoperationen, die logisch zueinander gehören, auf mehrere verschiedene weiter zu entwerfende Dienste verteilt werden. Somit ist letztlich die Kohäsion des jeweiligen Dienstes verletzt. Dies kann bei zukünftigen Erweiterungen zur Folge haben, dass Änderungen an mehreren, bereits betriebenen Diensten erfolgen, diese aufgrund der Verletzung des Kohäsionsprinzips jedoch unkoordiniert voneinander durchgeführt werden. Im schlimmsten Fall kann dies zu mehreren gleichartigen Funktionalitäten in mehreren Diensten führen, wodurch der Aspekt der Disjunktheit verletzt wird.

Die Vollständigkeit eines datenzentrierten Dienstes kann einfach überprüft werden, indem ermittelt wird, ob bereits in der Analyse oder im Entwurf alle benötigten Operationen (*Create Operation (CO)*, *Read Operation (RO)*, *Update Operation (UO)*) beim Dienstkandidaten bzw. bei der definierten Dienstschnittstelle vorhanden sind. Die Wertigkeit ergibt sich als Quotient der bei einem Dienstkandidaten (MSC_x) bzw. bei einer Dienstschnittstelle (MSI_x) definierten Dienstoperationen:

$$RoC_{MSC}(MSC_x) = \frac{EX(CO, MSC_x) + EX(RO, MSC_x) + EX(UO, MSC_x)}{3}$$

sowie für Managementdienstschnittstellen:

$$RoC_{MSI}(MSI_x) = \frac{EX(CO, MSI_x) + EX(RO, MSI_x) + EX(UP, MSI_x)}{3}$$

Die Auswahlfunktion $EX(x)$ wird hier genutzt, um die Existenz der erforderlichen Operationen aus dem CRU-Muster zu überprüfen.

Bezeichner	**Bedeutung**
$RoC_{MSC}(MSC_x)$	*Ratio of Completeness of a Service Candidate* Bezeichnet die Wertigkeit zur Vollständigkeit eines datenzentrierten Managementdienstkandidaten
$RoC_{MSI}(MSI_x)$	*Ratio of Completeness of a Service Interface* Bezeichnet die Wertigkeit zur Vollständigkeit einer datenzentrierten Managementdienstschnittstelle

MSC_x	*Management Service Candidate* Bezeichnet den untersuchten Managementdienstkandidaten
MSI_x	*Management Service Interface* Bezeichnet die untersuchte Managementdienstschnittstelle

Tabelle 7 Bezeichner und deren Bedeutung bei der Bewertung der Vollständigkeit

Aufgrund der Struktur der Berechnungsvorschriften für $RoC_{MSC}(MSC_x)$ und $RoC_{MSI}(MSI_x)$ können die Werte $0, \frac{1}{3}, \frac{2}{3}$ und 1 berechnet werden. Für alle Werte kleiner 1 folgt, dass der betrachtete Dienstkandidat bzw. die betrachtete Dienstschnittstelle nicht vollständig ist und somit noch nicht optimal bezüglich des Aspektes der Vollständigkeit.

Disjunktheit

Ein wesentliches Ziel beim Entwurf von verteilten Softwaresystemen besteht darin, dass die einzelnen Komponenten dieses Systems eine klare Funktion erbringen. Gleichartige Funktionalität, die einem ähnlichen Kontext entspringt, wird in logisch zusammengehörigen Komponenten gebündelt. Um Redundanzen bezüglich der Funktionalität zu vermeiden, ist es daher erforderlich, dass verschiedene Komponenten zueinander funktional-disjunkt sind. Dies vergrößert letztlich die Wahrscheinlichkeit, dass die entworfenen Komponenten direkt in einem ähnlichen Kontext wiederverwendet werden können. In einem Softwaresystem, das auf Wiederverwendbarkeit hin ausgelegt ist und durch den Einsatz bestehender Komponenten und deren angebotener Funktionalität neue Anforderungen umsetzt, beugt Redundanz zwischen den Komponenten des Systems vor. Gilt dies im Allgemeinen für die Konstruktion von Softwaresystemen, gilt dies auch im Speziellen für den Entwurf dienstorientierter Systeme. Funktional-disjunkte Managementdienste sind demnach eine Voraussetzung, um die Wiederverwendung von Managementdiensten zu unterstützen.

Betrachtet man die Entwurfsphase eines dienstorientierten Entwicklungsprozesses um die Wiederverwendbarkeit einer implementierbaren Managementdienstkomponente bezüglich der Disjunktheit abschätzen zu können, kann in den Dienstkandidaten und Dienstschnittstellenmodellen festgestellt werden, inwiefern sich identifizierte Dienstoperationen überdecken. Hierzu ist ein Bezug in den einzelnen Modellen zu den jeweiligen Domänenmodellen erforderlich, was letztlich durch die Angabe und Nutzung der vorgestellten Kataloge zur Klassifikation ermöglicht wird. Eine Überdeckung angebotener Funktionalität kann sich durch die Überprüfung der jeweiligen Operationsnamen oder bei einem klaren semantischen Bezug der einzelnen Operationen auch unabhängig von Namensgleichheit ermittelt lassen.

Überdeckung hinsichtlich des funktionalen Kontextes

Ein Maß zur Bestimmung der Überdeckung verschiedener Managementdienste hinsichtlich deren unterstützten funktionalen Managementbereiches kann über eine Betrachtung der semantisch

erweiterten Dienstkandidaten- oder Dienstschnittstellenmodelle erfolgen. Da durch die semantische Erweiterung von Dienstkandidaten respektive Dienstschnittstellen eine Klassifikation eingeführt wird, kann die Existenz einer deckungsgleichen Klassifikation verschiedener Kandidaten oder Schnittstellen auf eine Überdeckung hinsichtlich des betrachteten Kontextes hindeuten. Dies kann in den vorliegenden Modellen einfach über die Betrachtung der Deckungsgleichheit der an einem vorliegenden Artefakt angehängten Klassifikationsmerkmale ermittelt werden. Die Bestimmtung der Disjunktheit des funktionalen Kontextes beschreibt demnach ein Maß das eine qualitative Aussage hinsichtlich der übrigen Dienstkandidaten eines vorliegenden Entwurfs trifft.

Die Wertigkeit einer Überdeckung des funktionalen Kontextes für Managementdienstkandidaten $AoSFC_{MSC}(MSC_x)$ ergibt sich daher als Quotient aus der Überdeckung der Klassifikationsdimensionen eines Dienstkandidaten und der Gesamtmenge aller Klassifikationsdimensionen der übrigen Dienstkandidaten.

$$AoSFC_{MSC}(MSC_x) = \sum_{i=0}^{i<ToCD_{MSC}} \sum_{j=0}^{j<Ao-1} \sum_{k=0}^{k<AoCD_{MSC}(MSC_j)} COV(CE_i, CE_k)$$

Somit ergibt sich der Quotient aus der Anzahl aller Überdeckungen und der Anzahl aller vorhandener Klassifikationsdimensionen sowie der Normung auf die Anzahl vorhandener Dienstkandidaten:

$$RoSFC_{MSC}(MSC_x) = \frac{1}{Ao_{MSC}} * \frac{AoSFC_{MSC}(MSC_x)}{\sum_{i=0}^{i<Ao_{MSC}} AoCD_{MSC}(MSC_i)}$$

Die Wertigkeit für Managementdienstschnittstellen ergibt sich analog ebenso als Quotient aus der Überdeckung der Klassifikationsdimensionen einer Dienstschnittstelle und der Gesamtmenge aller Klassifikationsdimensionen der übrigen Dienstschnittstellen.

Zunächst folgt die Berechnungsvorschrift zur Ermittlung der absoluten Anzahl an Überdeckungen des funktionalen Kontextes:

$$AoSFC_{MSI}(MSI_x) = \sum_{i=0}^{i<ToCD_{MSI}} \sum_{j=0}^{j<Ao_{MSI}-1} \sum_{k=0}^{k<AoCD_{MSI}(MSI_j)} COV(CE_i, CE_k)$$

und analog hierzu der ebenfalls genormte Quotient:

$$RoSFC_{MSI}(MSI_x) = \frac{1}{Ao_{MSI}} * \frac{AoSFC_{MSI}(MSI_x)}{\sum_{i=0}^{i<Ao_{MSI}} AoCD_{MSI}(MSC_i)}$$

wobei die Funktion $COV(x, y)$ feststellt, ob eine Überdeckung der Klassifikationselemente x, y des jeweiligen Dienstkandidaten oder Dienstschnittstelle existiert:

$$COV(x, y) = \begin{cases} 1, falls\ x\ eine\ \text{Überdeckung}\ mit\ y\ besitzt \\ 0, sonst \end{cases}$$

Die Hilfsfunktionen Ao_{MSC} bzw. Ao_{MSI} liefern die Anzahl der im vorliegenden Entwurf vorhandenen Managementdienstkandidaten bzw. Managementdienstschnittstellen.

In der nachfolgenden Tabelle sind die einzelnen Bezeichner sowie deren Bedeutung zusammengestellt.

Bezeichner	Bedeutung
$AoSFC_{MSC}$	*Amount of Superposition of the Function Context of a Service Candidate* Absoluter Betrag der Überdeckung des funktionalen Kontextes eines betrachteten Dienstkandidaten
$AoSFC_{MSI}$	*Amount of Superposition of the Function Context of a Service Interface* Absoluter Betrag der Überdeckung des funktionalen Kontextes einer betrachteten Dienstschnittstelle
$RoSFC_{MSC}$	*Ratio of Superposition of the Function Context of a Service Candidate* Wertigkeit der Überdeckung des funktionalen Kontextes eines betrachteten Dienstkandidaten
$RoSFC_{MSI}$	*Ratio of Superposition of the Function Context of a Service Interface* Wertigkeit der Überdeckung des funktionalen Kontextes einer Dienstschnittstelle
MSC_x	*Management Service Candidate* Bezeichnet den untersuchten Managementdienstkandidaten
MSI_x	*Management Service Interface* Bezeichnet die untersuchte Managementdienstschnittstelle
CD_i	*Classification Dimension* Bezeichnet eine Klassifikationsdimension
Ao_{MSC}	*Amount of Management Service Candidates* Funktion zur Bestimmung aller in einem konreten Dienstentwurf vorliegenden Managementdienstkandidaten

Ao_{MSI}	*Amount of Management Service Interfaces* Funktion zur Bestimmung aller in einem konreten Dienstentwurf vorliegenden Managementdienstschnittstellen

Tabelle 8 Bezeichner und deren Bedeutung bei der Bewertung der kontextbezogenen Disjunktheit

Die Berechnungsvorschriften können aufgrund ihrer Struktur Werte zwischen 0 und der maximal möglichen Anzahl an Klassifikationselementen annehmen, wobei 0 bedeutet, dass keine Überdeckung des jeweils betrachteten Dienstkandidaten oder der jeweiligen betrachteten Dienstschnittstelle zu den weiteren im konkreten Szenario betrachteten Kandidaten oder Schnittstellen vorherrscht und der maximale Wert eine vollständige Überdeckung des funktionalen Kontextes impliziert.

Die Ergebnisse in der Anwendung dieser Berechnungsvorschrift für den Teilaspekt der Disjunktheit verdeutlichen die Problematik, die durch eine strikte Verfolgung von Wiederverwendbarkeit in Erscheinung treten. Zwar ist eine Aufteilung in abgeschlossene Einheiten und hiermit verbunden eine oftmals möglichst feingranulare Verfeinerung von Managementdiensten möglich, jedoch führt dies dann zu einem Entwurf, der in der Gesamtheit aus vielen Diensten besteht, die in einem zusammengehörenden funktionalen Kontext eingeordnet sind.

Die Betrachtung der Disjunktheit hinsichtlich des funktionalen Kontextes kann demnach auch dahingehend interpretiert werden, dass eine möglichst hohe Überdeckung verschiedener Managementdienste hinsichtlich deren funktionalen Kontextes vorliegen muss, um als kohärente Einheit aufgefasst zu werden. Hierin wird ersichtlich, dass ein absolut optimaler Entwurf wiederverwendbarer Managementdienste nicht zu erreichen ist, vielmehr bedarf es der Vereinbarung auf verfolgte Ziele bereits zu Beginn der Entwicklungsvorgehens, um letztlich die Beachtung verschiedener Handlungsalternativen einzugrenzen.

Überdeckung von Dienstoperationen

Ein weiteres Maß zur Bestimmung der Überdeckung verschiedener Managementdienste kann durch die Analyse der jeweiligen Dienstoperationen ermittelt werden.

Die Überdeckung von Dienstoperationen über verschiedene Dienstkandidaten oder Dienstschnittstellen hinweg verläuft analog zur Ermittlung der Überdeckung des funktionalen Kontextes eines Dienstkandidaten oder einer Dienstschnittstelle Die Wertigkeit bestimmt sich demnach durch die Anzahl von Dienstoperationen, die in weiteren Dienstkandidaten oder Dienstschnittstellen vorkommen. Hierbei interessiert jedoch lediglich die absolute Anzahl an nicht-disjunkten Dienstoperationen, da dies zu vermeiden ist.

$$AoSSO_{MSC}(MSC_x) = \sum_{i=0}^{i<AoOP(MSC_x)} \sum_{j=0}^{j<Ao_{MSC}-1} \sum_{k=0}^{k<AoOP(MSC_j)} COV(MSCOp_i, MSCOp_k)$$

Analog zur Ermittlung der nicht-disjunkten Dienstoperationen eines Dienstkandidaten wird die Anzahl nicht-disjunkter Dienstoperationen einer Dienstschnittstelle wie folgt bestimmt:

$$AoSSO_{MSI}(MSI_x) = \sum_{i=0}^{i<AoOP(MSC_x)} \sum_{j=0}^{j<Ao_{MSI}-1} \sum_{k=0}^{k<AoOP(MSI_j)} COV(MSIOp_i, MSIOp_k)$$

Die Funktionen zur Ermittlung der Überdeckung der Operationen sind wie oben angegeben definiert.

In nachfolgender Tabelle 9 sind die einzelnen Elemente sowie deren Bedeutung der vorgestellten Berechnungsvorschrift zusammen gestellt.

Bezeichner	Bedeutung
$AoSSO_{MSC}$	*Amount of Superposition of Service Operations of a Service Candidate* Anzahl an Dienstoperationen, die in weiteren Schnittstellendefinitionen erscheinen
$AoSSO_{MSI}$	*Amount of Superposition of Service Operations of a Service Interface* Anzahl an Dienstoperationen, die in weiteren Schnittstellendefinitionen erscheinen
$MSCOp$	*Management Service Candidate Operation* Bezeichnet die untersuchte Operation eines Managementdienstkandidaten
$MSIOp$	*Management Service Interface Operation* Bezeichnet die untersuchte Operation einer Managementdienstschnittstelle
$AoOP(MSC_x)$	*Amount of Operations of a Service Candidate* Liefert die Anzahl an Operationskanidaten an einem Dienstkandidat
$AoOP(MSI_x)$	*Amount of Operations of a Service Candidate* Liefert die Anzahl an Operationskanidaten an einem Dienstkandidat

Tabelle 9 Bezeichner und deren Bedeutung bei der Bewertung disjunkter Dienstoperationen

Durch die Struktur der vorgestellten Berechnungsvorschrift sind Werte zwischen 0 und der Anzahl an Dienstoperationen, multipliziert mit der Anzahl aller vorliegenden Dienstschnittstellen, möglich. Idealerweise führt die Anwendung für jede einzelne Dienstschnittstelle zum Wert 0 (keine Überdeckung). Wird ein Wert größer als 0 ermittelt, deutet dies auf eine Dienstoperation hin, die bei der Spezifikation der Schnittstelle einem weiteren Dienstkandidaten hinzugefügt wurde. In diesem Fall ist eine Änderung entweder der ursprünglichen oder der jeweils betrachteten Dienstschnittstelle sinnvoll. Die genauen Entwurfsentscheidungen werden durch die beteiligten Akteure (Analyst oder

Architekt) getroffen und könnenunter Umständen zu einer weiteren Überarbeitung der erstellten Artefakte führen.

4.3.2 Zusammenfassung Wiederverwendbarkeit

Im Allgemeinen nimmt die Wiederverwendung in dienstorientierten Architekturen einen zentralen Stellenwert ein. Damit Dienste wiederverwendet werden können, ist jedoch das Entwicklungsvorgehen darauf auszurichten [FG+02]. Um automatisierbare Betreiberprozesse zu unterstützen und die Vorteile des dienstorientierten Ansatzes wie beispielsweise schnelle Anpassung der IT-Unterstützung bei sich ändernden Prozessabläufen zu nutzen, ist der Entwurf wiederverwendbarer Dienste für die Domäne IT-Management im Besonderen relevant. Dies gilt gerade auch im Hinblick auf die Integration bestehender Managementwerkzeuge, da durch diese Zielsetzung ein schrittweiser Übergang der oftmals gewachsenen Anwendungslandschaften der bestehenden Managementwerkzeuge hin zu einer flexiblen Unterstützung für anpassbare Betreiberprozesse ermöglicht wird.

Um frühzeitig im Entwicklungsvorgehen eine gesicherte Aussage über diejenigen Bedingungen treffen zu können, die eine spätere Wiederverwendung günstig beeinflussen, wurden geeignete Abstraktionen betrachtet, die eine Untersuchung der im Entwicklungsvorgehen erstellten Artefakte ermöglichen. Hierzu werden drei grundlegende Teilaspekte von wiederverwendbaren Managementdiensten detailliert untersucht, die sich durch eine Fokussierung der in der bestehenden Literatur diskutierten Aspekte ergeben. Die einzelnen Aspekte können durch die Angabe von Berechnungsvorschriften jeweils im Detail an den in der dienstorientierten Analyse oder im dienstorientierten Entwurf erstellen Artefakten festgemacht werden. Insgesamt wird in diesem Abschnitt aufgezeigt, inwiefern der Einsatz einer Ontologie zur Definition eines Metamodells der Domäne genutzt werden kann, um die Aspekte der Klassifikation, Vollständigkeit und Disjunktheit in SoaML-basierten Dienstmodellen darzustellen und durch die Angabe von Berechnungsvorschriften bestimmen zu können. Die vorgestellten Berechungsvorschriften wurden bereits auszugsweise in [PL+11, Le11] publiziert.

4.4 Zusammenfassung und erzielter Beitrag

Da die Konstruktion heutiger Softwaresysteme nicht zuletzt aufgrund der vielfältigen Anforderungen seitens der Nutzer durch eine immer größer werden Heterogenität von möglichen einsetzbaren Technologien massiv erschwert wird, müssen geeignete Maßnahmen beschrieben und zur Verfügung gestellt werden, um die Fertigstellung dieser Softwareprodukte zielführend unterstützen zu können. Dies gilt im Besonderen für die Konstruktion von Managementsystemen zur automatisierten Ausführung von Betreiberprozessen. Der Einsatz dienstorientierter Architekturen zur Strukturierung eines Managementsystems von fachlich-motivierten Managementanforderungen aus kann die Integration bestehender Managementwerkzeuge erleichtern. Hierzu muss ein durchgeführter Softwareentwicklungsprozess ganzheitlich von den drei relevanten Perspektiven Managementprozess, Managementwerkzeug und dienstorientierte Architektur betrachtet werden.

Das vorliegende Kapitel stellt daher einen wesentlichen Beitrag für die Konstruktion von Managementdiensten dar. Zunächst werden die wesentliche Konzepte der Domäne IT-Management

eingeführt und abstrahiert, worauf aufbauend eine Taxonomie der Domäne vorgestellt wird. Diese Taxonomie bildet die Grundlage, um ein umfassendes Metamodell der Domäne zu spezifizieren und eine OWL-Ontologie zur praktischen Anwendbarkeit des Metamodells in konkreten Entwicklungsprojekten zu definieren. Auf Basis der im Metamodell definierten Elemente wird die Ausgestaltung der Analyse- und Entwurfsphase eines dienstorientierten Entwicklungsprozesses sowie die Ausprägung der einzelnen Phasenschritte im nachfolgenden Kapitel detailliert beschrieben.

In Abbildung 29 sind die einzelnen Modelle im Rahmen des betrachteten Entwicklungsprozesses zusammengetragen.

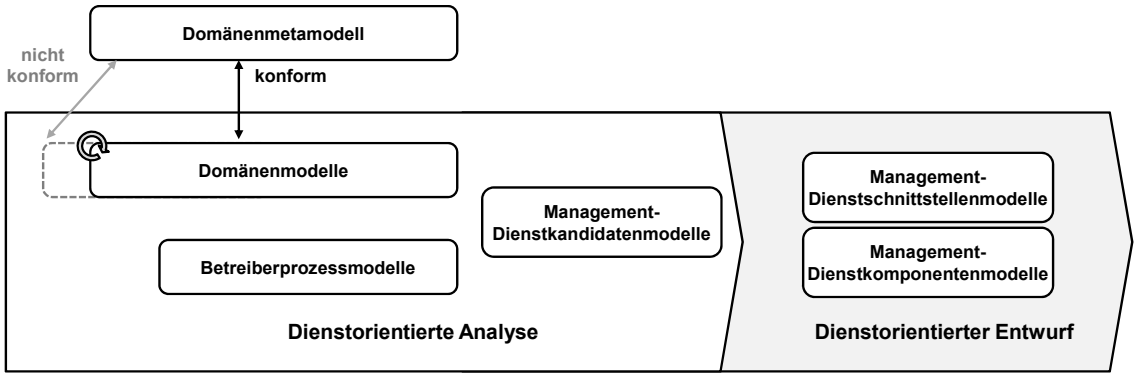

Abbildung 29 Modelle im betrachteten Entwicklungsvorgehen

Die Bewertung der Wiederverwendbarkeit der durch das Metamodell und dem vorgestellten Vorgehensmodell entworfenen Managementdienste wird explizit betrachtet. Eine kritische Evaluation einer dienstorientierten Entwurfsmethode bezüglich dieser zentralen Eigenschaft einer dienstorientierten Architektur ist wesentliche Voraussetzung, um die automatisierte Ausführung von Betreiberprozessen auf Basis bestehender Managementwerkzeuge langfristig auch effizient zu ermöglichen.

Insgesamt kann festgestellt werden, dass die in Abschnitt 3.1 formulierten Anforderungen zur Adressierung der in Abschnitt 1.3 beschriebenen Problemstellungen erfüllt werden. Der Ansatz für den fachlichen Entwurf von Managementdiensten ist abstrahiert von konkreten Entwicklungsprojekten und damit insgesamt gut nachvollziehbar. Durch den Einsatz standardisierter Modellierungssprachen wird der gesamte Ansatz insgesamt in der Praxis anwendbar (Anforderung 1.1). Die entworfenen Managementdienste sind unabhängig von konkreten Managementwerkzeugen (Anforderung 1.2). Vielmehr kann durch die strukturelle Modellierung der betrachteten Fachdomäne und durch die Möglichkeit, bereits bestehende Managementwerkzeuge im integrierten Metamodell darzustellen und mit den fachlich-motivierten Anforderungen auf eine implementierungsunabhängige Art und Weise zu erfassen die Möglichkeit eröffnet werden, die Integration bestehender Werkzeuge in den betrachteten Ansatz einzuführen. Kern dieses Abschnittes ist die Spezifikation eines integrierten Metamodells der Domäne. Das vorgestellte Modell wird aus den wesentlichen Konzepten der Domäne abgeleitet und unterstützt den Entwurf daher zielgerichtet (Anforderung 1.3). Der Nachweis der Wiederverwendbarkeit der entworfenen Dienste erfolgt explizit und durch Unterstützung geeigneter Modellbildungen. (Anforderung 1.4).

5 Domänengetriebener Entwurf wiederverwendbarer Managementdienste

Die Realisierung der Elemente einer dienstorientierten Architektur im IT-Management – den Managementdiensten – kann prinzipiell auf zwei unterschiedliche Arten erfolgen. Existiert keine Implementierung (in Form bestehender Managementwerkzeuge oder bestehender Managementdienste), die die durch den Dienstentwurf identifizierten und benötigten Managementfunktionen umsetzen, kann eine vollständige vorwärts-orientierte (engl. *Top-Down*) Implementierung der spezifizierten Dienstentwürfe durchgeführt werden. Hierbei besteht ein hohes Maß an Handlungsfreiheiten, da aus der Menge von zur Verfügung stehenden Technologien und Ansätzen diejenigen ausgewählt werden können, die die identifizierten Anforderungen auf eine optimale Art und Weise umsetzen. Häufiger jedoch wird gefordert, die Logik und Daten in bestehenden Managementwerkzeugen zu nutzen und diese als Implementierung von spezifizierten Dienstschnittstellen einzusetzen. Wesentlicher Baustein auf dem Weg hin zu einer automatisierten Ausführung von Betreiberprozessen ist daher die Integration von bestehenden Managementwerkzeugen. Hierbei müssen die vom Referenzentwurf abgeleiteten Schnittstellen auf die von den vorhandenen Werkzeugen angebotene Funktionalität abgebildet und umgesetzt werden. Da bestehende Werkzeuge in der Regel nicht auf Basis eines einheitlichen und standardisierten Verständnisses über die zugrunde liegende Domäne entworfen wurden, besteht die Schwierigkeit bei der Integration dieser Werkzeuge in der Konstruktion von Schnittstellen, die entlang an den zu unterstützenden Betreiberprozessen ausgerichtet sind und ein möglichst hohes Maß an Wiederverwendbarkeit aufweisen.

Mit dem im vorigen Kapitel vorgestellten Metamodell und der Betrachtung der verschiedenen Aspekte der Wiederverwendbarkeit an Dienstkandidaten und Dienstentwurfsmodellen ist die Grundlage gelegt worden, um diese Beiträge im Kontext eines strukturierten Entwicklungsvorgehens einsetzen zu können. Dies ist Gegenstand des dieses Kapitels.

5.1 Dienstorientierte Integration

Für die Integration bestehender Managementwerkzeuge ist zunächst die Analyse der durch die Werkzeuge erbrachten Funktionalitäten sowie die Erfassung der relevanten Managementdaten notwendig. Diese beiden Aspekte (Managementfunktionalität, Managementdaten) stellen die Grundlage dar, um die durch den Dienstentwurf konstruierten Schnittstellen auf die bestehenden Werkzeuge umzusetzen. Wie in Abschnitt 4.1 (Seite 73) aufgezeigt, ist bei der Einordnung in den Softwareentwicklungsprozess zu beachten, dass die Modellierung der Fähigkeiten der bestehenden Managementwerkzeuge auf vielfältige Arten und Weisen erfolgen kann, wobei eine zwangsläufige Konformität zum vorgestellten Metamodell der Domäne nicht unbedingt gegeben sein muss. Eine Überarbeitung der entstandenen Modelle kann erforderlich sein und muss daher Bestandteil eines methodischen Vorgehens für die dienstorientierte Integration zur Abbildung automatisierbarer Betreiberprozesse sein.

Im Gegensatz zur vorwärts-orientierten Herangehensweise für den Entwurf von Dienstschnittstellen auf Basis fachfunktionaler Anforderungen wird bei der Integration bestehender Softwareartefakte eine

rückwärts-orientierte Herangehensweise angewandt. Für die Integration bestehender Managementwerkzeuge werden die durch die dienstorientierte Analyse ermittelten Dienstkandidaten genutzt, um definierte Integrationspunkte festzulegen. Hierdurch kann sichergestellt werden, dass die von den bestehenden Managementwerkzeugen angebotenen Funktionalitäten und Daten ausgerichtet an fachfunktionalen Anforderungen erfasst werden.

Prinzipiell bestehen unterschiedliche Ausgangssituationen bei der Integration eines bestehenden Managementwerkzeuges, die auf die weitere konkrete Ausprägung der für die Integration nötigen Schritte im Entwicklungsprozess eine Auswirkung besitzen:

- **Fall 1**: Es ist keine Programmierschnittstelle zu Managementfunktionalitäten und/oder Managementdaten des Werkzeuges vorhanden oder es ist eine Schnittstelle vorhanden. Managementwerkzeuge, die dieses Charakteristikum aufweisen, sind nicht Gegenstand der Betrachtungen des vorliegenden Ansatzes, da dadurch zusätzlicher Aufwand entsteht (z. B. durch die Konstruktion einer nachempfundenen Benutzerinteraktion und somit die Anforderung nach der Unabhängigkeit von konkreten Managementwerkzeugen nicht erfüllt wird.

- **Fall 2**: Es ist eine Schnittstelle zu den Managementfunktionen eines Werkzeuges vorhanden. Diese Schnittstelle ist nicht dienstorientiert; die Konstruktion einer Adapterkomponente ist erforderlich, die eine Nutzung der Werkzeuglogik als Implementierung eines Managementdienstkandidaten ermöglicht. Hierzu wird die Spezifikation einer Ontologie bezüglich Funktionen und Daten des bestehenden Werkzeuges erforderlich, um die Abbildung von benötigter und vorhandener Semantik nachvollziehbar gestalten zu können. Hierzu ist sowohl die Betrachtung der Daten als auch der angebotenen Funktionen des zu integrierenden Werkzeuges erforderlich.

- **Fall 3**: Es ist eine Schnittstelle zu den Managementfunktionen eines Werkzeuges vorhanden. Diese Schnittstelle ist dienstorientiert, womit die Spezifikation des funktionalen und semantischen Bezuges der Dienstoperationen vorliegt. In diesem Fall kann eine Adapterkomponente wie in Fall 2 konstruiert werden, die die Semantik der verschiedenen vorliegenden Schnittstellen umsetzt.

- **Fall 4**: Es ist eine Schnittstelle zu einem Werkzeug vorhanden, die dienstorientiert gestaltet ist. Diese Schnittstelle ist auf der Grundlage des im vorigen Kapitel spezifizierten integrierten Entwicklungsvorgehens konstruiert worden, wodurch die Semantik von angebotenen Dienstoperationen mitsamt zugehörigen Nachrichten klar definiert ist. In diesem Fall kann das bestehende Werkzeug direkt als Implementierung der identifizierten Dienstkandidaten herangezogen werden. Dadurch wird der Referenzcharakter der vorgestellten Methode verdeutlicht.

Die Gesamtheit der für die Integration notwendigen Schritte ist in Abbildung 30 im Überblick dargestellt. Das im vorigen Kapitel vorgestellte Metamodell der Domäne IT-Management ermöglicht die Erfassung fachlicher Anforderungen in Form verschiedener Domänenmodelle. Diese Domänenmodelle dienen als Grundlage zur Spezifikation von Kandidaten für Managementdienste und im weiteren Verlauf eines Softwareentwicklungsprozesses von abstrakten Managementdiensten. Diese abstrakten Managementdienste werden in SoaML durch die Definition von `ServiceInterfaces` festgelegt und dienen bei der Integration eines bestehenden Werkzeuges als fixe Orientierungspunkte. Im Gegensatz zum vorwärts-orientierten Vorgehen bei der Spezifikation von fachlich-motivierten

Managementdiensten ist bei der Integration zusätzlich ein rückwärts-orientiertes (engl. *bottom up*) Vorgehen erkennbar, da hier zunächst von einer bestehenden Komponente abstrahiert wird. Dieser zusätzliche Schritt kann unabhängig von der eigentlichen Definition der fachlich-motivierten Managementdienste erfolgen, wodurch eine Abkopplung der anfallenden Integrationsaufgaben vom eigentlichen Entwicklungsprozess möglich wird. Ersichtlich wird hierbei auch der auf Wiederverwendung ausgelegte Ansatz, bestehende Managementdienste zur Umsetzung erweiterter fachlicher Anforderungen heranzuziehen.

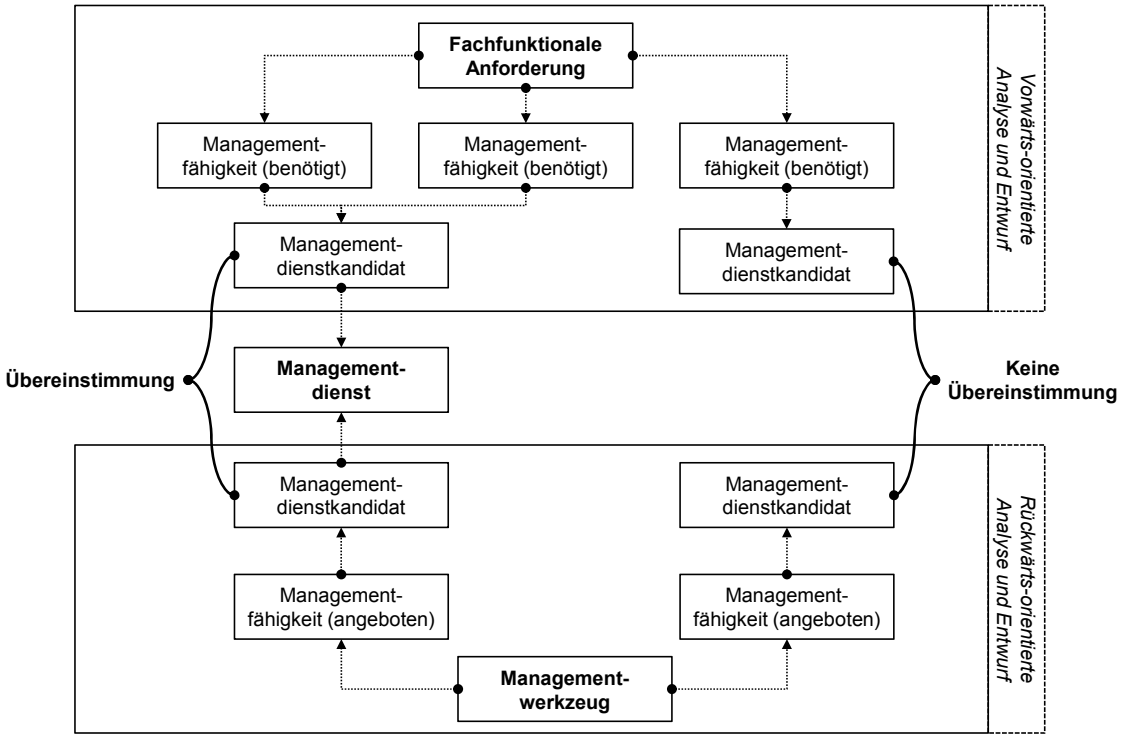

Abbildung 30 Prinzipielle dienstorientierte Integration im Überblick

Im Kern der Frage bei diesem Vorgehen steht die Feststellung der Übereinstimmung zwischen ermittelten Managementdienstkandidaten, die sich aus fachfunktionalen Anforderungen ermitteln lassen, und denjenigen Dienstkandidaten, die sich aus der Analyse bestehender Werkzeuge und den hieraus resultierenden Entwurfsschritten ergeben. Im Falle, dass sich keine Übereinstimmungen zwischen benötigten und angebotenen Dienstkandidaten ergeben, kann hier keine bestehende Implementierung zur Realisierung fachlicher Anforderungen genutzt werden.

Um eine Übereinstimmung zwischen den Managementdienstkandidaten des vorwärts-orientierten und rückwärts-orientierten Entwurfes zu erreichen, ist die Interpretation der durch die Dienstkandidaten angebotenen Managementfunktionen in Form der Managementfähigkeiten notwendig. Im einfachsten Fall kann eine Überdeckung ermittelt werden, indem ein semantischer Bezug der verschiedenen Dienstoperationsnamen hergestellt wird. Da jedoch nicht zwangsläufig eine namentliche Übereinstimmung vorherrschen muss (z. B. führt der Einsatz unterschiedlicher natürlicher Sprachen während der Analysephase zwangsläufig zu Inkonsistenzen bei den Bezeichnern), kann der Einsatz

entsprechender Ansätze für die formale Definition der einem Bezeichner zugeordneten Semantik die Abbildung benötigter Managementfähigkeit auf angebotene Managementfähigkeit zielgerichtet unterstützen.

5.2 Domänenmodell und Betreiberprozessmodell

Die Grundlage für den Entwurf von Managementdiensten bildet die Analyse der Anforderungen. Auf Basis des im vorigen Kapitels vorgestellten Metamodells der Domäne wird im Folgenden aufgezeigt, wie die Artefakte der Domänenmodellierung zielgerichtet konstruiert werden können. Die beiden wesentlichen Aspekte der strukturellen als auch dynamischen Perspektive werden durch das Domänenmodell sowie darauf aufbauend durch das Betreiberprozessmodell dargestellt.

5.2.1 Das Domänenmodell

Referenzmodelle

Der nachvollziehbare Entwurf wiederverwendbarer Managementdienste zur Abbildung automatisierbarer Betreiberprozesse kann durch den Einsatz eines Metamodells der Domäne, wie in den vorangegangenen Kapiteln gezeigt, zielgerichtet unterstützt werden. Um wiederkehrende Schritte in der dienstorientierten Analyse und dem dienstorientierten Entwurf zu vermeiden, können auf Basis des vorgestellten Metamodells verschiedene Referenzmodelle für verschiedene Managementbereiche entwickelt werden, die in realen Entwurfsprojekten die Erfassung fachfunktionaler Anforderungen und die Ableitung von hieraus resultierenden Managementdienstkandidaten nachhaltig unterstützen. Diese Referenzmodelle definieren verbindliche Aspekte verschiedener Managementbereiche, indem z. B. verschiedene Teilnehmer, Aktivitäten oder Entitäten klar und eindeutig definiert und festgelegt werden.

Da durch die Umsetzung des konzeptionellen Modells der Domäne auf eine OWL-Ontologie [W3C-OWL] prinzipiell die Möglichkeit eröffnet wird, unterschiedliche Bedeutungen von Modellelementen in erstellten Artefakten einzubeziehen und durch den Einsatz von automatisierten Klassifikationsmechanismen zu erfassen, bietet sich die Konstruktion verschiedener Referenzmodelle für die unterschiedlichen Managementbereiche ebenfalls in Form von OWL-Ontologien an. Deswegen kann einfach der Bezug zum Metamodell durch die Nutzung von *Object Properties* zwischen Metamodellelementen und entsprechenden Modellelementinstanzen erreicht werden. Abbildung 31 demonstriert diese Möglichkeit am Beispiel der Managemententität Störungsbeleg (engl.: *Incident Record*).

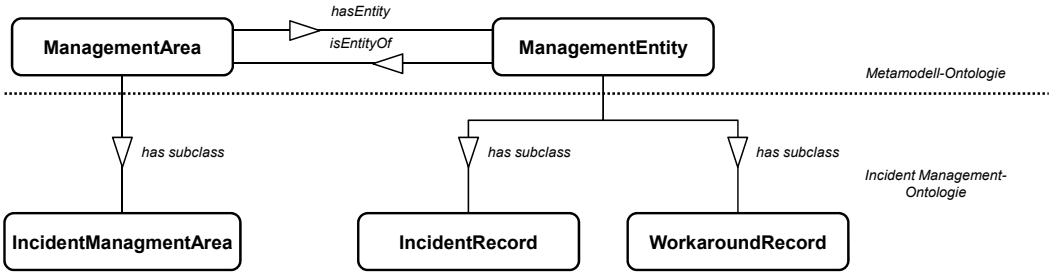

Abbildung 31 Instanziierung eines Metamodellelementes mithilfe einer OWL object property

Die Konstruktion verschiedener OWL-Ontologien hat zum Ziel, formale Referenzmodelle der jeweils betrachteten Managementbereiche zu erschaffen. Zur Ausgestaltung der jeweiligen Ontologie wird daher eine Abstraktion der in ISO/IEC20000-1:2005 definierten Managementbereiche angestrebt. Im Beispiel werden für den Managementbereich `IncidentManagementArea` zwei hierin enthaltene Entitäten (`IncidentRecord` und `WorkaroundRecord`) definiert und durch die `isA`-Objekteigenschaft (engl. *Object Property*) den entsprechenden Metamodellelementen zugewiesen. Durch den Einsatz entsprechender Entwicklungswerkzeuge für eine Ontologie (z. B. Protégé [Protege]) kann die Nutzung von Klassifikationsalgorithmen eine Zuordnung der verschiedenen *Incident-Management*-Ontologieelemente auf Basis der entsprechenden Metamodell-Ontologieelemente durchgeführt werden.

Eine Ontologie für die Störungsbearbeitung

Da die Störungsbearbeitung einen zentralen Stellenwert innerhalb der Domäne IT-Management inne hat, wird anhand der bei der Störungsbearbeitung beteiligten Managementbereiche aufgezeigt, inwiefern das in der Definition des Standards ISO/IEC20000-1:2005 vorgegebene Wissen bezüglich funktionaler Mindestanforderungen an ein Managementsystem formal dargestellt werden kann. Als ergänzender Beitrag in diesem Kapitel wird daher im Folgenden eine Ontologie für die an der Störungsbearbeitung beteiligten Managementbereiche vorgestellt. Hierdurch wird die Anwendung des Metamodells unter dem Einsatz einer OWL-Ontologie aufgezeigt. Prinzipiell kann für jeden weiteren der dreizehn der durch die Standarddefinition für Betreiberprozesse vorgegebenen Managementbereiche eine entsprechende Referenzmodellierung durchgeführt werden. Aufgrund der Charakteristika der an der Störungsbearbeitung beteiligten Managementbereiche eignen sich die im Folgenden betrachteten Managementbereiche *Incident Management*, *Problem Management*, *Change Management*, *Release Management* und *Configuration Management* besser, eine Unterstützung bei der Ausführung automatisierbarer Betreiberprozesse als die übrigen Bereiche [Br06, SB08, GP+08].

Insgesamt gliedert sich die Störungsbearbeitung in die gemäß ISO/IEC20000-1:2005 definierten Managementbereiche *Incident Management*, *Problem Management*, *Change Management*, *Release Management*, und *Configuration Management*. Der Zusammenhang der einzelnen Managementbereiche innerhalb der Störungsbearbeitung ist in der folgender Abbildung 32 dargestellt.

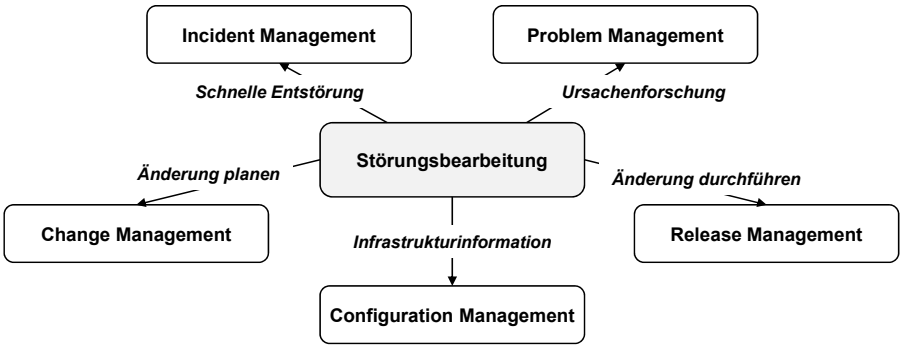

Abbildung 32 An der Störungsbearbeitung beteiligte Managementbereiche

Insgesamt sind verschiedene Aspekte bei der Störungsbearbeitung auf verschiedene Managementbereiche aufgeteilt. Somit können die jeweiligen Ziele eines Managementbereiches klar definiert und voneinander abgetrennt werden. So werden bei der Definition des *Incident Management* funktionale Mindestanforderungen für das Ziel einer schnellstmöglichen Behebung eines aufgetretenen Fehlers definiert. Für eine tiefergreifende Untersuchung einer aufgetretenen Störung ist dagegen die Definition funktionaler Mindestanforderung des *Problem Managements* heranzuziehen. Eine vergleichbare Aufteilung verschiedener Aspekte kann bei der Planung und Durchführung von Änderungen an Infrastrukturelementen festgestellt werden. Für die Planung durchzuführender Änderungen werden funktionale Mindestanforderungen im Managementbereich *Change Management* definiert, während die eigentliche Durchführung von geplanten Änderungen im *Release Management* festgelegt wird. Als letzter Managementbereich ist das *Configuration Management* in die Bearbeitung einer Störung involviert. In diesem Managementbereich sind funktionale Mindestanforderungen an ein Managementsystem definiert, das querschnittliche Funktionalität anbietet, um den einzelnen Managementbereichen qualifizierte Managementinformationen zur Verfügung zu stellen.

Die hier abgebildeten grafischen Darstellungen der OWL-Ontologie sind aus Platzgründen lediglich auszugsweise dargestellt. Eine vollständige Spezifikation der erstellten Ontologien befindet sich in den Anhängen (Abschnitt H.Artefakte).

Incident Management

Die zielgerichtete Bearbeitung einer aufgetretenen Störung erfolgt gemäß ISO/IEC20000-1:2005 durch zwei verschiedene Managementbereiche. Während das *Incident Management* die schnellstmögliche Behebung einer Störung zum Ziel hat, fokussiert das *Problem Management* eine tiefergehende Ursachenuntersuchung. Beide Managementbereiche sind daher eng miteinander verbunden.

Im Incident Management werden geeignete Maßnahmen durchgeführt, um eine gemeldete Störung so schnell wie möglich zu beheben. Hierzu gehört aber auch, auf Anfragen von Dienstkunden zu reagieren. Beispiele hierfür sind in der Rücksetzung eines Passwortes zu finden. Da solche Anfragen bezüglich eines Dienstes (engl. *Service Request*) gleichermaßen eine Beeinträchtigung des

vereinbarten IT-Dienstes für einen Dienstnutzer bedeuten, ist es zielführend, auch solche einen vereinbarten Dienst betreffende Anfragen durch den *Incident-Management*-Prozess zu bearbeiten.

Insgesamt können einige wesentliche Elemente zur Modellierung des Managementbereiches unabhängig von einem konkreten Betreiberszenario ermittelt werden, wie in nachfolgender Tabelle 10 dargestellt ist.

Metamodellelement	**Modellelement**
Managementbereich	*Incident Management*
Managemententität	- Datenhoheit: *Incident Record* - Weitere Managemententitäten: *Know Error Record*, *Workaround Record*, *Configuration Management Database Record* (*CMDB Record*)
Managementaktivität	*Record Incident*, *Determine Business Impact*, *Prioritize Incident*, *Classify Incident*, *Escalate Incident*, *Resolve Incident*, *Inform Customer*, *Create Incident Record*, *Update Incident Record*
Managementrichtlinie	*Incident Record Structure Policy*

Tabelle 10 Elemente des Managementbereiches Incident Management

Aus den identifizierten Managementaktivitäten können funktionale Mindestanforderungen abgeleitet werden, die durch geeignete Managementsysteme unterstützt werden müssen. Hiermit ist – wie bei allen weiteren Definitionen von Managementbereichen – noch keine konkrete Ausgestaltung eines Managementprozesses verknüpft. Hierdurch kann auf die jeweiligen Situationen bei konkreten Betreibern eingegangen werden, indem z. B. verschiedene Managementaktivitäten durch jeweils unterschiedliche beteiligte Akteure ausgeführt werden. Für den Entwurf von Managementbasisdiensten ist dieser Aspekt dahingehend unwichtig, da durch Managementbasisdienste ja genau diejenigen Managementfunktionen umgesetzt werden sollen, die zunächst kontextunabhängig – also unabhängig von bestimmten Akteuren in einem prozessorientierten Kontext – sind.

Die identifizierten Modellelemente sind wie in nachfolgender Abbildung 33 dargestellt in einer auf OWL-basierenden Ontologie spezifiziert. Somit wird die den einzelnen Modellelementen zugrunde liegende Semantik formal definiert, wodurch die erstellten Referenzmodelle im Kontext modell-getriebener Entwicklungsvorgehen einsetzbar werden.

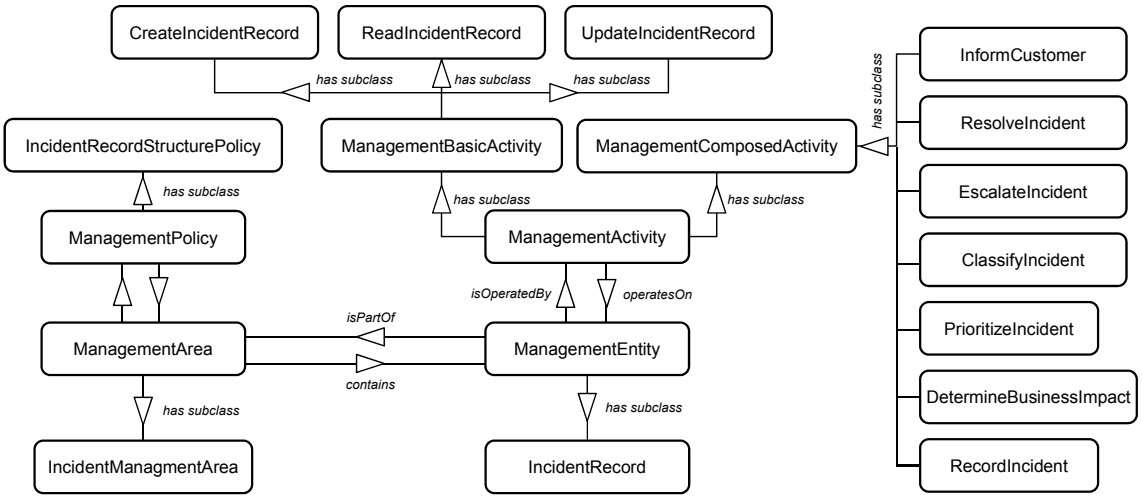

Abbildung 33 Ausschnitt aus der OWL-Ontologie für das Incident Management

Aus der Analyse der Elemente des Managementbereiches *Incident Management* kann eine OWL-Ontologie erstellt werden. Abbildung 33 zeigt einen Ausschnitt aus der modellierten OWL-Ontologie mit den zu den jeweiligen Metamodellinstanzen zugehörigen Metamodellelemente. Die Elemente des entsprechenden Ausschnittes aus dem Metamodellbereich sind farblich leicht hervorgehoben.

Managementaktivitäten zur Manipulation der im Incident Management erforderlichen Managemententitäten *Known Error Record*, *Problem Resolution Record* sowie *CMDB Record* werden in den Managementbereichen Problem Management sowie Configuration Management definiert. Im *Incident Management* wird lediglich lesender Zugriff benötigt. Aus diesem Grund wird die Definition dieser Managemententitäten in die den Kontext der Definition der entsprechenden Domänenmodelle gestellt.

Problem Management

Während das *Incident Management* eine schnellstmögliche Behebung einer aufgetretenen Störung zum Ziel hat, wird im *Problem Management* eine tiefergehende Untersuchung gemeldeter oder identifizierter Störungen durchgeführt. In verschiedenen Managementaktivitäten der beiden Managementbereiche wird daher auf Managemententitäten des jeweils anderen Managementbereiches zurückgegriffen, eine Trennung der beiden Zielsetzungen in verschiedene Managementbereiche ist dennoch zielführend.

Die Festlegung der verschiedenen kausalen Aspekte ist zunächst unabhängig von konkreten Situationen in einer Betreiberorganisation und kann daher unabhängig des jeweiligen Kontexts angegeben werden, wie in nachfolgender Tabelle 11 dargestellt ist.

Metamodellelement	Modellelement

Managementbereich	*Problem Management*
Managemententität	- Datenhoheit: *Known Problem Record*, *Known Error Record*, *Workaround Record* - Weitere Managementbereiche: *Request for Change Record*
Managementaktivität	*Record Problem, Identify Impact, Classify Problem, Escalate Problem, Resolve Problem, Create Problem Record, Update Problem Record, Read Problem Record, Create Problem Resolution Record, Read Problem Resolution Record, Update Problem Resolution Record, Create Known Error Record, Read Known Error Record, Update Known Error Record*
Managementrichtlinie	*Problem Record Structure Policy, Problem Resolution Record Structure Policy, Known Error Record Structure Policy, Known Problem Structure Policy*

Tabelle 11 Elemente des Managementbereiches Problem Management

Prinzipiell kann der Managementbereich *Problem Management* bezüglich kausaler Aspekte in das proaktive bzw. reaktive *Problem Management* aufgeteilt werden. Beim proaktiven *Problem Management* wird versucht, durch Trendanalysen eventuell Tendenzen für sich änderndes Verhalten von IT-Diensten zu ermitteln, um dadurch vorausschauend zukünftige auftretende Störungen zu identifizieren. Beim reaktiven *Problem Management* werden geeignete Maßnahmen definiert, um im Falle schwerwiegender Störungen direkt eine Änderung (beispielsweise in Form temporärer Problemlösungen) durchführen zu können.

Die im *Problem Management* notwendigen Managemententitäten gliedern sich entlang den unterschiedlichen Charakteristika der jeweiligen Artefakte während des Ablaufes in der Problemuntersuchung. Kann im *Incident Management* keine direkte Entstörung eines Dienstes durchgeführt werden, wird aus dem entsprechenden *Incident Record* im *Problem Management* zunächst ein *Known Problem Record* erstellt. Stellt sich im weiteren Verlauf heraus, dass die gemeldete Störung einfach behoben werden kann (typischerweise durch Eingriffe durch den eine gemeldete Störung betreuenden Mitarbeiter beim Betreiber oder direkt durch den meldenden Nutzer selber), werden die zur Anwendung der Lösung erforderlichen Schritte im *Workaround Record* dokumentiert und dem *Incident Management* für zukünftige Störungsmeldungen zur Verfügung gestellt. Kann keine direkte Entstörung durchgeführt werden, weil z. B. eine komplexe Änderung an Bestandskomponenten der IT-Infrastruktur erforderlich sind, wird ein *Known Error Record* generiert. In diesem Fall wird gleichzeitig ein Änderungswunsch (*Request for Change*) dokumentiert und dem *Change Management* übergeben.

Wie auch bei der Definition des entsprechenden Modells für das *Incident Management* wird bei der Definition des Domänenmodells für das *Problem Management* auf diejenigen Managemententitäten

oder Managementaktivitäten fokussiert, die direkt im Kontext des Managementbereiches eingeordnet werden können.

Change Management

Im Gegensatz zu den beiden Bereichen *Incident Management* und *Problem Management* wird im *Change Management* (Änderungsmanagement) nicht die direkte Behebung einer aufgetretenen Störung betrachtet, sondern es werden diejenigen Aktivitäten betrachtet, die bei der Änderung von Infrastrukturkomponenten eines (gestörten) IT-Dienstes relevant sind. Ziel ist es, Änderungen zu planen, damit im nachfolgenden *Release Management* eine geplante Änderung umgesetzt werden kann. Ähnlich zur Trennung der bei einer aufgetretenen Störung unterstützenden Managementbereiche Incident und Problem Management wird bei der Planung und Durchführung einer Änderung eine Trennung verschiedener Aspekte vorgenommen. Hierdurch wird der organisatorische Ablauf bei der Planungsänderung vom eigentlichen Durchführen der Änderung getrennt.

Änderungen können nicht direkt von den Nutzern eines IT-Dienstes ausgelöst werden, sondern müssen über die Aktivitäten des *Incident Management* oder weiterer Managementbereiche (z. B. *Service-Level*-Management bei Änderungswünschen die Eigenschaften eines vereinbarten IT-Dienstes betreffend) angestoßen werden.

Metamodellelement	Modellelement
Managementbereich	*Change Management*
Managemententität	- Datenhoheit: *Request for Change Record, Change Plan Record*
Managementaktivität	*Record Request for Change, Classify Request for Change, Assess Request for Change, Reverse Change, Approve Change, Review Change, Create Request for Change Record, Read Request for Change Record, Update Request for Change Record, Create Change Plan Record, Read Change Plan Record, Update Change Plan Record*
Managementrichtlinie	*Request for Change Structure Policy, Change Plan Structure Policy*

Tabelle 12 Elemente des Managementbereiches Change Management

Insgesamt werden die Managementfunktionen des *Change Management* daher nur unmittelbar an der organisatorischen Grenze eines IT-Dienstleisters genutzt und entsprechen aus disem Grund den in [EH+08] beschriebenen Teildiensten. Insgesamt weist der Managementbereich *Change Management* ein hohes Maß an organisatorischer Struktur auf. Eine Automatisierung durch Werkzeugunterstützung ist jedoch nur mittelbar möglich [Br06], da nicht ausgeschlossen werden kann, dass die Planung von

Änderungen die Interaktion menschlicher Teilnehmer erfordert. Durch die Auslagerung der Durchführung der eigentlichen Änderungen in Aktivitäten des Managementbereiches *Release Management* beschränkt sich die Zielsetzung des *Change Management* auf die Planung, Zuteilung, Überwachung und Steuerung von Änderungen.

Zentrale Managemententität im *Change Management* ist der Änderungseintrag (*Request for Change Record*). Zur Dokumentation der geplanten Änderungen sowie zur Steuerung der eigentlichen Änderungsdurchführung wird ein *Change Plan Record* erstellt.

Release Management

Im *Release Management* werden durch das *Change Management* geplante Änderungen durchgeführt, und im Fehlerfall wieder rückgängig gemacht. Ziel ist es, geplante Änderungen schnell und zielführend in die Produktionsumgebung einer IT-Infrastruktur zu überführen. Hierbei können geplante Änderungen zunächst in hierfür dediziert vorgesehene Testumgebungen evaluiert werden. Da die Managementaktivitäten des *Release Management* lediglich den ausführenden Teil bei der Durchführung geplanter Änderungen beinhalten, ist ein Entwurf von Managementdiensten mit dem Entwurf von Managementdiensten für das *Change Management* sinnvoll. Eine Trennung beider Managementbereiche ist jedoch aus Gründen der unterschiedlichen Zielsetzungen der einzelnen Bereiche erforderlich.

Metamodellelement	Modellelement
Managementbereich	*Release Management*
Managemententität	- Datenhoheit: *Release Plan* - Weitere Managementbereiche: *Request for Change*
Managementaktivität	*Plan Release, Reverse Release, Assess Release Impact, Update Configuration Information, Build Release, Test Release, Report Release Statistics, Create Release Plan Record, Read Release Plan Record, Update Release Plan Record*
Managementrichtlinie	*Release Plan Structure Policy*

Tabelle 13 Elemente des Managementbereiches Release Management

Für eine optimale Unterstützung und Akzeptanz bei der Durchführung von Änderungen wird gefordert, dass der Betreiber einer Infrastruktur mit den jeweiligen, eine Änderung betreffende Nutzern eine Vereinbarung trifft, die den Umfang sowie die Häufigkeit regelmäßiger Änderungen umfasst. Hierzu gehört beispielsweise die Vereinbarung von regelmäßigen Wartungsfenstern, die

entsprechende Nicht-Verfügbarkeitszeiträume regelt. Diese Vereinbarungen werden im *Release Plan* festgehalten.

Neben den eigentlichen Managementaktivitäten zur Durchführung einer geplanten Änderung wird gefordert, dass statistische Daten zu den Aktivitäten dieses Bereiches bereitgestellt werden.

Configuration Management

Zielsetzung des Managementbereiches *Configuration Management* ist die Bereitstellung von Managementinformation bezüglich der von einem IT-Dienstleister betriebenen IT-Infrastruktur. Hierzu werden neben der Abbildung von Hardware- und Softwaregegenständen vor allem auch die von den einzelnen Managementbereichen eingeführten Managemententitäten hinzugezählt. Diese Information wird in einer logischen zentralen Komponente bereitgehalten (engl. *Configuration Management Database*, CMDB). Um einen vollständigen integrierten Ansatz für den Betrieb vernetzter Infrastrukturen zu ermöglichen, ist ein enger Bezug der Einträge einer CMDB mit den entsprechenden Abstraktionen von zu betreibenden Hardware- und Softwarekomponenten der entsprechenden technischen Überwachungssysteme erforderlich.

Metamodellelement	Modellelement
Managementbereich	*Configuration Management*
Managemententität	-Datenhoheit: *Configuration Item*
Managementaktivität	*Create Configuration Item, Update Configuration Item, Read Configuration Item, Determine Relationships, Take Baseline*
Managementrichtlinie	*Configuration Item Structure Policy*

Tabelle 14 Elemente des Managementbereiches Configuration Management

Insgesamt stellt dieser Managementbereich zentrale, von allen weiteren Managementbereichen genutzte Managementfunktionalitäten bereit, wenngleich die geringe organisatorische Struktur dieses Bereiches einen geringen Nutzen aus einer möglichen Automatisierung komplexer Abläufe erzielen lässt. Unter dem Gesichtspunkt der Vermeidung von Fehlern beim Anlegen oder Ändern von Einträgen in der CMDB ist die Existenz eines technischen Systems erforderlich. Da die diesem Managementbereich zugeordnete Managemententität in der Fachliteratur durchgängig als *Configuration Item* bezeichnet wird, wird an dieser Stelle von dem Benennungsschema, den Managemententitäten des jeweiligen Managementbereichs das Suffix *Record* anzuhängen, abgewichen.

Entwurf weiterer Referenzmodelle

Die betrachteten Referenzmodelle für die Störungsbearbeitung zeigen den Ansatz auf, die in der Definition des Standards ISO/IEC20000-1:2005 [ISO05] festgelegten Managementbereiche in Form weiterer Referenzmodelle zu spezifizieren. Für alle weiteren zu betrachtenden Bereiche gilt, dass die wesentlichen strukturellen Modellelemente Managementbereich, Managemententität, Managementaktivität und Managementrichtlinie in die zu definierenden Referenzmodelle einfließen, während die übrigen Modellelemente bei der Untersuchung eines konkreten Szenarios betrachtet werden.

Erstellen von fachlichen Modellen

Auf Basis der vorgestellten semantischen Domänenmodelle können in realen Entwicklungsprojekten konkrete Anforderungen bei der Umsetzung innerhalb einer Betreiberorganisation aufgegriffen werden. Während die vorgestellten Ontologien einen grundlegenden Ausgangspunkt bilden und aufgrund ihres Referenzcharakters sich stark an den funktionalen Mindestanforderungen aus der Standardspezifikation von ISO/IEC20000-1:2005 orientieren, werden durch die Betrachtung konkreter Betreiberorganisationen zusätzliche Aspekte relevant.

Da die Modellierung der fachlichen Anforderungen ein kreativer Vorgang ist, ist die Festlegung auf eine genaue Abfolge von einzelnen verschiedenen Modellierungsschritten wenig zielführend. Für den Fall, dass ein Referenzmodell vorliegt (z. B. aus früheren Entwicklungsprojekten oder aufgrund des Einsatzes einer der vorgestellten Referenzmodelle) wird keine vollständige Modellierung der zu unterstützenden Domäne vorgenommen, da hier Teile der Modelle wiederverwendet werden können. Ist dies nicht der Fall, erfolgt eine Konstruktion von Domänenmodellen, die die wesentlichen fachlichen Aspekte beinhalten und konform zu dem im vorigen Kapitel vorgestellten Domänenmetamodell sind.

Die ungefähre Struktur im Ablauf zur Erstellung von fachlichen Modellen ist in nachfolgender Abbildung in Form eines Pseudo-BPMN-Diagramms dargestellt. Hierzu wurde auf die Definition von Pools/Lanes verzichtet.

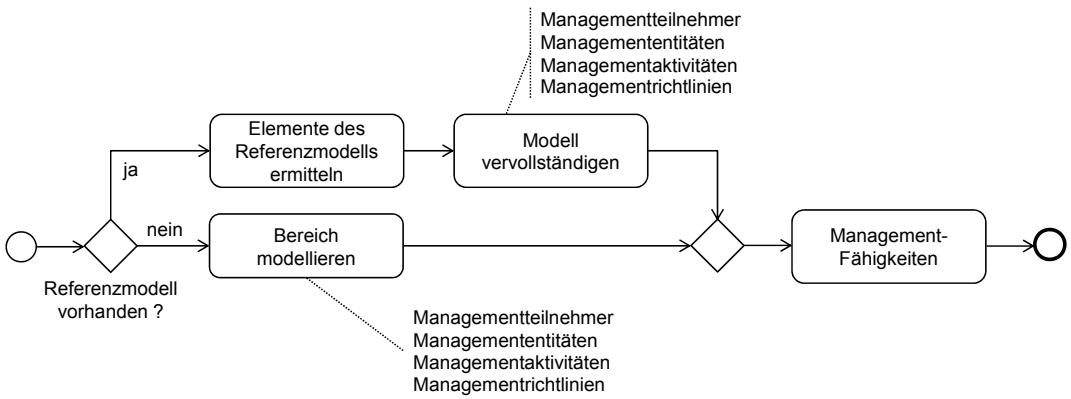

Abbildung 34 Schritte zur Erstellung von Domänenmodellen

Werden bestehende Referenzmodelle eingesetzt, müssen diese ergänzt werden. Hierzu gehört vor allem die Festlegung, welche beteiligten Akteure eine Betreiberorganisation im jeweiligen Szenario beteiligt sind. Ist bereits in dieser Phase erkennbar, dass identifizierte Managementaktivitäten als Komposition von weiteren Managementaktivitäten realisiert werden, kann in den entsprechenden Modellen eine Verfeinerung in die jeweiligen Metamodellelemente Basismanagementaktivität und zusammengesetzte Managementaktivität vorgenommen werden. Dies stellt einen ersten Schritt zur Abbildung von automatisierbaren Betreiberprozessen dar.

Um die in den Domänenmodellen identifizierten Managementaktivitäten im weiteren Entwicklungsvorgehen durch geeignete Operationen von Managementdiensten zu unterstützen, ist eine Konkretisierung der jeweiligen Modellelemente erforderlich. Hierbei soll die durch eine Managementaktivität bezeichnete Semantik erhalten bleiben. Im Metamodell der Domäne wird für diesen Zweck das Metamodellelement Managementfähigkeit eingeführt. Durch die Entkopplung von implementierender Managementdienstoperation und unterstützender Aktivität durch das Modellelement Managementfähigkeit kann ein Austausch der Implementierung vorgenommen werden, ohne eine Änderung auf der fachlichen Ebene vornehmen zu müssen.

Modellierung bestehender Managementwerkzeuge

Sind keine direkt nutzbaren Dienstschnittstellen bei der Integration bestehender Managementwerkzeuge vorhanden, ist bei der Erstellung von Werkzeugmodellen die Beteiligung der entsprechenden Werkzeugnutzer erforderlich. Prinzipiell sind die Schritte vergleichbar, da durch die Elemente des Metamodells eine Modellierung bestehender Managementwerkzeuge ermöglicht wird. Diese Eigenschaft stellt letztlich eine wesentliche Anforderung an die verfolgte Lösung dar (Anforderung 1.2).

In nachfolgender tabellarischer Übersicht ist der Einsatz der Elemente des Domänenmetamodells zur Modellierung der durch ein bestehendes Managementwerkzeug angebotenen Managementfunktionalität dargestellt.

Aspekt des bestehenden Werkzeuges	Nutzbares Metamodellelement
Implementierung der Softwarekomponente	Managementteilnehmer
Managementfunktion	Angebotene Managementfähigkeit
Managementinformation	Managemententität, Entitätsstrukturrichtlinie

Tabelle 15 Darstellbare Aspekte eines bestehenden Managementwerkzeuges

Den in der Modellierung bestehender Managementwerkzeuge beteiligten Akteuren stehen vielfältige Handlungsfreiräume zur Verfügung. Während die Konstruktion fachlicher Modelle durch den Einsatz von Referenzmodellen unterstützt werden kann und damit eine Einhaltung vorgegebener

Namenskonventionen oder semantischer Bezüge für festgelegte Konzepte ermöglicht wird, besteht diese Möglichkeit für die Konstruktion der Werkzeugmodelle aufgrund der Heterogenität der Werkzeuge in der Regel nicht. Zwar können Managementwerkzeuge, die konform zu standardisierten Informationsmodellen wie beispielsweise CIM [DMTF-CIM], SNMP, CMIP sind durch den Einsatz hierauf spezialisierter Ontologien beschrieben und in einen einheitlichen Entwicklungsprozess integriert werden [VV+02, VV+03]. Generell kann dieser Ansatz jedoch aufgrund der nicht notwendigerweise erzwungenen Konformität zu den standardisierten Informationsmodellen nicht zum Einsatz kommen.

Überarbeiten von Domänenmodellen

Die Überarbeitung der erstellen Domänenmodelle kann in mehreren iterativen Schritten erfolgen. Ziel der Überarbeitung ist eine Angleichung der ermittelten Begrifflichkeiten der beiden verschiedenen Modellaspekte, um somit den weiteren Entwurf von Managementdiensten zu vereinheitlichen. Ausschlaggebend ist die Existenz vorrangiger Modellelemente in den fachlichen Modellen gegenüber eventuell ermittelten Modellelementen in den Werkzeugmodellen. Der Grund hierfür kann in der geforderten fachlichen Ausrichtung der zu konstruierenden Dienstschnittstellen gefunden werden.

Insgesamt erfordert dieser Schritt – wie auch die beiden vorangegangen Schritte – das Fachwissen aller beteiligten Akteure.

5.2.2 Das Betreiberprozessmodell

Die Modellierung von dynamischen Abläufen innerhalb einer Betreiberorganisation hat zum Ziel, den Informationsfluss zwischen den einzelnen beteiligten Akteuren zu erfassen. Somit werden die verschiedenen Schritte in den verschiedenen Abläufen dargestellt. Die erstellten Modelle abstrahieren von konkreten technischen Details und bieten eine ganzheitliche Sicht auf den Kontext einzelner Prozessschritte. Erste Arbeiten betrachten bereits semantische Ansätze, um die Definition von Betreiberprozessmodellen nachvollziehbar zu gestalten [SB+07], eine Einordnung in einen durchgehenden Entwicklungsprozesses findet hier aber nicht statt.

Die erstellten Betreiberprozessmodelle stellen die Grundlage dar, um auf der Grundlage der identifizierten Managementbasisdienste die Komposition zu höherwertigen Managementfunktionen umzusetzen. Diese Kompositionen werden im weiteren Verlauf durch die Spezifikation von Managementprozessdiensten realisiert, wobei unterschiedliche Implementierungstechnologien zum Einsatz kommen können. Die Grundlage stellt aber in jedem Fall ein Modell des umzusetzenden Betreiberprozesses dar.

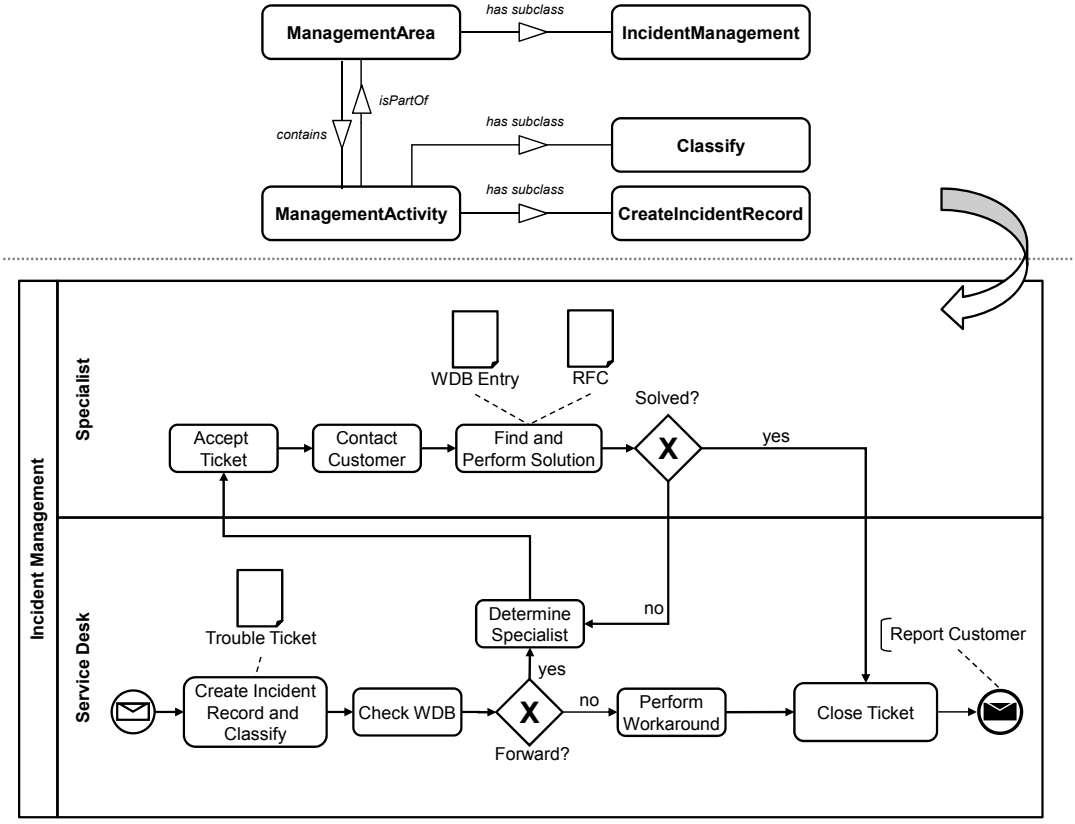

Abbildung 35 Einfaches Modell eines Incident-Management-Prozesses

Abbildung 35 stellt ein einfaches Modell für einen Betreiberprozess des Managementbereiches *Incident Management* dar. Das Modell ist in der *Business Process Modeling Notation* (BPMN) [OMG-BPMN]dargestellt. Dieser von der OMG verantwortete Ansatz ermöglicht mit der formalen Definition der statischen Semantik der einzelnen Modellelemente den Einsatz in einem auf Modellen basierenden strukturierten Entwicklungsvorgehen. Ersichtlich in diesem Modell sind die einzelnen Rollen der teilnehmenden Akteure (Service Desk, Spezialist), wobei im Prozessmodell von konkreten Akteuren abstrahiert wird. Die einzelnen Managementaktivitäten sind bei einem jeweiligen Akteur dargestellt, der Sequenzfluss der einzelnen aufeinanderfolgenden Aktivitäten durch Pfeile miteinander verbunden.

Modellierung von Betreiberprozessen

Die erstellten Modelle der Betreiberprozesse bilden einen Ausschnitt der Realität ab und fokussieren den dynamischen Aspekt in der Ablaufsteuerung komplexer Managementaktivitäten. Da die Grundlage der einzelnen Managementaktivitäten letztlich die in den strukturellen Domänenmodellen erfassten Managementaktivitäten sind, wird durch die Modellierung von Betreiberprozessen den Modellen lediglich ein weiterer Aspekt hinzugefügt. Daher wird die Modellierung der Betreiberprozesse auf Basis der zuvor erstellten Domänenmodelle betrachtet. Hierdurch wird zum einen sichergestellt, dass die in den Betreiberprozessmodellen definierten Akteure, Aktivitäten und Entitäten den in den Domänenmodellen festgelegten Bezeichnern entsprechen, zum anderen wird der

semantische Bezug der jeweiligen Modellelemente klar definiert. Dies ist insofern von Bedeutung, da die Untersuchungen zur Bewertung der Wiederverwendbarkeit eines vorliegenden Entwurfes von Managementdiensten besonders bei der Betrachtung des Aspektes der Kontextunabhängigkeit auf den einen vorliegenden Betreiberprozess definierenden Elemente bezogen ist.

Um die Definition von Betreiberprozessmodellen auf Basis der BPMN [OMG-BPMN]nachvollziehbar zu gestalten, wird ein systematisches Vorgehen festgelegt, um die einzelnen Elemente der zugrunde liegenden Domänenmodelle schrittweise in ein BPMN-Modell zu überführen. Da durch die Fokussierung auf die Erfassung der strukturellen Zusammenhänge in einem Domänenmodell zunächst keine Information über die Ausgestaltung des dynamischen Ablaufes im Betreiberprozessmodell vorliegt, erfolgt die Definition des logischen Ablaufes in einem nachgelagerten Schritt. Nachfolgende Tabelle 16 gibt eine kurze Übersicht über die prinzipielle Abbildung von Elementen aus den Domänenmodellen sowie den hierfür semantisch gleichbedeutenden Elementen im BPMN-Betreiberprozessmodell.

Element aus dem Domänenmodell	**Element aus dem Betreiberprozessmodell**
Management Area	*Pool*
Management Participant	*Lane*
Management Basic Activity	*Task*
Management Composed Activity	*Sub Process*
Management Entity	*Data Object*

Tabelle 16 Abzubildenden Elemente eines Domänenmodells in ein BPMN-Modell

Die Überführung der einzelnen Modellelemente wird im Folgenden jeweils am Beispiel des in Abbildung 35 vorgestellten *Incident-Management*-Prozesses kurz erläutert.

Pool

Die Unterteilung eines Systems in logisch zueinander gehörende Teile erleichtert nicht nur die schrittweise Verfeinerung eines komplexen Gesamtproblems, sondern ermöglicht auch die vertiefende Betrachtung einzelner Systemteile unter verschiedenen Gesichtspunkten. Bezogen auf die Modellierung von dynamischen Abläufen in Betreiberprozessen wird durch die Aufteilung in kleinere Einheiten eine Modellierung verschiedener logisch zueinander gehörender Managementaktivitäten eines Betreiberprozesses möglich. Dadurch kann auf die jeweiligen Eigenschaften eines einzelnen Betreiberprozesses eingegangen werden. Diese Aufteilung findet sich in der Definition der in ISO/IEC20000-1:2005 eingeführten verschiedenen Managementbereiche wieder und kann im BPMN-Modell durch das Modellelement *Pool* dargestellt werden.

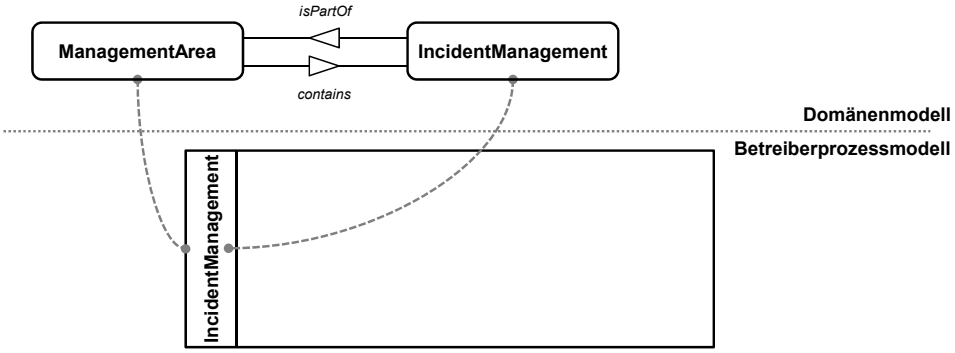

Abbildung 36 Überführen eines Managementbereiches in einen BPMN-Pool

Abbildung 36 zeigt die Überführung eines Managementbereiches in einen BPMN-*Pool* auf. Hierbei wird entsprechend der Überführung der einzelnen Metamodellelemente wie aus der Abbildung ersichtlich der Bezeichner für den konkreten Managementbereich als Benennung für den Prozess-*Pool* übernommen.

Lane

Neben der Modellierung der Außengrenze bei der schrittweisen Verfeinerung eines Betreiberprozessmodells ist vor allem auch der innere Aufbau mit den logisch zueinander gehörenden internen Verantwortlichkeiten in Form verschiedener Rollen der verschiedenen Akteure von Relevanz. Diese weitere Unterteilung kann in der BPMN durch die Unterteilung eines *Pool* in verschiedene *Lanes* dargestellt werden. Eine *Lane* stellt in einem BPMN-Prozessmodell einen Teilnehmer innerhalb eines definierten und abgeschlossen dynamischen Ablaufes dar.

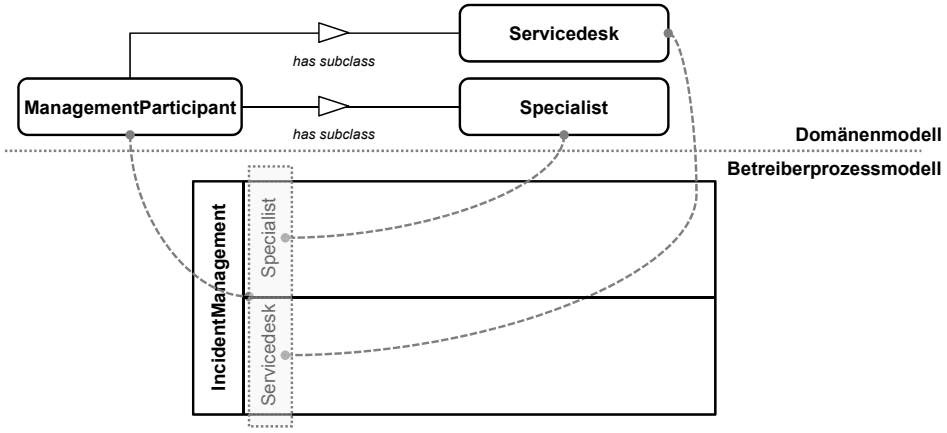

Abbildung 37 Überführung von Teilnehmern in verschiedene BPMN-Lanes

Abbildung 37 stellt die Überführung der im Domänenmodell identifizierten Teilnehmer in dem den Teilnehmern zugeordneten Managementbereich dar.

Da durch die Modellierung von Betreiberprozessen vor allem eine Sicht auf die organisatorische Ablaufsteuerung erstellt werden soll, sind in diesem Schritt die von Managementteilnehmern aktiv durchgeführten Managementhandlungen von Interesse. Diese auf die fachliche Perspektive ausgelegte Sicht geht an dieser Stelle noch nicht auf einzelne technische Systeme in Form der bestehenden Managementwerkzeuge ein. Da durch die bestehenden Managementwerkzeuge keine aktiven Managementaktivitäten durchgeführt werden, werden die als Managementteilnehmer modellierten bestehenden Managementwerkzeuge nicht bei der Überführung von Domänenmodellen in Betreiberprozessmodelle berücksichtigt. Dies kann durch eine einfache Überprüfung eines Managementteilnehmers im Domänenmodell auf Existenz einer zugeordneten Managementaktivität ermittelt werden.

Task

Mit der Definition von Prozessgrenze (BPMN-Element *Pool*) und Prozessteilnehmer (BPMN-Element *Lane*) sind die strukturellen Aspekte im Betreiberprozessmodell festgelegt, um die einzelnen Aktivitäten aus den Domänenmodellen zu überführen. Insgesamt bietet die BPMN vielfältige Modellierungselemente an, die ausgerichtet an ihrer jeweiligen Semantik, verschiedenartige Eigenschaften einer Prozessaktivität umsetzen. Diese Elemente übernehmen und verfeinern letztlich die Semantik des Modellelementes *Task*. Eine einfache Überführung der im Domänenmodell definierten einfachen Managementaktivitäten in BPMN-*Tasks* setzt diese Semantik im Prozessmodell um.

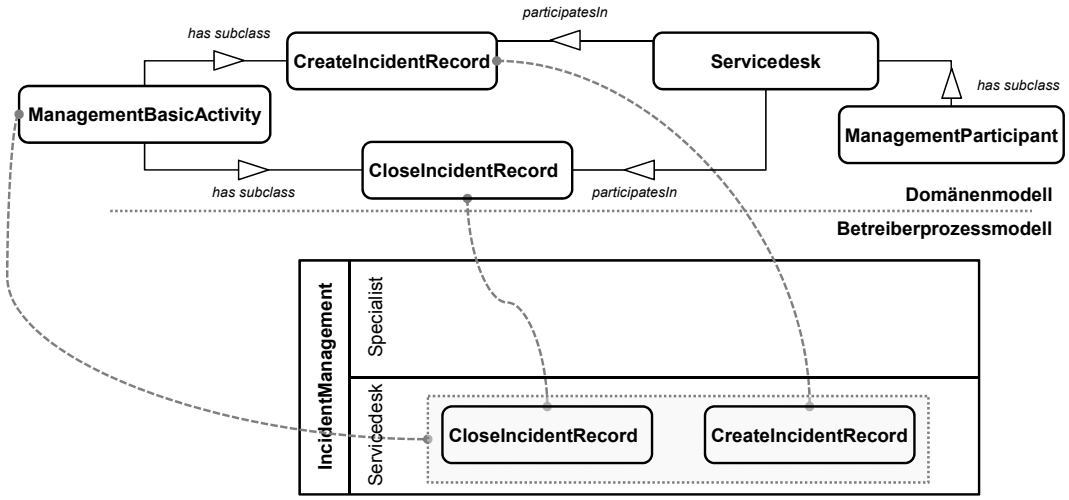

Abbildung 38 Überführung der einfachen Managementaktivitäten eines Teilnehmers

Abbildung 38 demonstriert die Überführung der verschiedenen bei einem Managementteilnehmer festgelegten Managementaktivitäten aus dem Domänenmodell. Da der Fokus bei der Konzeption des Metamodells der Domäne auf den prinzipiellen Strukturierungselementen liegt, ist in der Domänenmodellierung zunächst keine weitere Unterscheidung verschiedener Eigenschaften von Managementaktivitäten vorgesehen. Eine Ausnahme bildet die Unterscheidung in einfache Managementaktivitäten (engl. *Management Basic Activities*) und zusammengesetzte

Managementaktivitäten (engl. *Management Composed Activities*), da somit der weitere Entwurf von Managementbasisdiensten oder Managementprozessdiensten und letztlich die Bewertung der Wiederverwendbarkeit direkt und maßgeblich beeinflusst wird.

Sub Process

Neben den einfachen Managementaktivitäten werden die identifizierten und im Domänenmodell definierten zusammengesetzten Managementaktivitäten als Teilprozesse im BPMN-Betreiberprozessmodell dargestellt. Hierzu wird das Modellelement *Sub Process* eingesetzt.

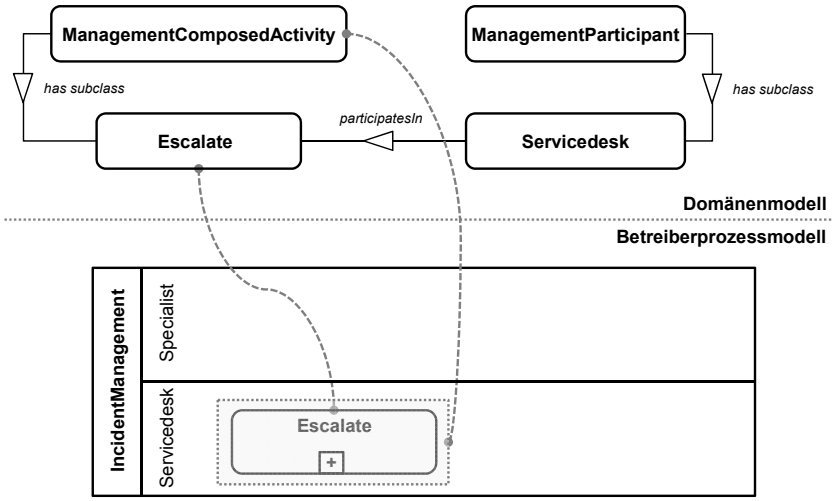

Abbildung 39 Überführung einer zusammengesetzten Managementaktivität

Die Überführung der zusammengesetzten Managementaktivitäten erfolgt im Prinzip analog zur Überführung der Basismanagementaktivitäten. Hierbei wird die einem definierten Teilnehmer zugeordnete Managementaktivität in der entsprechenden BPMN-*Lane* hinzugefügt und als BPMN-Modellelement *Sub Process* spezifiziert.

Data Object

Managemententitäten werden im BPMN-Betreiberprozessmodell durch die Definition eines *Data Object* überführt. Neben der einfachen Darstellung des *Data Objects* besteht zusätzlich die Möglichkeit, aufgrund der Semantik der eine Managemententität betreffende Managementaktivität die Art des Zugriffs auf die Managemententität im BPMN-Prozessmodell zu modellieren.

In Abbildung 40 ist diese Überführung aus dem Domänenmodell dargestellt.

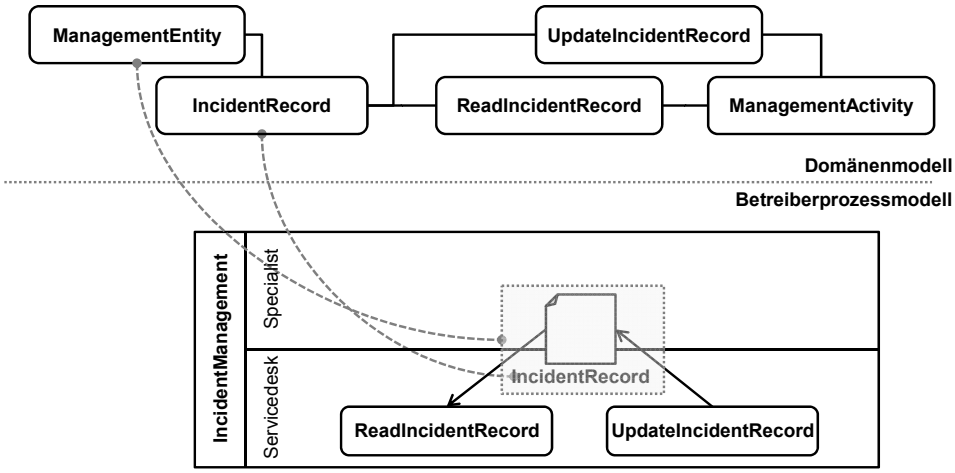

Abbildung 40 Überführen einer Managemententität

Von der Perspektive der dynamischen Ablaufmodellierung stellt eine Managemententität ein Artefakt dar, um den Zustand in der Ausführung eines Betreiberprozesses zu speichern. Um klar herauszuheben, dass verschiedene Managementaktivitäten den Zustand einer Managemententität verschiedentlich beeinflussen, kann die Richtung der jeweiligen Operation im BPMN-Modell dargestellt werden. Im Beispiel aus Abbildung 40 ist der lesende Zugriff der Aktivität `ReadIncidentRecord` bzw. der schreibende Zugriff der Aktivität `UpdateIncieentRecord` durch die Richtung des entsprechenden Pfeils dargestellt. Im weiteren Verlauf kann auf der Grundlage dieser Information aus dem Betreiberprozessmodell der Entwurf der Signaturen der eine Managementaktivität unterstützende Dienstoperation damit unterstützt werden.

Überarbeiten von Betreiberprozessmodellen

Mit der Überführung der Elemente aus den Domänenmodellen in BPMN-Betreiberprozessmodelle ist die Ausgangsbasis geschaffen, die dynamischen Abläufe eines konkret betrachteten Szenarios im Detail zu modellieren. Da noch keine logischen Abfolgen einzelner Managementaktivitäten durch die Überführung der Domänenmodelle vorliegen, kann die Anwendung verschiedener Muster eine Modellierung der Betreiberprozesse unterstützen. Da im Domänenmodell lediglich die grundlegenden und für den Entwurf von betreiberprozessorientierten und wiederverwendbaren Managementdiensten erforderlichen Aspekte Beachtung finden, kann eine Überarbeitung der erstellten Betreiberprozessmodelle erforderlich werden. Beispielsweise können verschiedene temporale Aspekte (Zeitgrenzen), kausale Aspekte (Ereignis-Abhängigkeiten) oder logische Aspekte (Bedingungen) im Domänenmodell nicht dargestellt werden. Um beispielsweise die Aspekte der Benutzerinteraktion bereits während der Modellierung der Prozesse zu erfassen, kann eine weitere Überarbeitung der Modelle nach der in [Li09] vorgestellten Methode erfolgen.

Der Einsatz von Referenzmodellen, vergleichbar zu den Referenzmodellen für die zur Störungsbearbeitung zählenden Managementbereiche, kann die weitere Konstruktion von Soll-Betreiberprozessen weiter vereinfachen. Demgegenüber steht die Erfassung des Ist-Zustandes in einem konkreten Szenario. Hierbei werden zusätzliche Details eines Betreiberprozessmodells aus einer konkreten Betreiberorganisation mit den jeweiligen Einschränkungen erfasst.

Insgesamt betrachtet stellt die Definition der dynamischen Abläufe lediglich einen Aspekt im strukturierten Entwurf wiederverwendbarer und betreiberprozessorientierter Managementdienste bereit. Je nach Zielsetzung können verschiedene Aspekte bezüglich der Optimierung der entstandenen Prozessmodelle vertieft betrachtet werden, um Rückschlüsse für eine Umgestaltung und Verbesserung im Rahmen der Einführung einer dienstorientierten Architektur zur automatisierten Ausführung von Betreiberprozessen zu erfassen.

Erfolgt eine Überarbeitung, Änderung oder Anpassung der Betreiberprozessmodelle, ist eine Rückführung dieser Änderung in die zugrunde liegenden Domänenmodelle erforderlich. Dies ist notwendig, da für den weiteren Verlauf des betrachteten Entwicklungsvorgehens die definierten Domänenmodelle zugrunde gelegt werden. Die Rückführung der geänderten Aspekte kann invers zu der Überführung der Elemente der Domänenmodelle in die Elemente der Betreiberprozessmodelle vorgenommen werden. Der Einsatz automatisierter Modelltransformationen kann in diesem Schritt die manuelle Rückführung der geänderten Aspekte den beteiligten Akteuren am Entwicklungsprozess unterstützen. Die Definition eines vollständig automatisierten und auf Modelltransformationen beruhenden Entwicklungsvorgehens ist jedoch nicht im Fokus der vorliegenden Arbeit. Im Ausblick (Abschnitt 7.3, Seite 203) werden die Einsatzmöglichkeiten für Modelltransformationen im Kontext weiterer, auf der vorliegenden Arbeit aufbauenden Beiträge skizziert.

5.3 Spezifikation von Managementdiensten

Auf der Grundlage der konstruierten Domänenmodelle und darauf aufbauend der Betreiberprozessmodelle können im weiteren Verlauf die Managementbasisdienste sowie die Managementprozessdienste entworfen werden. Diese stellen letztlich den Ausgangspunkt dar, um automatisierbare Betreiberprozesse auf Basis bestehender Managementwerkzeuge umzusetzen. Der Entwurf der Managementdienste bildet demnach den Lösungsbeitrag zu den identifizierten Problemstellungen, eine auf den fachlichen Konzepten ausgerichtete Softwarearchitektur in der Domäne IT-Management zu beschreiben.

Neben den in der vorliegenden Arbeit betrachteten Modellelementen der SoaML unterstützen weitere Modellelemente die Spezifikation von Managementdiensten [Am10, De10]. Beispielsweise kann auf Basis identifizierter Dienstkandidaten bereits eine erste Übersicht über die Kollaboration der möglichen Dienste erstellt und durch die Definition einer Dienstarchitektur (*ServicesArchitecture*) festgelegt werden. Hierdurch besteht die Möglichkeit, bereits frühzeitig in der Analyse einen ersten Überblick über komplexere Szenarien zu erarbeiten, bei denen verschiedene weiter zu spezifizierende Dienste beteiligt sind. Weiterhin können auf der Grundlage definierter Dienstschnittstellen und den modellierten Dienstarchitekturen vertragliche Zusicherungen an den strukturellen Aufbau in Form von Dienstverträgen (*ServiceContract*) festgelegt werden.

Weitergehende Modellierungsvorgehen werden im Folgenden nicht weiter betrachtet, da er Fokus der vorliegenden Arbeit auf dem Entwurf wiederverwendbarer Softwareartefakte liegt. Daher werden die hierzu erforderlichen Elemente Dienstkandidat (in SoaML: `Capability`) und (abstrakte) Dienstschnittstelle (in SoaML: `ServiceInterface`) fokussiert betrachtet. Es werden die

weitergehenden Möglichkeiten bei der Gestaltung der übergeordneten Geschäftsarchitektur jedoch ausschnittsweise angedeutet.

5.3.1 Identifikation von Managementdienstkandidaten

Die Ableitung von Dienstkandidaten hat zum Ziel, die identifizierten und in den entsprechenden Domänenmodellen erfassten fachfunktionalen Anforderungen und angebotenen Werkzeugfähigkeiten in den Entwurf von Managementdiensten zu überführen, die hin bezüglich r Eigenschaften der Betreiberprozessorientierung und Wiederverwendbarkeit ausgelegt sind.

Konzeptionell wird zwischen den Anforderungen auf der einen und den konkreten, spezifizierten Schnittstellen auf der anderen Seite zunächst ein dediziertes Konzept – dass der Dienstkandidaten – eingeführt. Dies hat zum Ziel, das gesamte Entwicklungsvorgehen insgesamt nachvollziehbarer zu gestalten, da hierdurch von konkreten, zu implementierenden Diensten abstrahiert werden kann. Dienstkandidaten können einer iterativen Überarbeitung unterzogen werden, um sich ändernde Anforderungen auf der fachlichen Ebene einzuarbeiten. Außerdem kann durch dieses Konzept eine nachträgliche Integration bestehender Managementwerkzeuge erfolgen, ohne dass deswegen bereits spezifizierte Dienstschnittstellen angepasst werden müssen.

Modellierung in SoaML

Managementdienstkandidaten können in den SoaML-Modellen mittels des Elementes `Capability` dargestellt und festgelegt werden. Die Definition der identifizierten Dienstnachrichten erfolgt durch die Angabe von `MessageType`-Elementen als stereotypisierte UML-*DataTypes* [OMG-UML-Super].

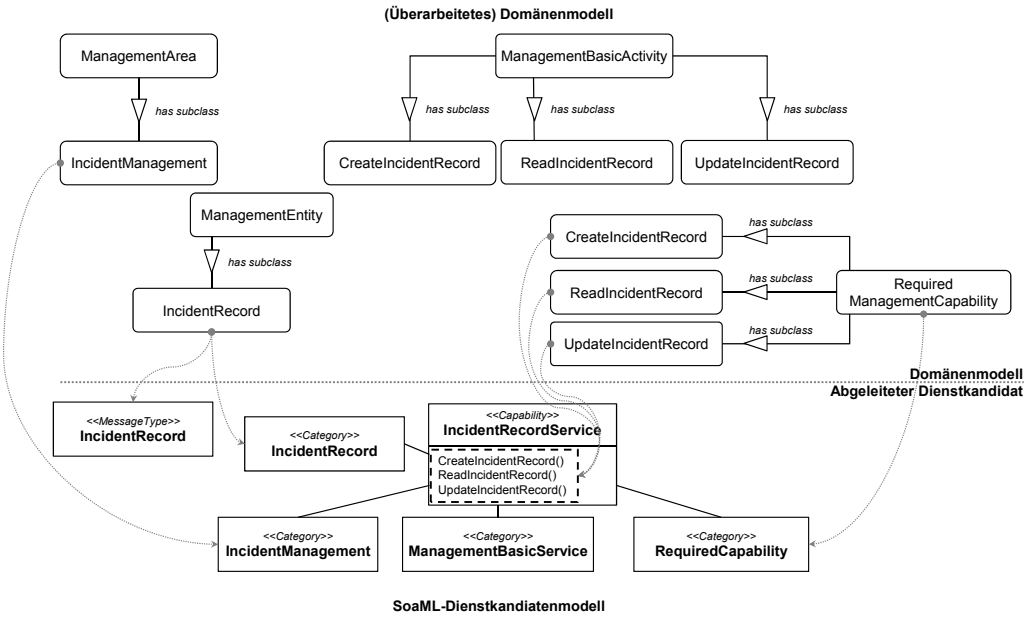

Abbildung 41 Ableiten eines Managementbasisdienstkandidaten

In Abbildung 41 ist dargestellt, wie auf Basis eines Ausschnittes der Ontologie für das *Incident Management* ein zugehöriger Managementbasisdienstkandidat abgeleitet werden kann. Die Überführung umfasst mehrere Schritte und geht über die eigentliche Modellierung der Dienstkandidaten in Form von SoaML-`Capabilities` hinaus. Da die Bewertung der Wiederverwendbarkeit einen integralen Bestandteil der vorgestellten Methode umfasst, wird der modellierte Dienstkandidat um verschiedene Aspekte erweitert. Beispielsweise erfolgt eine Erweiterung des Dienstkandidatenmodells um die in Abschnitt 4.2.4 (Seite 89 ff.) spezifizierten Kataloge, um eine Einordnung des Dienstkandidaten in entsprechende Kategorien zu ermöglichen.

Regeln für die Identifikation von Managementdienstkandidaten

Die einzelnen verschiedenen Schritte zur Ableitung von Managementdienstkandidaten kann durch die Festlegung einheitlicher Regeln zielführend unterstützt werden [PP+10, Pa10]. Da die Zielsetzung der vorliegenden Arbeit darin besteht, die wesentlichen Konzepte für eine dienstorientierte Integration bestehender Managementwerkzeuge zu erarbeiten und im Kontext eines durchgehenden Softwareentwicklungsvorgehens zu diskutieren, wird zunächst keine Spezifikation konkreter Modelltransformationen angestrengt. Diese können jedoch auf Basis der vorgestellten Regeln umgesetzt werden. Beispielsweise kann durch den Einsatz des im *Ontology Definiton Metamodel* (ODM) der OMG [OMG-ODM] zur Spezifikation der konkreten Syntax der genutzten Ontologie in Form eines UML-Profils eine durchgehende UML-metamodellbasierte Modell-zu-Modell-Transformation von Domänenmodell auf SoaML-basiertes Dienstkandidatenmodell vorgenommen werden. Hier kann beispielsweise der Einsatz von QVT-Transformationen [No09] eine entsprechende Unterstützung für Entwicklungswerkzeuge ermöglichen.

(R1) Modellierung von Managemententitäten

> Managemententitäten werden als SoaML-`MessageType` modelliert und erhalten Datenfelder anhand der entsprechenden Entitätsstrukturrichtlinien.

(R2) Definition von weiteren Managementbasisdienstkandidaten

> Für alle Managementfähigkeiten, wird ein Managementbasisdienstkandidat für jede zugeordnete Managementaktivität definiert und als `[ManagementActivity]Service` bezeichnet.

(R3) Definition von Managementbasisdienstkandidaten für Managemententitäten

> Für alle Managementfähigkeiten, die eine Managemententität betreffen, wird jeweils ein Managementbasisdienstkandidat definiert und als `[Entity]Service` benannt. Ein Managementbasisdienstkandidat, der auf Managemententitäten operiert, wird mit CRU-Operationen (*Create*, *Read* und *Update*) ausgestattet, da diese Operationen auf allen in der ISO/IEC 20000 aufgeführten Entitäten benötigt werden. Eine Löschoperation wird nicht definiert, da in der Standarddefinition explizit gefordert wird, Entitäten nicht zu löschen. Die Operationen werden als `Create[Entity]`, `Read[Entity]` und `Update[Entity]`

bezeichnet. Ausnahmen von dieser Regel sind im Domänenmodell als Entitätsstrukturrichtlinie modellierbar.

(R4) Modellierung von Managementdienstkandidaten

Jeder Managementdienstkandidat wird als SoaML-`Capability` im Managementdienstkandidatenmodell mit gleichem Namen modelliert. Die möglichen Dienstoperationen der Dienstkandidaten ergeben sich aus den zugeordneten Managementfähigkeiten und werden als Operationen der stereotypisierte UML-Metaklasse `Class` definiert.

(R5) Definition von Managementprozessdienstkandidaten für zusammengesetzte Managementaktivitäten

Zu allen Managementfähigkeiten, die eine zusammengesetzte Managementaktivität abstrahieren, werden Managementprozessdienstkandidaten definiert und als `[ManagementCapability]Service` bezeichnet.

(R6) Definition von Managementprozessdienstkandidaten

Es wird genau ein Managementprozessdienstkandidat definiert, der zur Ausführung des betrachteten Managementprozesses benötigt und als `[ManagementArea]Service` bezeichnet wird. Es wird eine Dienstoperation `Process[ManagementArea]Request` definiert.

Insgesamt entsteht durch den Verzicht auf verbindliche Modelltransformationen die Schwierigkeit, dass die Semantik der einzelnen Modellelemente beim Einsatz des hier vorgestellten Entwicklungsvorgehens bekannt sein muss. Prinzipiell kann dies zunächst als Schwachpunkt der Methode angesehen werden, da durch den Einsatz von Modelltransformationen – die eine hinreichend vollständige Spezifikation der statischen Semantik beinhalten müssen – eine automatisierte Überführung der fachlichen Modelle in Dienstkandidatenmodelle stattfinden kann. Dies gilt ebenso für die erstellten Werkzeugmodelle. Gleichwohl wird durch die Angabe grundlegender einzuhaltender Regeln und den Verzicht auf automatisierte Erzeugung von Dienstkandidatenartefakten die Möglichkeit eröffnet, frühzeitig im Entwicklungsvorgehen bestimmte Zielsetzungen bezüglich eines qualitätsorientierten Entwurfes von Managementdiensten einzubeziehen und so gezielt verschiedene Handlungsalternativen zu unterstützen. Dies ist vor allem vor dem Hintergrund der Bewertung der Wiederverwendbarkeit relevant, da die betrachteten Aspekte teilweise zu unterschiedlichen Ausprägungen von Dienstkandidatenmodellen führen. Es sind demnach zusätzliche Schritte zu betrachten, um bezüglich weiterer Vorgaben (z. B. wirtschaftliche Vorgaben) die Konstruktion von Dienstkandidatenmodellen zu unterstützen.

Muster für die Identifikation von Managemententitätsbasisdiensten

Managementbasisdienste bieten der Definition in Kapitel 4 folgend (siehe Taxonomie der Domäne, Seite 80 ff.) grundlegende Managementfunktionalität an deren Dienstschnittstelle an. Diese

angebotene Managementfunktionalität realisiert benötigte Managementfähigkeiten und ist generell unabhängig von einem spezifischen Prozesskontext. Der Zugriff auf Managemententitäten stellt für die Ausführung automatisierbarer Betreiberprozesse eine herausragende Rolle dar, da durch die Manipulation der Managemententität durch den entsprechenden Managementbasisdienst die Dokumentation im Fortschritt eines Betreiberprozesses erst ermöglicht wird. Für die Identifikation von Managementbasisdienstkandidaten können hierzu zwei Entwurfsmuster angegeben werden, da durch die Zuordnung von Managementfähigkeiten zur jeweiligen Managemententität eine eindeutige Ableitung möglich ist (siehe Abbildung 42).

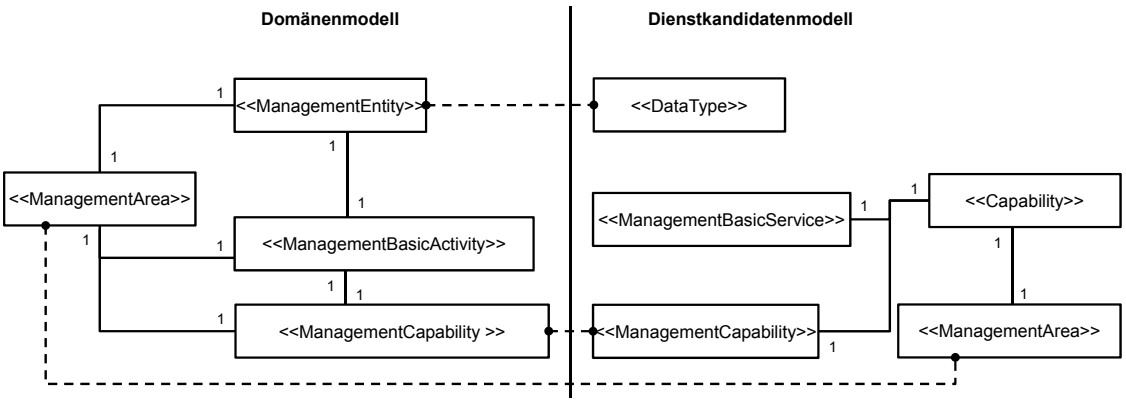

Abbildung 42 Muster für die Identifikation von Managementbasisdienstkandidaten

Hierbei ist es unerheblich, ob eine benötigte Managementfähigkeit oder eine angebotene Managementfähigkeit modelliert wird, da eine Ergänzung des Dienstkandidatenmodells zwecks Bewertung der Wiederverwendbarkeit auf Basis der in Kapitel 4 identifizierten und vorgestellten Klassifikationsmöglichkeiten vorgenommen wird.

Das in Abbildung 42 dargestellte Muster ermöglicht die Identifikation von Managementbasisdienstkandidaten, die verschiedene Managementfähigkeiten bezüglich identifizierter Managementaktivitäten für den Zugriff auf Managemententitäten umsetzen. Durch die Unterscheidung beim Einsatz des Musters der Managementfähigkeiten an der jeweiligen Konkretisierung im Domänenmodell wird eine Konkretisierung auf Basis der im Katalog `ManagementCapabilityType` festgelegten Kategorien möglich. Nachfolgende tabellarische Übersicht listet die einzelnen Elemente des Musters auf und beschreibt die jeweilige Semantik bei der Anwendung zur Identifikation von Basisdienstkandidaten.

Domänenmodellelement	Dienstkandidatenmodellelement
Management Area	Element aus dem Katalog `ManagementArea`
Management Entity	`DataType`
Management Basic Activity	- (Keine direkte Entsprechung)
Management Capability - *Required Management Capability* - *Provided Management Capability*	Element aus dem Katalog `ManagementCapability`: - `RequiredCapability` - `ProvidedCapability`
- (keine direkte Entsprechung)	`Capability`
- (keine direkte Entsprechung)	`ManagementBasicService` (Element aus dem Katalog `ManagementServiceType`)

Tabelle 17 Abbildung von Domänenmodellelementen auf Dienstkandidatenmodellelementen

Es wird ersichtlich, dass bei einigen Modellelementen (`ManagementActivity` im Domänenmodell bzw. `Capability` und `ManagementBasicService` im Dienstkandidatenmodell) keine direkte Entsprechung in den jeweils gegensätzlichen Modellen existiert.

Die Anwendung des vorgestellten Musters für die Identifikation von Managementbasisdienstkandidaten mit einem Bezug zu einer Managemententität erleichtert die Ermittlung entsprechender Dienstkandidaten aus den Domänenmodellen. Durch die Unabhängigkeit bei der Anwendung auf benötigte oder angebotene Managementfähigkeiten kann dieses Muster sowohl bei der Identifikation von Managementbasisdienstkandidaten in den fachlichen Modellen als auch bei der Identifikation von Managementbasisdienstkandidaten in den Werkzeugmodellen eingesetzt werden.

Überarbeiten

Die Identifikation von Dienstkandidaten auf Basis der erstellen Domänenmodelle sowie Betreiberprozessmodellen hat zum Ziel, die angebotenen bzw. benötigten Managementfähigkeiten zu fachlich-kohärenten Diensten zu bündeln. Hierbei können verschiedene Situationen auftreten, bei denen eine Überarbeitung der entstehenden Dienstkandidatenmodelle erforderlich wird. Wird beispielsweise bereits während der Analysephase ein bestehendes Managementwerkzeug betrachtet, kann die Überarbeitung von angebotener und benötigter Managementfähigkeit die Integration bereits auf Ebene der Dienstkandidaten unterstützen. Weiterhin kann es erforderlich sein, eine Überarbeitung

der Modelle durchzuführen, um den identifizierten Aspekten der Wiederverwendbarkeit zu genügen, indem beispielsweise fehlende Klassifikationen ergänzt, Dienstoperationen verschoben werden, um Kontextunabhängigkeit oder Disjunktheit zu wahren oder Dienstoperationen zu ergänzen, und um Vollständigkeit sicherzustellen. Auch kann die nachträgliche Überarbeitung erforderlich werden, um die Einhaltung verschiedener Namenskonventionen gemäß den vorgestellten Regeln zu sichern.

Integration auf Dienstkandidatenebene

Die Konstruktion fachlicher Modelle sowie die Konstruktion von Werkzeugmodellen entsprechen den in Abschnitt 5.1 beschriebenen Aspekten des vorwärts-orientierten bzw. dem rückwärts-orientierten Entwurfes. Im einfachen Fall (es existieren keine bestehenden Werkzeuge) entfällt die Konstruktion der Werkzeugmodelle und ein vollständiger vorwärts-orientierter Entwurf inklusive einer Implementierung kann erfolgen.

Wird die dienstorientierte Integration bestehender Managementwerkzeuge betrachtet, führt die Konstruktion von fachlichen Modellen und von Werkzeugmodellen zu verschiedenen Dienstkandidaten für Managementbasisdienste. Um frühzeitig eine Überdeckung von angebotenen und benötigten Managementfähigkeiten ermitteln zu können, ist daher die Überarbeitung der entstandenen Modelle für Managementbasisdienstkandidaten erforderlich. Abbildung 43 stellt diesen Sachverhalt an einem einfachen Ausschnitt aus der Analyse von Anforderungen für den Entwurf von Managementdiensten für das Incident Management dar.

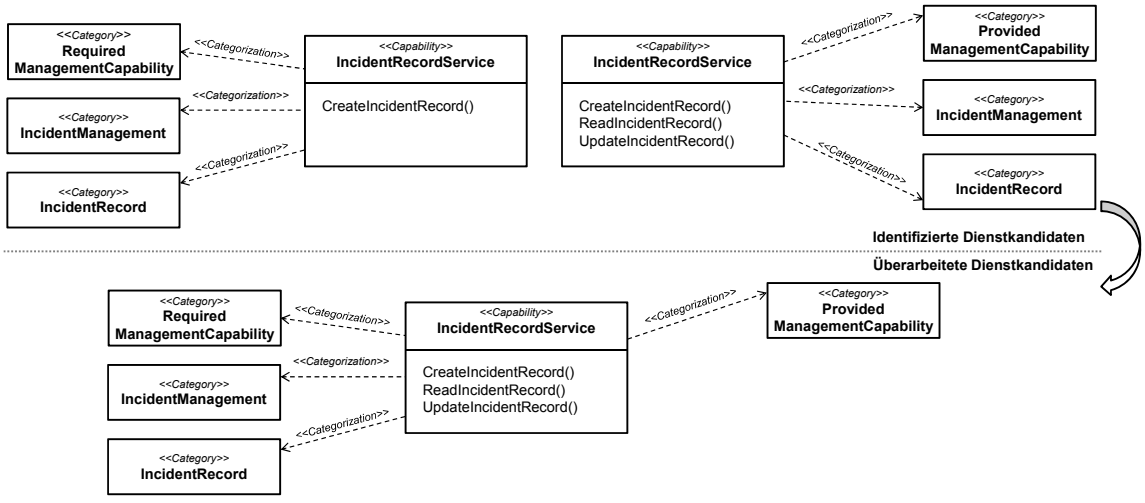

Abbildung 43 Identifikation eines Managementdienstes zur Integration

Die Überarbeitung bei der Identifikation von Dienstkandidaten hat demnach das Ziel, mögliche Integrationspunkte in den durch die Analyse bestehender Managementwerkzeuge ermittelten angebotenen Managementfähigkeiten zu erfassen. Hierbei werden die zur Integration einsetzbaren möglichen Managementdienste in der Überarbeitung durch sowohl eine angebotene

Managementfähigkeit (`ProvidedManagementCapability`) als auch durch eine benötige Managementfähigkeit (`RequiredManagementCapability`) ausgezeichnet.

Wiederverwendbarkeit

Die Überarbeitung der erstellen Dienstkandidaten bezüglich der Aspekte der Wiederverwendbarkeit gliedert sich in die betrachteten Kriterien (Klassifikation, Vollständigkeit, Disjunktheit). Da diese vier Kriterien sich teilweise gegenseitig beeinflussen, kann die mehrfach-iterative Überarbeitung der erstellten Dienstkandidaten erforderlich werden.

Überarbeiten der Klassifikation

Die Überarbeitung der Klassifikation hat zum Ziel, gemäß dem identifizierten Grad an klassifizierten Elementen die fehlenden Klassifikationen an die modellierten Dienstkandidaten anzubringen. Die Anwendung der Berechnungsvorschrift aus Abschnitt 4.3.1 (Seite 108 ff.) zeigt auf, inwiefern eine Überarbeitung erforderlich ist sowie welche Klassifikationselemente hinzugefügt werden müssen.

Die Aggregation von zusammengesetzten Komponenten aus einfacheren Komponenten bedingt auch die Übernahme aller Klassifikationen der einfachen Komponenten durch die zusammengesetzten Komponenten [RH06]. Da Managementprozessdienste später in der Implementierung als Komposition von Managementbasisdiensten realisiert werden, können Managementprozessdienste als logische Komponente aufgefasst werden, die die einfachen Komponenten der Managementbasisdienste aggregieren. Somit übernehmen Managementprozessdienste auch die Klassifikation der zugrunde liegenden Managementbasisdienste. Um die Konsistenz zwischen den Analysemodellen und den Entwurfsmodellen zu wahren und gegebenenfalls bereits frühzeitig bestimmte fachliche Abwägungen in der dienstorientierten Analyse zu beachten, wird bei der Identifikation und Definition von Managementprozessdienstkandidaten eine Überarbeitung der Klassifikationselemente auf der Grundlage der definierten Abhängigkeiten zu den identifizierten Managementbasisdienstkandidaten vorgenommen. Hierbei werden die Klassifikationselemente – wie in nachfolgender Tabelle 18 dargestellt – überführt und als Klassifikation dem jeweiligen Managementprozessdienstkandidaten zugeordnet.

Klassifikationselement	Bedeutung für den Managementprozessdienstkandidaten
Management Area	Der Dienstkandidat wird in mehreren Managementbereichen benötigt.
Management Entity	Der Dienstkandidat operiert indirekt auf einer Managemententität.

Tabelle 18 Zusätzliche Klassifikationselemente bei Managementprozessdienstkandidaten

Überarbeiten der Vollständigkeit

Die Betrachtung der identifizierten Dienstkandidaten bezogen auf den Aspekt der Vollständigkeit führt eventuell zu einer Ergänzung der bislang ermittelten Operationen bei den möglichen Dienstkandidaten.

Überarbeiten der Disjunktheit

Wird festgestellt, dass sich identifizierte Dienste in ihrem funktionalen Kontext oder in der konkreten Ausprägung von festgelegten Dienstoperationen überschneiden, werden diese in der Überarbeitung der Dienstkandidaten aufgeteilt und zu den jeweils passenden Dienstkandidaten verschoben. Falls noch kein passender Dienstkandidat existiert, wird ein neuer Dienstkandidat eingeführt und definiert.

5.3.2 Definition von abstrakten Dienstschnittstellen

Die Spezifikation von abstrakten Dienstschnittstellen bildet die Grundlage für die weitere Implementierung und stellt somit die Schnittstelle zwischen der Entwurfsphase und der Implementierungsphase im Softwareentwicklungsprozess dar. In diesem Schritt wird demnach die Abbildung der unterschiedlichen identifizierten Managementdienstkandidaten betrachtet mit dem klaren Ziel, implementierbare Softwareartefakte realisieren zu können. Neben der eigentlichen Definition für die Implementierung verbindlicher Schnittstellen wird in diesem Schritt identifiziertes Verhalten in Form von Protokollen dargestellt, sowie die Spezifikation von Dienstnachrichten erarbeitet. Da eine Überarbeitung von identifizierten Dienstkandidaten zu diesem Zeitpunkt im Entwicklungsprozess bereits erfolgt ist, kann eine einfache und direkte Abbildung der identifizierten Dienstkandidaten auf Dienstschnittstellen durchgeführt werden.

Modellierung in SoaML

Für die Modellierung von spezifizierten abstrakten Dienstschnittstellen wird das SoaML-Modellelement `ServiceInterface` eingesetzt [Am10, De10]. Die Standardspezifikation von SoaML sieht zwei verschiedene Typen von Schnittstellen vor, die ihrer jeweiligen Semantik entsprechend für einfache bzw. zusammenarbeitende Dienstschnittstellen genutzt werden [OMG-SoaML]. Der Unterschied der einfachen Dienstschnittstelle im Gegensatz zur zusammengesetzten Dienstschnittstelle besteht in der Möglichkeit der zusammengesetzten Dienstschnittstelle, dass das Verhalten bei der Nutzung der Schnittstelle beschrieben werden kann. Diese Möglichkeit fehlt bei der einfachen Dienstschnittstelle. Das Verhalten kann durch die von der UML möglichen verschiedenen Verhaltensmodelle (z. B. Sequenzfluss- oder Aktivitätsmodell) definiert werden.

Analog zur Modellierung von benötigten Dienstkandidaten und der hieraus möglichen Definition einer Abhängigkeitsbeziehung zwischen den Dienstkandidaten kann diese Information in der Spezifikation von Dienstschnittstellen dargestellt werden. Hierbei können die einzelnen benötigten Schnittstellen als Elemente (UML-*Parts*)der betrachteten Dienstschnittstelle definiert werden. In Abbildung 44 ist die beispielhafte Modellierung zweier Dienstschnittstellen auf Basis deren identifizierten und bereits vollständig überarbeiteten Dienstkandidaten dargestellt. Der Übersicht wegen sind in dieser Abbildung

die vollständig ergänzten Klassifikationselemente nicht dargestellt. Ersichtlich wird die Beziehung einer Dienstschnittstelle zu ihrem jeweiligen Dienstkandidaten über die Darstellung einer `expose`-Relation sowie die Modellierung der Abhängigkeitsbeziehung zwischen den Dienstkandidaten durch die Darstellung einer `use`-Relation. Die modellierten Dienstschnittstellen werden jeweils durch die Angabe von einfachen `Interface`-Definitionen weiter spezifiziert, da hier die eigentlichen zur Verfügung gestellten Dienstoperationen festgelegt werden.

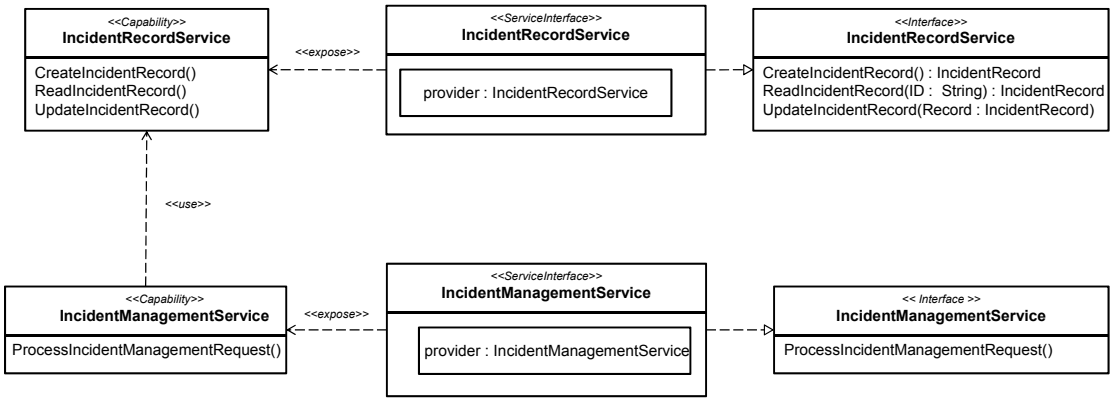

Abbildung 44 Modellierte Dienstschnittstelle und dazu gehörende technische Schnittstelle

Bei der Definition einer Dienstschnittstelle können die verschiedenen Rollen bei der Nutzung eines Dienstes festgelegt werden. Dies wird durch die Definition von `Parts` in der Definition der Dienstschnittstelle festgelegt.

Durch den Verzicht, bereits bei der Definition der abstrakten Dienstschnittstelle eine genaue Festlegung aller abhängigen weiteren Dienstschnittstellen einzugehen, sind die definierten Dienstschnittstellen generischer. Daher sind die aus den definierten Dienstschnittstellen resultierenden Dienstverträge insgesamt auch generischer, was unter dem Gesichtspunkt der betrachteten Aspekte der Wiederverwendbarkeit (insbesondere Aspekt A7) angestrebt wird. Dessen ungeachtet können die Dienstschnittstellen eines konkreten vorliegenden Dienstes aufrufende Dienste durch die Definition entsprechender `use`-Relationen dargestellt werden. Dies kann genutzt werden, um beispielsweise Callback-Schnittstellen im Fall einer asynchronen Nachrichtenkommunikation zu spezifizieren.

Durch die Trennung der verschiedenen Aspekte von den fachlich-orientierten Dienstkandidaten über die zweigeteilte Definition von Dienstschnittstellen wird die Einsatzmöglichkeit der SoaML als Modellierungssprache für den domänengetriebenen Entwurf von Managementdiensten deutlich. Vor allem durch die Zweiteilung in ein die Dienstschnittstellendefinition umfassendes Element und ein technisch-spezifizierendes Element besteht die Möglichkeit, implementierungsnahe Details an der Dienstschnittstelle zu verbergen und die eigentlichen fachlichen Aspekte darzustellen.

Neben der Modellierung der strukturellen Dienstoperationen besteht durch die SoaML die Möglichkeit, ein gültiges Nutzungsprotokoll einer Dienstschnittstelle darzustellen. Dies kann insofern zielführend sein, da somit Einschränkungen der an einer Dienstnutzung beteiligten Teilnehmer dargestellt werden können. Beispielsweise kann so bereits zur Entwurfszeit festgelegt werden, wie auf bestimmte Nachrichten zu reagieren ist oder wie eine gültige Aufrufsequenz verschiedener

Dienstoperationen definiert wird. Insgesamt werden hierzu drei unterschiedliche Aspekte betrachtet: die technisch-orientierten angebotenen bzw. benötigten Schnittstellen (`Provided Interface`, `Required Interface`), die Spezifikation des Verhaltens (`Owned Behaviour`) sowie das eine Dienstschnittstellenspezifikation umfassende UML-Element `ServiceInterface`.

Die vorgestellten Schritte zur Spezifikation einer abstrakten Dienstschnittstelle erfolgen ebenfalls iterativ, da durch die Fortführung der Entwurfsschritte mit der Integration bestehender Managementwerkzeuge eventuell verschiedene Aspekte erst während des eigentlichen Integrationsprozesses ersichtlich werden.

Regeln für die Spezifikation von Managementdienstschnittstellen

Es können – analog zu den Regeln für die Spezifikation von Managementbasisdienstkandidaten – auch für die Spezifikation von abstrakten Schnittstellen für Managementdienste einige grundlegende Konventionen festgelegt werden, die die Überführung der identifizierten Dienstkandidaten erleichtern und somit den gesamten Ansatz nachvollziehbarer gestalten (siehe auch Anforderung 1.1 auf Seite 39 ff. hierzu) [PP+10, Pa10].

(R7) Schnittstellen von Managementdiensten

> Die Schnittstelle für einen Managementdienst wird als SoaML-`ServiceInterface` modelliert, das die Managementfähigkeit in Form der Dienstkandidatenoperationen bereitstellt (`exposes`-Relation). Die angebotene Schnittstelle wird ebenfalls als genersiches UML-`Interface` modelliert und erhält die beim Dienstkandidaten identifizierte Operation, die um entsprechende Operationssignaturen erweitert werden.

(R8) Ausgabeparameter von Create- und Read-Operationen

> Operationen mit Namen `Create[Entity]` und `Read[Entity]` erhalten einen Ausgabeparameter vom Typ des `DataType`, der die jeweilige Managemententität modelliert.

(R9) Eingabeparameter von Update-Operationen

> Operationen mit Namen `Update[Entity]` erhalten einen Eingabeparameter vom Typ des `DataType`, der die jeweilige Entität modelliert.

(R10) Eingabeparameter von Read-Operationen

> Operationen mit Namen `Read[Entity]` erhalten einen Eingabeparameter vom Typ String, der als `[Entity]ID` bezeichnet wird.

(R11) Operationen von Managementprozessdiensten

Für alle Aktivitäten, die die Fähigkeit, die ein Managementprozessdienst bereitstellt, benötigen, werden alle Unteraktivitäten (mittelbar und unmittelbar) betrachtet und auf „CRU-Fähigkeiten“ nach untersucht. Der Managementprozessdienst erhält zu jeder Aktivität, die eine solche Fähigkeit benötigt, eine Operation mit dem Namen `[ManagementActivity]`.

(R12) Eingabeparameter von Operationen von genutzten Managementbasisdiensten

Die Dienstoperationen des spezifizierten Entwurfes von Managementprozessdiensten, die einen Managementbasisdienst bezüglich einer Managemententität benutzen, erhalten als Eingabeparameter ein Objekt des definierten `MessageType` zum zentralen Geschäftsobjekt des Managementbereiches.

(R13) Ausgabeparameter von Operationen

Die identifizierten Operationen erhalten als Ausgabeparameter einen Nachrichtentyp, der alle `MessageType`-Typen enthält, die in den `ServiceInterfaces` benutzter (`use`-Relation) Managementbasisdienste vorkommen.

(R14) Eingabe-/Ausgabeparameter von Operationen

Die identifizierten Operationen, die sich auf eine *Update*-Fähigkeit eines Managementbasisdienstes beziehen, erhalten als Eingabeparameter einen Datentyp entsprechend des zu aktualisierenden Feldes im Geschäftsobjekt, wie in der Entitätsstrukturrichtlinie zu diesem Objekt festgelegt. Falls der Typ nicht über eine Entitätsstrukturrichtlinie festgelegt ist wird ein *String* verwendet. Analog hierzu wird mit identifizierten Operationen verfahren, die sich auf eine *Read*-Fähigkeit beziehen und entsprechend einen Ausgabeparameter festgelegen.

Die Spezifikation der abstrakten Dienstschnittstellen bildet die Grundlage, um die weiteren Schritte im Entwurf und in der Implementierung von Managementdiensten zu betrachten. Unmittelbar wird in der Entwurfsphase noch eine genaue Festlegung der logischen Anwendungskomponenten der zu entwerfenden Managementdienste durchgeführt, wobei explizit die zu integrierenden bestehenden Managementwerkzeuge betrachtet werden. Weiterhin werden die Interaktionsprotokolle der Managementdienste definiert, falls diese bereits bekannt sind. In der Regel leiten sich diese aus den zugrunde gelegten Modellen der Betreiberprozesse ab.

5.3.3 Definition von Dienstkomponenten

Neben der Spezifikation einer verbindlichen Dienstschnittstelle und der damit einhergehenden Festlegung verschiedener Rollen und Nutzungsprotokolle ist für die genaue Eingrenzung eines zu implementierenden Dienstes die Definition einer logischen Dienstkomponente erforderlich. Die logische Dienstkomponente stellt die Implementierung der fachlichen Anforderungen dar und definiert die Außengrenze der einen implementierten Dienst ausmachenden Logik. Hierbei wird eine *Black-*

Box-Sicht eingenommen, da die genaue Umsetzung der spezifizierten Dienstschnittstellen nicht Gegenstand der Entwurfsphase ist. Hierbei ist wichtig, dass eine logische Dienstkomponente noch keine wirkliche Softwarekomponente im eigentlichen Sinn darstellt, sondern vielmehr die Gesamtheit aller zu einem zu implementierenden Dienst gehörenden Komponenten umfasst. Da im betrachteten Entwicklungsvorgehen zunächst die Definition der erforderlichen Dienstschnittstellen erfolgt, besteht die Möglichkeit, dass mehr als eine logische Dienstkomponente die gleiche Dienstschnittstelle realisiert. Hierbei wird ersichtlich, inwiefern die Definition der Schnittstelle sowie die dazu gehörende Implementierung voneinander unabhängig sind, wodurch die eigentliche Implementierung austauschbar wird, ohne zugesicherte Funktionalität zu beinträchtigen.

Implementierungen von bestehenden Managementwerkzeugen werden in diesem Schritt als eigenständige logische Komponenten definiert und im Fall unterschiedlicher Schnittstellen (syntaktische Differenzen der Dienstoperationen) als adaptierte Komponenten modelliert. Daher kann der am Entwurf beteiligte Architekt bereits frühzeitig entsprechende Hinweise für die weitere Implementierung festlegen.

Definition logischer Dienstkomponenten

Um eine logische Dienstkomponente in einem SoaML-Modell darzustellen, stehen hierzu prinzipiell die beiden Modellelemente `Participant` sowie `Agent` zur Verfügung. Ein `Participant` ist zunächst ein Teilnehmer in einer dienstorientierten Architektur und von der Semantik der SoaML-Spezifikation her nicht notwendigerweise eine implementierte Softwarekomponente. Somit wird eine spätere Entscheidung für eine konkrete Implementierungstechnologie nicht unnötig eingeschränkt. Ein `Participant` weist eine eher statische Ausrichtung der angebotenen oder benötigten Fähigkeiten auf, die sich nicht im Laufe der Zeit ändert [OMG-SoaML]. Im Gegensatz hierzu wird durch einen `Agent` eine autonome Einheit definiert, die im strukturellen Aufbau wiederum aus mehreren Instanzen bestehen kann, dessen angebotene oder benötigte Fähigkeiten sich jedoch im Laufe der Zeit verändern können. Für die Modellierung von den zu implementierenden Managementdiensten wird daher die Modellierung logischer Dienstkomponenten durch die Definition von SoaML-`Participants` verfolgt [Ge11].

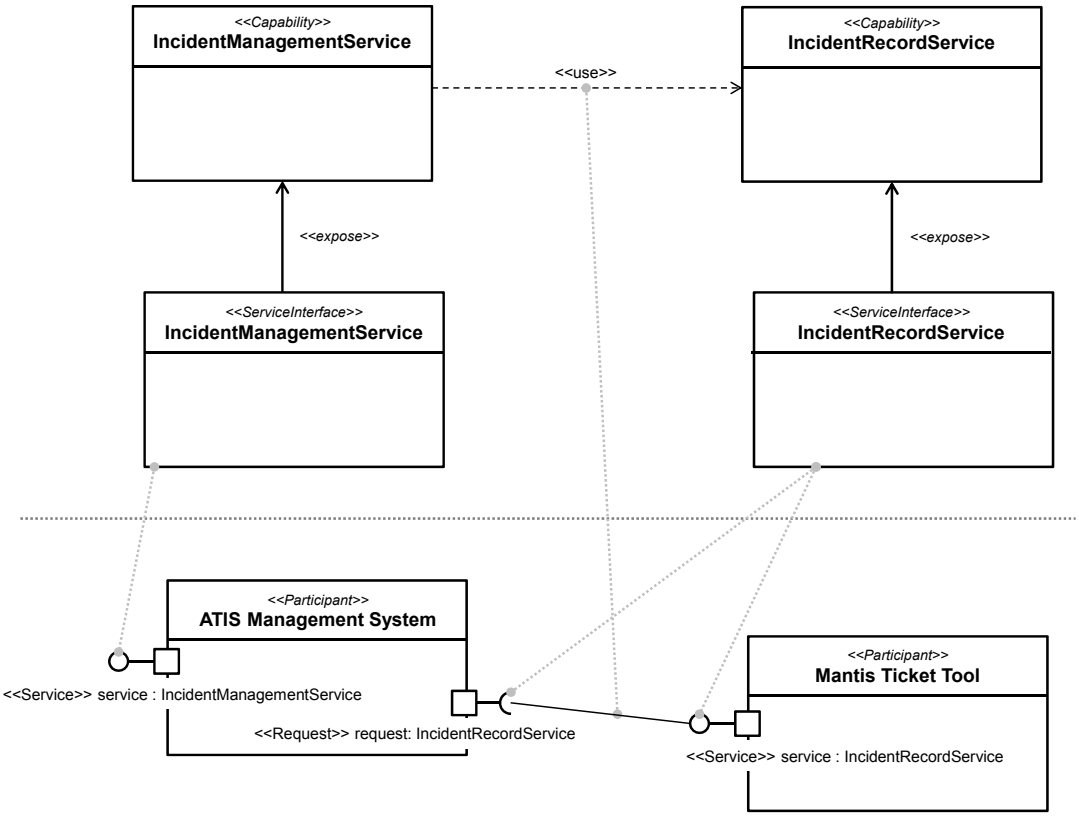

Abbildung 45 Dienstkomponenten zu den spezifizierten Dienstschnittstellen

In Abbildung 45 ist die beispielhafte Definition zweier logischer Dienstkomponenten auf Basis deren identifizierten Dienstschnittstelle dargestellt. Der Übersicht wegen wurden die Dienstoperationen der Dienstschnittstellen in der Abbildung nicht dargestellt. Die Überführung der `use`-Assoziation zwischen den Dienstkandidaten wird im Dienstkomponentenmodell durch die Verbindung zwischen benötigtem Dienst beim Teilnehmer `ATIS Management System` (`request port:IncidentRecordService`) und angebotenem Dienst beim Teilnehmer `Mantis Ticket Tool` (`service port:IncidentRecordService`) dargestellt.

Adaption bestehender Managementdienste

Da die betrachteten bestehenden Managementwerkzeuge in der Regel zwar Programmierschnittstellen besitzen, diese jedoch nicht entlang den vorgestellten Domänenmodellen ausgerichtet sind, ist die Definition spezieller Adapterkomponenten erforderlich. In der Regel nehmen diese Adapterkomponenten die Rolle eines Übersetzers ein, der eingehende Dienstnachrichten für die referenzmodellkonformen Dienstschnittstellen in Dienst- oder Methodenaufrufe transformieren, die konform zur verfügbaren Programmierschnittstelle des bestehenden Werkzeuges sind. Da hierdurch zusätzliche Anwendungskomponenten entwickelt werden müssen, ist es erforderlich, diese Details bereits im Entwurf der zu realisierenden Managementdienste zu beachten.

Die Definition der einen Managementdienst zur dienstorientierten Integration realisierender Anwendungskomponenten festlegenden Elemente wird demnach durch die Definition von Adaptern dargestellt. Hierbei übernimmt der adaptierende und aus der dienstorientierten Analyse und dem dienstorientierten Entwurf stammende Managementdienst die Übersetzerfunktion.

Die zu adaptierende Komponente (im Beispiel in nachfolgender Abbildung 46 dargestellt als bestehende Dienstkomponente `Mantis Ticket Tool`) wird als Verfeinerung im SoaML-Teilnehmermodell durch Untergliederung des übergeordneten Teilnehmers dargestellt. Somit wird durch den Architekten (als beteiligten Akteur im betrachteten Entwicklungsvorgehen) eindeutig festgelegt, dass eine Adaption der Implementierung des bestehenden Werkzeuges zur Realisierung der fachlich-orientierten Managementdienstschnittstelle vorgenommen werden soll.

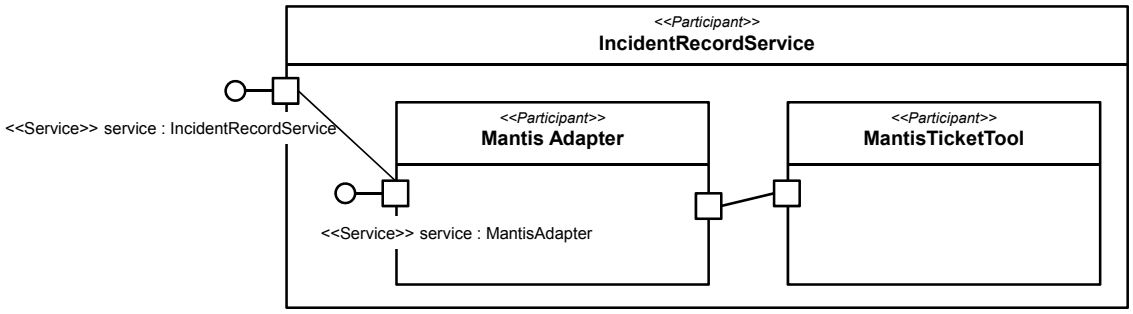

Abbildung 46 Adaption eines bestehenden Managementwerkzeuges

5.3.4 Ausblick: Spezifikation konkreter Dienstschnittstellen

Mit dem Abschluss der Spezifikation von abstrakten Dienstschnittstellen von zu implementierenden Managementdiensten ist die Grundlage gelegt, die Implementierung wiederverwendbarer und betreiberprozessorientierter Managementdienste durchzuführen. Die Verfolgung des strukturierten Entwicklungsvorgehens auf Basis des vorgestellten Metamodells der Domäne und den hieraus erstellten konformen Domänenmodellen ermöglicht die Umsetzung der fachlichen Anforderungen unter der Berücksichtigung der verschiedenen betrachteten Aspekte der Wiederverwendbarkeit und mündet in Entwurfsartefakte, die zunächst unabhängig von einer konkreten Realisierungstechnologie sind. Die Definition von abstrakten Dienstschnittstellen stellt somit den Ausgangspunkt dar, die weitere schrittweise Realisierung von Managementdiensten durch den Einsatz konkreter Implementierungstechnologien voranzutreiben.

Gegenwärtig wird für die Realisierung dienstorientierter Architekturen vor allem ein auf Webservices basierender Implementierungsansatz verfolgt. Hierbei werden konkrete Schnittstellenbeschreibungen von weiter zu implementierenden Diensten durch die Definition von Artefakten in der von der W3C standardisierten *Web Service Description Language* (WSDL) vorgenommen. Das domänengetriebene Vorgehen mit der semantischen Erweiterung der erstellen Dienstkandidaten- und Dienstschnittstellenmodelle kann durch die semantische Annotation der erstellen WSDL-Dateien mit dem ebenfalls von der W3C verantworteten *Semantic Annotations for WSDL and XML Schema* (SAWSDL) fortgeführt werden. Hierdurch können in implementierten und betriebenen

Managementdiensten Bezüge zu dem einen Dienst betreffenden Ausschnitt aus dem Domänenmodell hergestellt werden. Damit wird die spätere Integration bestehender Managementdienste vereinfacht, da die Auffindbarkeit aufgrund der klassifizierten Dienstmodelle gesichert wird.

Die Definition von konkreten Dienstschnittstellen auf Basis semantisch annotierter Webservice-Schnittstellenbeschreibungen wird am Beispiel der Umsetzung eines Managementsystems in den Tragfähigkeitsnachweisen aufgezeigt.

5.4 Zusammenfassung und erzielter Beitrag

Da der Einsatz von Informationstechnologie aus heutiger Sicht lediglich ein Mittel zum Zweck darstellt, ist es aus wirtschaftlichen Gesichtspunkten wünschenswert, sowohl kalkulierbare Kosten als auch eine zuverlässige Erbringung an der Schnittstelle zu der eine geschäftliche Aktivität unterstützenden IT-Infrastruktur vorgeben zu können. Die automatisierte Ausführung betrieblicher Abläufe beim Betreiber dieser IT-Infrastruktur rückt somit in den Vordergrund. In diesem Kapitel wird aufgezeigt, wie – ausgerichtet am im vorigen Kapitel vorgestellten Metamodell der Domäne – ein systematisches und nachvollziehbares Vorgehen gestaltet werden kann, um sowohl den Entwurf von Elementen eines Managementsystems zu unterstützen, als auch die Abbildung von Modellen dieser betrieblichen Abläufe auf das entworfene Managementsystem zu ermöglichen.

Die Spezifikation von Managementbasisdiensten bildet den Ausgangspunkt, um bestehende Werkzeuge nutzen zu können, um Betreiberprozesse zu automatisieren. Hierbei wird, ausgehend von verschiedenen Perspektiven auf die strukturellen Modelle des betrachteten Domänenausschnitts, die Ableitung von Dienstkandidaten betrachtet, die die Grundlage für die Spezifikation abstrakter Dienstschnittstellen bilden. Diese Dienstschnittstellen können im weiteren Verlauf durch logische Anwendungskomponenten implementiert werden. Hierbei können beispielsweise auch die Implementierungen bestehender Managementwerkzeuge herangezogen werden. Da durch die Einbeziehung dieser Werkzeuge in die dienstorientierte Analyse und den dienstorientierten Entwurf insgesamt eine fachliche Ausrichtung der entstehenden Werkzeugschnittstellen erreicht wird, können diese Schnittstellen direkt zur Umsetzung von automatisierbaren Betreiberprozessen genutzt werden.

Als Fortführung der bei der Spezifikation der Managementbasisdienste eingeführten Methodenschritte werden bei der Spezifikation von Managementprozessdiensten notwendige Erweiterungen am Vorgehen betrachtet, um die Erfassung dynamischer Zusammenhänge ganzheitlich betrachten zu können. Konkret wird auf Basis der erstellen Domänenmodelle ein von konkreten Prozessmodellierungssprachen unabhängiges Vorgehen aufgezeigt, um die dynamischen Zusammenhänge in Betreiberprozessen darstellen zu können. Hierdurch wird sichergestellt, dass die erstellten Domänenmodelle nicht nur den Ausgangspunkt für den Entwurf der Basisdienste bilden, sondern auch den Ausgangspunkt für den Entwurf der Prozessdienste.

Diese beiden Abschnitte bilden den Kern des vorliegenden Kapitels und konkretisieren die Methode zur Abbildung automatisierbarer Betreiberprozesse.

Insgesamt erfüllt die in diesem Kapitel vorgestellte Methode die in Kapitel 3 definierten Anforderungen zur Begegnung der identifizierten Problemstellungen. Durch die Abstraktion von konkreten Managementwerkzeugen und Betreiberprozessen kann die vorgestellte Methode flexibel

eingesetzt werden (Anforderung 2.1). Sowohl Managementbasisdienste als auch Managementprozessdienste werden auf der Grundlage einheitlicher Begrifflichkeiten eines Domänenmodells entworfen. Als ergänzender Beitrag wird aufgezeigt, inwiefern die Methode bei der Konstruktion fachlicher Referenzmodelle genutzt werden kann. Dies steigert die Nachvollziehbarkeit der einzelnen Modellelemente in realen Entwicklungsprojekten wesentlich (Anforderung 2.2). Die Modellierung betrieblicher Abläufe kann auf Basis formal spezifizierter Metamodelle erfolgen, wodurch geeignete Entwicklungswerkzeuge zum Einsatz kommen können (Anforderung 2.3).

6 Tragfähigkeitsnachweis

Die Demonstration der Tragfähigkeit des in der vorliegenden Arbeit vorgestellten Entwicklungsvorgehens zur dienstorientierten Integration von Managementwerkzeugen wird anhand der erstellten Artefakte in einem konkreten Projekt demonstriert. Hierbei wird ausgehend von Modellen des zu unterstützenden Managementbereiches auf Basis des in Kapitel 4 vorgestellten Domänenmetamodells sowie der in Kapitel 5 vorgestellten Entwurfsmethode für Managementdienste die Spezifikation von Schnittstellen für Managementdienste demonstriert. Hierbei steht die explizite Bewertung der Wiederverwendbarkeit der erstellten Dienstkandidaten zur frühzeitigen Abschätzung der weiter zu entwerfenden Dienste im Vordergrund. Die Abbildung unterschiedlicher Managementprozesse verdeutlicht die Vorteile des verfolgten Ansatzes, den Entwurf wiederverwendbarer und betreiberprozessorientierter Managementdienste durch das domänengetriebene Entwurfsvorgehen zu betrachten.

6.1 Projekt "Incident Management am KIT"

Die Abteilung Technische Infrastruktur (ATIS) am Karlsruher Institut für Technologie (KIT) ist der für die Fakultät für Informatik zuständige IT-Dienstleister. Aufgabe der ATIS ist es, von allen Forschungsinstituten gemeinsam benötige IT-Infrastrukturressourcen zu betreiben, um hierdurch den Gesamtaufwand einzelner Forschungsgruppen für den Betrieb eigener IT-basierter Dienste zu minimieren. Hierzu gehören anwendungsbasierte Dienste wie beispielswese ein E-Mail-Dienst , ein Benutzerverzeichnisdienst, verschiedene Dienste für den kooperativen Zugriff auf Datenspeicher, aber auch netz- und systembasierte Dienste wie ein zentraler und einheitlicher Ethernet-Zugang innerhalb der Standorte der Fakultät.

Die Koordination bei der Bearbeitung aufgetretener Störungen wird durch den Einsatz verschiedener Werkzeuge unterstützt. Hinzu kommt die Tatsache, dass verschiedene Fachabteilungen der ATIS unterschiedliche Werkzeuge einsetzen. Eine Integration der verschiedenen Werkzeuge ist nicht gegeben, genauso wenig wie eine einheitliche Durchführung vergleichbarer Schritte im Gesamtprozess der Störungsbearbeitung. Eine automatisierte Ausführung des Störungsbearbeitungsprozesses ist daher nur schwer möglich.

Um im Rahmen des Zusammenschlusses der ehemaligen Universität Karlsruhe (TH) und des ebenfalls sich in Karlsruhe befindlichen Forschungszentrum Karlsruhe (FZK) wachsenden Drucks, die von der ATIS bereitgestellten Dienste einem vergrößerten Nutzerkreis qualitätsgesichert zur Verfügung stellen zu können, wurde innerhalb der ATIS die Verbesserung des Störungsbearbeitungsprozesses als eine wesentliche strategische Zielsetzung identifiziert. Als ein Ergebnis der Zusammenarbeit der Forschungsgruppe *Cooperation & Management* und der ATIS wird derzeit ein Entwicklungsprojekt durchgeführt, das zur Zielsetzung hat, auf Basis der bestehenden Managementwerkzeuge zur Fehlerbehandlung eine teilautomatisierte Ausführung von Prozessen der Störungsbearbeitung innerhalb der ATIS umzusetzen. Hierzu soll eine dienstorientierte Managementplattform geschaffen werden, die für zukünftige Erweiterungen offen ist, gleichwohl eine schrittweise Integration der in der ATIS eingesetzten Managementwerkzeuge ermöglicht. Im Rahmen dieses Projektes wird zunächst die

Umsetzung eines automatisierten *Incident-Management*-Prozesses auf Basis der bestehenden Managementwerkzeuge untersucht.

6.1.1 Szenario

Eine der grundlegenden Managementfunktionen ist das Fehlermanagement (engl. *Failure Management*). Das Fehlermanagement hat zum Gegenstand, die Ursache von Fehlern schnellstmöglich einzugrenzen und zu beheben. Ein Fehler kann sowohl aufseiten eines Betreibers durch den Einsatz von Überwachungswerkzeugen erkannt werden, aber auch durch die Nichtverfügbarkeit oder Abweichung von vereinbarter Dienstqualität aufseiten eines Dienstnutzers. Ein Teilaspekt des Fehlermanagements ist die sogenannte Entstörung, also diejenigen Schritte, die zu einer unmittelbaren Behebung eines aufgetretenen Fehlers führen. Eine zielgerichtete Entstörung setzt voraus, dass der komplette Prozessfortschritt dokumentiert wird, wobei alle an der Entstörung beteiligten Mitarbeiter einen Zugriff auf für sie relevante Informationen besitzen müssen. In ISO/IEC20000-1:2005 wird mit dem *Incident-Management*-Prozess ein abgeschlossener Prozess definiert, der die schnellstmögliche Wiederherstellung vereinbarter Dienstqualitäten zum Ziel hat [ISO05] und daher als Grundlage für die Erfassung fachfunktionaler Anforderungen dienen kann. Abbildung 47 stellt die auftretenden geschäftlichen Anwendungsfälle in einem UML-Anwendungsfallmodell dar.

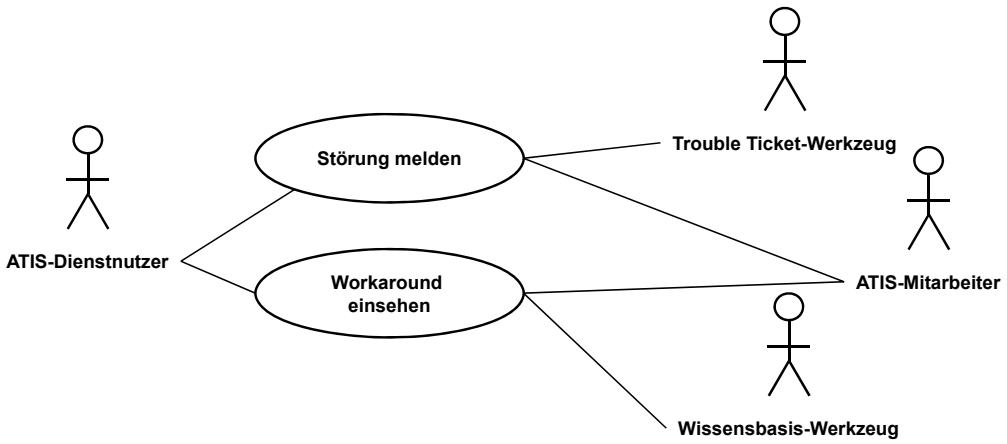

Abbildung 47 Betrachtete Anwendungsfälle im Szenario

Aus der Betrachtung der Anwendungsfälle ergibt sich gemäß [EH+08], dass zwei Geschäftsdienste erforderlich sind, um die Anforderungen der ATIS-Dienstnutzer abzudecken. Ein Geschäftsdienst beschreibt Engels et al. zufolge ein Element geschäftlichen Verhaltens, dass direkt eine Trennung einer fachlich-motivierten Außenansicht auf ein System sowie die hierzu realisierende Innenansicht ermöglicht. Im Beispiel werden die beiden Geschäftsdienste **Störung melden** (*Report Incident*) sowie **Workarounds einsehen** (*Lookup Workaround*) identifiziert und als fachlich-motivierte Funktionalität mit Außenansicht auf das zu integrierte Managementwerkzeug aufgefasst. Die betrachteten Geschäftsdienste werden durch Geschäftsprozesse umgesetzt [EH+08]. Konkret wird der

Geschäftsdienst Störung melden durch den *Incident-Management*-Prozess und der Geschäftsdienst Workaround einsehen durch den *Problem-Management*-Prozess umgesetzt. Durch eine automatisierte Ausführung zur Unterstützung der ATIS-Dienstnutzer bei der Störungsmeldung soll der Entstörungsprozess insgesamt effektiver durchgeführt werden. Die Möglichkeit, dass bekannte Vorgehensweisen zur Entstörung direkt von den ATIS-Dienstnutzern eingesehen werden können, erfolgt bislang verteilt an mehreren Stellen (Webauftritt der ATIS, direkte Nachfrage bei einzelnen ATIS-Mitarbeitern), und soll im Zuge des Szenarios in das zu entwerfende System integriert werden.

Auf der Seite des im Szenario betrachteten Dienstleisters wurden bewusst einige Vereinfachungen angenommen, die im Weiteren lediglich zur Fokussierung der Demonstration der Tragfähigkeit des vorgestellten Entwicklungsansatzes dienen. So wird vereinfachend angenommen, dass am betrachteten Szenario lediglich interne Mitarbeiter der ATIS in den betrachteten Prozessen beteiligt sind. Die Auswahl der jeweils bestehenden Werkzeuge wurde auf die Untersuchung eines in der Abteilung bereits verfügbaren *Trouble-Ticket*-Werkzeuges (mantisBT [MantisBT]) sowie ein in der Abteilung zum Einsatz kommendes Werkzeug zur Wissensverwaltung (Microsoft Sharepoint 2003 [MSSP]) gelenkt.

Trouble-Ticket-Werkzeug

Die vom bereits bestehenden *Trouble-Ticket*-Werkzeug zur Verfügung gestellte Schnittstelle wird nicht direkt als Webservice-Schnittstelle angeboten, kann aber durch einen zusätzlich verfügbaren Integrationsadapter über eine Webservice-Schnittstelle genutzt werden. Die Definition dieser Webservice-Schnittstelle erfolgt auf Basis der *Web Service Description Language* (WSDL). In nachfolgender Abbildung 48 ist der Ist-Zustand des eingesetzten *Trouble-Ticket*-Werkzeuges in Form eines UML-Komponentendiagrammes dargestellt. Der betrachtete Ausschnitt fokussiert auf die prinzipiellen Anwendungskomponenten des *Trouble-Ticket*-Werkzeuges (Komponente `Mantis Core` und Komponente `Mantis Database`) sowie eine zusätzlich bestehende Webservice-Schnittstelle mit der dazugehörigen Implementierung (Komponente `nuSOAP Server` und Komponente `Mantis Connect`). Das *Trouble-Ticket*-Werkzeug wurde bislang direkt über einen Client-seitigen Web-Browser genutzt.

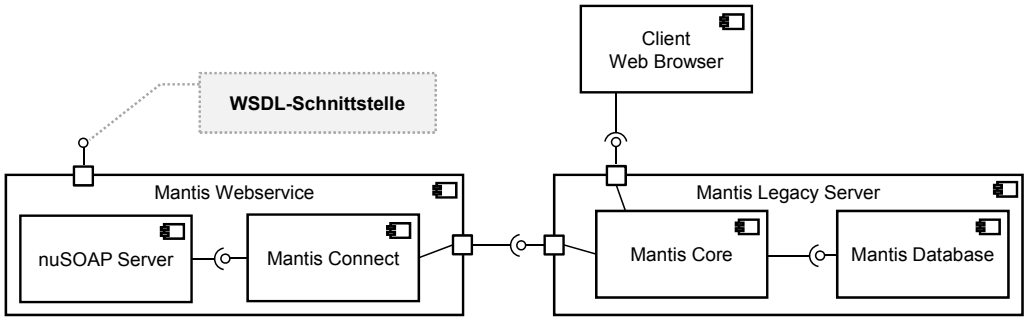

Abbildung 48 Architektur des bestehenden Trouble-Ticket-Werkzeuges

Die zur Verfügung gestellte Webservice-Schnittstelle ist nicht auf der Grundlage eines gemeinsamen Domänenverständnisses entworfen, was aus der technisch-orientierten Namensgebung der einzelnen Dienstoperationen sowie der Definition der ein-/ausgehenden Dienstnachrichten ersichtlich wird.

Am Beispiel der von der WSDL-Schnittstelle angebotenen Dienstoperation `mc_issue_add` wird dies im folgenden Quellcode-Ausschnitt deutlich.

```
<operation name="mc_issue_add">
   <documentation>Submit the specified issue details.</documentation>
   <input message="tns:mc_issue_addRequest"/>
   <output message="tns:mc_issue_addResponse"/>
 </operation>

<message name="mc_issue_addRequest">
   <part name="username" type="xsd:string"/>
   <part name="password" type="xsd:string"/>
   <part name="issue" type="tns:IssueData"/>
</message>

<message name="mc_issue_addResponse">
   <part name="return" type="xsd:integer"/>
</message>
```

Quelltext 22 Definition einer Operation in der bestehenden WSDL-Schnittstelle

Der Name der Dienstoperation (`mc_issue_add`) sowie die Definition der Dienstnachrichten sind an der Syntax der API des bestehenden Werkzeuges ausgerichtet. Somit wird die Auffindbarkeit der bestehenden Schnittstelle eingeschränkt, wenngleich eine grundlegende Dienstorientierung durch den Einsatz einer WSDL-Schnittstellendefinition vorliegt.

Wissensbasis-Werkzeug

Das in der ATIS bestehende Werkzeug zur Realisierung einer Wissensbasis (engl. *Knowledge Base*) erfüllt den Zweck, eine Unterstützung bei der langfristigen Speicherung und Dokumentation von Vorgehensweisen zu bieten, die im Falle bekannter Störungen angewendet werden können.

Zur Beurteilung der Ausgangssituation ist in nachfolgender Abbildung 49 die Architektur des bestehenden Wissensbasiswerkzeuges dargestellt.

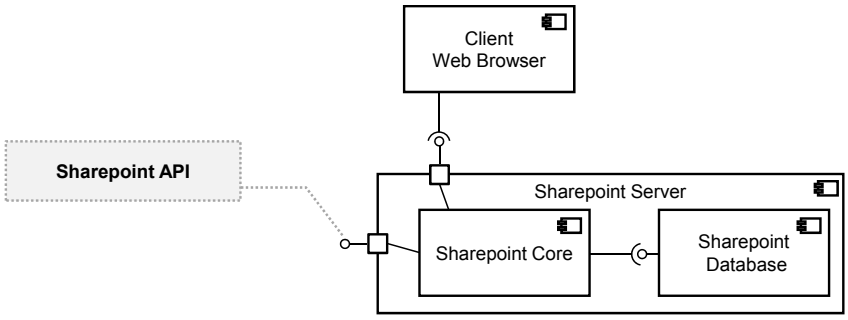

Abbildung 49 Architektur des bestehenden Wissensbasiswerkzeuges

Vergleichbar zur Ausgangssituation bei der Analyse des bestehenden *Trouble-Ticket*-Werkzeuges wird durch das Wissensbasiswerkzeug prinzipiell eine Schnittstelle zur Programmlogik zur Verfügung gestellt; diese Schnittstelle ist jedoch nicht über eine WSDL-Schnittstelle nutzbar. Ergänzend kommt bei der Integration des Wissensbasiswerkzeuges die Konstruktion einer Webservice-Schnittstelle hinzu.

```
<wsdl:operation name="UpdateListItems">
   <wsdl:input message="tns:UpdateListItemsSoapIn" />
   <wsdl:output message="tns:UpdateListItemsSoapOut" />
</wsdl:operation>

<wsdl:message name="UpdateListItemsSoapIn">
  <wsdl:part name="parameters" element="tns:UpdateListItems" />
</wsdl:message>

<wsdl:message name="UpdateListItemsSoapOut">
  <wsdl:part name="parameters" element="tns:UpdateListItemsResponse" />
</wsdl:message>
```

Quelltext 23 Auszug aus der API-Definition des Wissensbasiswerkzeuges

Es ist ersichtlich, dass sich die von der bestehenden API angebotenen Operationen nicht an den identifizierten Konzepten der Domäne IT-Management orientieren. Bei der Konstruktion eines Managementdienstes muss dies berücksichtigt werden, weshalb auch hier festgestellt werden kann, dass der Entwurf von Managementdiensten für den Managementbereich *Problem Management* auf dem vorgestellten Ansatz prinzipiell unterstützt werden kann.

Zusammenfassung

Die Analyse der im betrachteten Szenario bestehenden Managementwerkzeuge zeigt die Notwendigkeit für ein von bestehenden Managementwerkzeugen unabhängiges Vorgehen für den Entwurf betreiberprozessorientierter und wiederverwendbarer Managementdienste auf. In der Regel bieten die im Betrieb eingesetzten Managementwerkzeuge bereits verschiedene Schnittstellen an, um die durch das Werkzeug bereitgestellte Programmlogik durch den externen Zugriff zu nutzen und damit eine prinzipielle Integration zu ermöglichen. Jedoch ist, wie am Beispiel des bestehenden

Trouble-Ticket-Werkzeuges aufgezeigt, die Gestaltung dieser Schnittstellen nicht zwangsläufig betreiberprozessorientiert und wiederverwendbar.

Als Teil eines größeren Projektes zur Modernisierung der IT-Infrastruktur und in diesem Zuge auch zur Anpassung der hierfür erforderlichen Managementwerkzeuge ist daher die Umsetzung einer anpassbaren und flexibel erweiterbaren Architektur angedacht. Eine wesentliche Zielsetzung dieses Projektes ist es daher, die bestehenden Managementwerkzeuge im betrachteten Szenario dienstorientiert zu integrieren und bei der automatisierten Ausführung der betrachteten Betreiberprozesse zu nutzen.

6.1.2 Dienstorientierte Analyse

Die dienstorientierte Analyse umfasst die Definition von Domänenmodellen und den umzusetzenden Betreiberprozessmodellen und liefert als Ergebnis ein Modell identifizierter Dienstkandidaten. Diese stellen die Grundlage dar, um durch schrittweise Verfeinerung weiterer Entwurfsentscheidungen und die Hinzunahme konkreter Technologieentscheidungen eine implementierbare Spezifikation von Managementdiensten zu erarbeiten.

Domänenmodelle

Damit zu entwerfende Informationssysteme ideal bezüglich deren umzusetzender Anforderungen gestaltet werden können, ist die Gruppierung der realisierenden Komponenten des Informationssystems als Mittel zur Reduktion der Komplexität auf Basis von Domänen zielführend [EH+08]. Domänen können hierarchisch geschachtelt sein und so eine schrittweise Verfeinerung ermöglichen. Die (logischen) Komponenten eines Informationssystems sind einer Domäne zugeordnet. Im vorgestellten Ansatz wird, angelehnt an den dem Entwurf zugrunde gelegten Standard ISO/IEC20000-1:2005 [ISO05], mit dem Modellelement Managementbereich die Möglichkeit geschaffen, logisch kohärente Elemente zu gruppieren und hierdurch die Aufteilung in verschiedene Aspekte zu ermöglichen.

Die Identifikation der Managementbereiche im betrachteten Szenario ergab bereits in der grundlegenden Analyse, dass im weiteren Verlauf das *Incident Management* sowie das *Problem Management* hinsichtlich eines domänengetriebenen Entwurfes zugrunde gelegt werden. Die frühzeitige Eingrenzung auf zugrunde zu legende Managementbereiche bietet den Vorteil, dass eine klare Trennung verschiedener funktional-kohärenter Managementbereiche vorgenommen werden kann. Dies beeinflusst die Wiederverwendbarkeit günstig, beispielsweise bezüglich der betrachteten Aspekte der Vollständigkeit und der Disjunktheit. In Abbildung 50 ist der entsprechende Ausschnitt aus dem Domänenmodell in OWL-Notation [W3C-OWL] dargestellt.

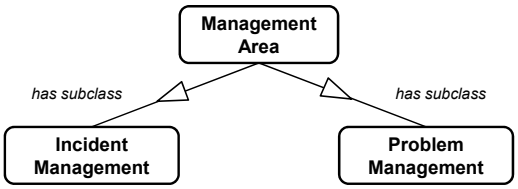

Abbildung 50 Managementbereiche im betrachteten Szenario

Ersichtlich ist die Definition von `Incident Management`- und `ProblemManagement`-Managementbereichen als Unterklassen des Elementes `Management Area`. Letztgenanntes Element entstammt dem vorgestellten Metamodell und zeigt die Möglichkeit auf, in einer OWL-Ontologie direkt einen Bezug zum jeweiligen Metamodellelement vornehmen zu können. Im weiteren Verlauf wird zunächst der Entwurf der erforderlichen Managementdienste für das *Incident Management* betrachtet. Aus dem hier betrachteten Beispiel wird ebenfalls ersichtlich, wie eine klare Trennung der beiden Managementbereiche durch die Anwendung der vorgestellten Berechnungsvorschriften ermittelt werden kann.

Die Managementteilnehmer im Szenario ergeben sich nach einer iterativen Analyse wie in nachfolgender Abbildung 51 dargestellt ist. Die Modellierung erfolgt zunächst rein auf der Grundlage der vorgelegten Anforderungsartefakte. Zur Vereinfachung sind im Folgenden die Ausschnitte aus den jeweils konkret entstandenen Entwurfsartefakten dargestellt.

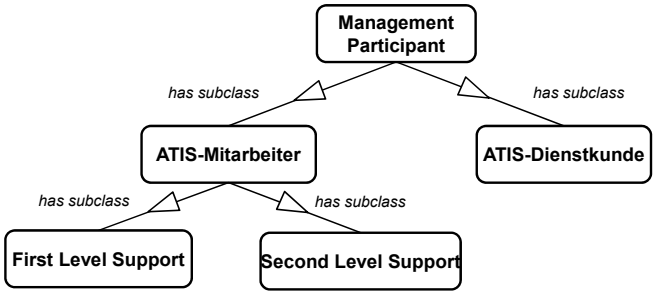

Abbildung 51 Managementteilnehmer im Szenario

Die Analyse und Modellierung der Managementteilnehmer ergibt von der Perspektive der fachfunktionalen Anforderung eine Unterteilung in die beiden Klassen ATIS-Dienstkunde und ATIS-Mitarbeiter. Diese beiden identifizierten Managementteilnehmer stellen menschliche Akteure da, die konkrete Managementaktivitäten durchführen. Die Klasse der ATIS-Mitarbeiter unterteilt sich ferner in die beiden verschiedenen Rollen, die in der Umsetzung von Störungsbearbeitungsprozessen in der Regel definiert werden: *First Level Support* und *Second Level Support*. Im *First Level Support* werden Störungsmeldungen entgegengenommen sowie generelle Anfragen an die Dienstleisterorganisation beantwortet. Ein wesentliches Ziel bei der Etablierung effizienter Betreiberprozesse ist es, die Störungsmeldungen bzw. Anfragen schnellstmöglich zu beantworten. Untersuchungen haben ergeben, dass ca. 4/5 aller beim *First Level Support* gestellten Anfragen direkt beantwortet werden können, wenn der *First Level Support* einen Zugriff auf die Information im Wissensbasis-Werkzeug besitzt. Anfragen, die nicht durch einen Mitarbeiter im *First Level Support*

beantwortet werden können, werden an einen für die Anfrage entsprechenden Spezialisten weitergeleitet (*Second Level Support*). Somit wird wiederum die Notwendigkeit für eine Integration der zur Verfügung stehenden Managementwerkzeuge entlang der zu unterstützenden Betreiberprozesse ersichtlich, wodurch die zu entwerfenden Managementdienste hinsichtlich Wiederverwendbarkeit ausgerichtet sein müssen.

Auf Basis dieser ersten vorgenommen Modellierungen der beteiligten Akteure kann eine Modellierung der Managementaktivitäten im betrachteten Szenario durchgeführt werden. In Abbildung 52 ist ein Ausschnitt aus dem erstellten ersten Domänenmodell dargestellt, das die relevanten, von einem ATIS-Mitarbeiter in der Rolle des *First Level Support* durchgeführten Managementaktivitäten aufzeigt.

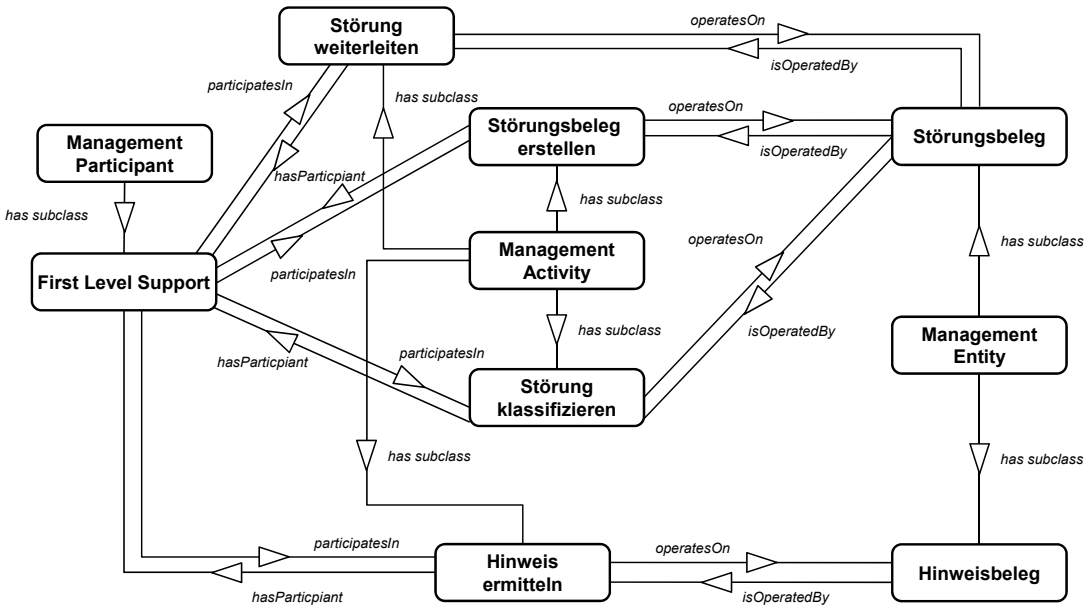

Abbildung 52 Identifizierte Managementaktivitäten und Managemententitäten

Die Elemente des erstellten Domänenmodells sind zunächst in deutscher Sprache verfasst. Es hat sich im Verlauf des Entwicklungsprojektes gezeigt, dass eine iterative Überarbeitung der erstellen Domänenmodelle erforderlich ist, wobei der Abgleich der erstellten Modelle mit den beteiligten Managementteilnehmern in einer natürlich sprachlicheren Form zielführender ist. Hierbei wird der deskriptive, also nicht einschränkende Charakter, des Einsatzes einer Ontologie während der Analysephase ersichtlich. Die Definition der Managementaktivitäten ergibt sich aus der Verfeinerung der Anwendungsfälle aus dem betrachteten Szenario. Aus Gründen der Übersichtlichkeit sind die `has subclass` Relationen zwischen den Instanzen der Metamodellelemente und dem zugrunde gelegten Managementbereich in Abbildung 52 nicht dargestellt.

Das überarbeitete Domänenmodell mit den an die vorgestellten Referenzmodelle angepassten Modellelementen ergibt sich wie in Abbildung 54 ausschnittsweise dargestellt. Hierbei sind die im ersten erstellten Modell erfassten Modellelemente, die zunächst in deutscher Sprache verfasst wurden, durch eine entsprechende Äquivalenzklassendefinition mit den in den Referenzmodellen in englischer

Sprache verfassten Modellelemente in Beziehung gebracht. Auf der Grundlage dieser ersten Übersicht werden weitere Verfeinerungen im Modell durch die Unterstützung im Ontologie-Entwicklungswerkzeug automatisiert nachgetragen. So wird beispielsweise die Definition der identifizierten Managementaktivitäten durch die Einschränkungen auf `Composed Management Activities` hinsichtlich der weiteren Analyse automatisiert durchgeführt, da die zunächst identifizierten Managementaktivitäten durch dir Äquivalenzklassendefinition automatisch den zusammengesetzten Managementaktivitäten zugeordnet werden. Abbildung 53 verdeutlicht diesen Sachverhalt.

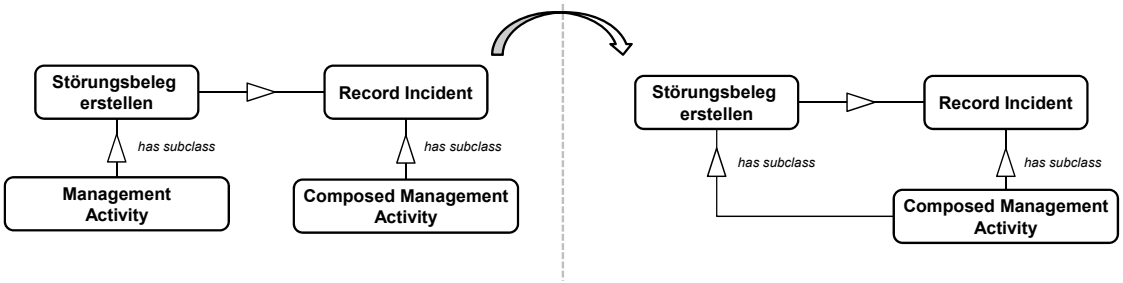

Abbildung 53 Klassifikation der identifizierten Managementaktivitäten

Das überarbeitete Modell der Domäne für den betrachteten Ausschnitt im *Incident Management* ist in Abbildung 54 dargestellt. Wie aus dieser Abbildung ersichtlich, sind zunächst die komplexen zusammengesetzten Managementaktivitäten erfasst. Diese bilden lediglich den Ausgangspunkt, um eine weitere Analyse und Modellierung zu unterstützen, die die grundlegenden Funktionalitäten identifiziert, die zum Entwurf von Managementbasisdiensten führen. Die identifizierten zusammengesetzten Managementaktivitäten werden im weiteren Verlauf durch Managementprozessdienste realisiert.

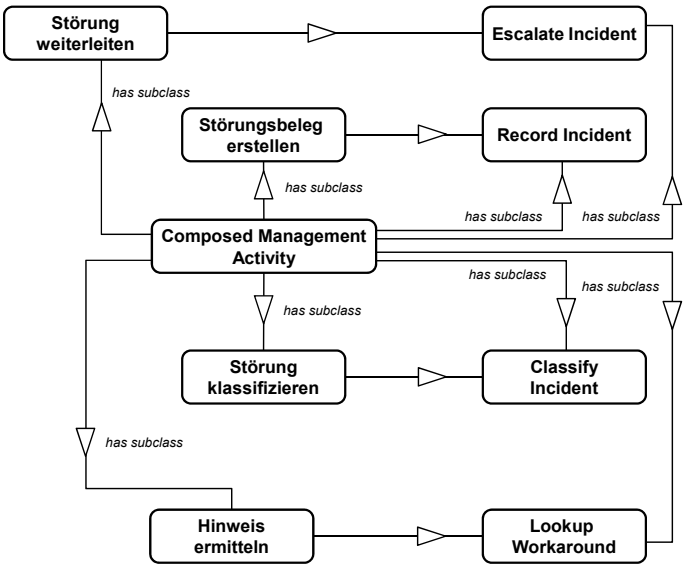

Abbildung 54 Überarbeitetes Domänenmodell mit Elementen aus dem Referenzmodell

Die identifizierten zusammengesetzten Managementaktivitäten werden durch Basismanagementaktivitäten gebildet. In der Analyse werden hierzu in den entsprechenden Modellen die zu einer zusammengesetzten Managementaktivität gehörenden Basismanagementaktivitäten definiert und den jeweiligen zusammengesetzten Managementaktivitäten zugewiesen. Hierbei wird der iterative Charakter der Domänenmodellierung ersichtlich. Im konkreten Projekt ergab sich beispielsweise während der Verfeinerung der zusammengesetzten Managementaktivitäten der Aspekt, dass zur Erstellung eines Störungsbeleges eine Authentifikation seitens der ATIS-Dienstnutzer stattfinden soll. Zusätzliche Aspekte oder erweiterte Anforderungen können durch die Modellierung der Domäne mit dem Auftraggeber des zu entwerfenden Systems daher rechtzeitig erfasst und in den Entwurf eingebracht werden.

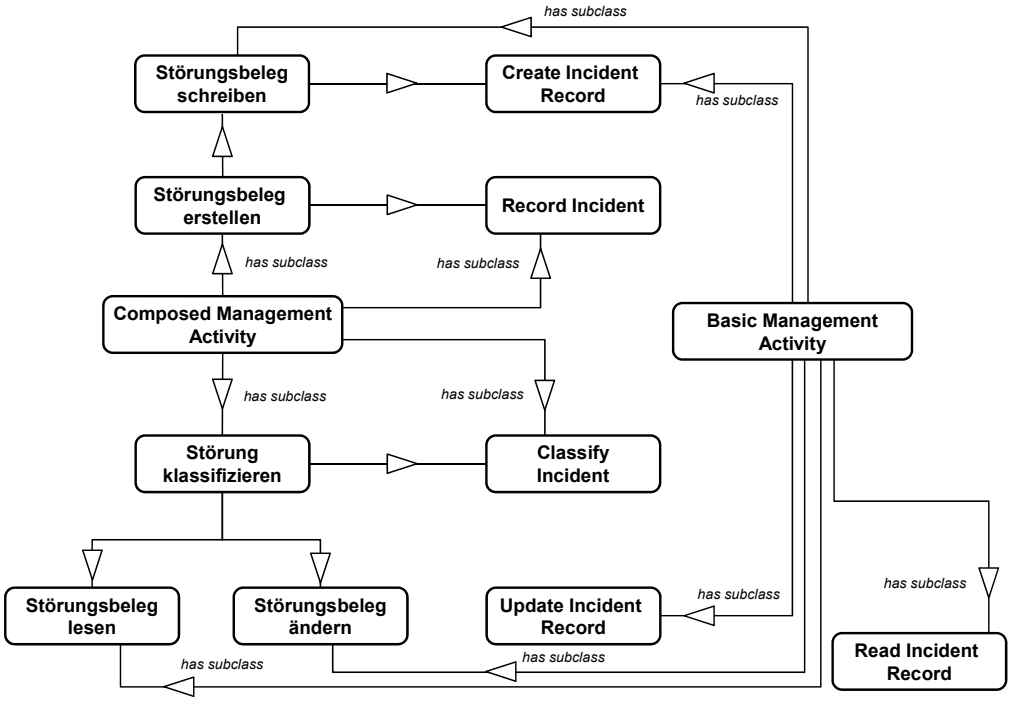

Abbildung 55 Erweitertes Modell der Managementaktivitäten

Den Abschluss der Modellierung der Domäne bildet die Definition der benötigten Managementfähigkeiten. Die explizite Betrachtung der benötigten Managementfähigkeiten bildet den Schnittpunkt zur Identifikation von Dienstkandidaten, wobei bei der Definition von benötigten Managementfähigkeiten zunächst noch eine technologieunabhängige Sicht eingenommen wird. In folgender Abbildung 56 ist ein Ausschnitt für die Definition benötigter Managementfähigkeiten zur Managementaktivität `Störungsbeleg erstellen` (`Record Incident`) dargestellt.

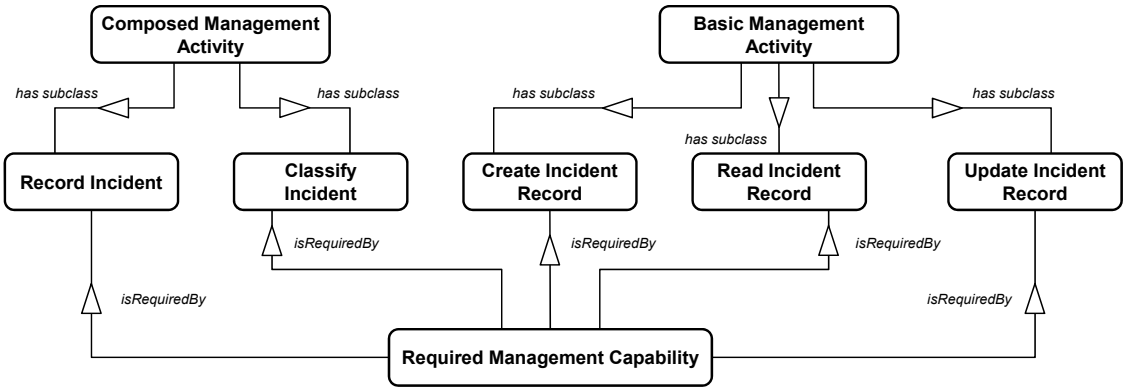

Abbildung 56 Benötige Managementfähigkeiten

Die identifizierten benötigten Managementfähigkeiten bilden im weiteren Verlauf die Grundlage, um die verschiedenen Managementdienstkandidaten als Ergebnis der dienstorientierten Analyse zu erstellen.

Der Analyse der benötigten Managementfähigkeiten steht die Untersuchung der von einem zur Integration betrachteten bestehenden Managementwerkzeug angebotenen Funktionalitäten gegenüber. Während die Definition der zur Umsetzung der fachlichen Anforderungen erforderlichen Aspekte unabhängig von den bereits bestehenden Managementwerkzeugen verläuft und somit vollständig unabhängig von einschränkenden Randbedingungen der konkreten eingesetzten Technologien der bestehenden Werkzeuge ist, wird mit der Betrachtung – und damit verbunden der Modellierung der angebotenen Managementfähigkeiten – eine von den ermittelten Anforderungen unabhängige Sicht auf die zu integrierende Werkzeuge eingenommen. Die Analyse der bestehenden Managementwerkzeuge führt schließlich über die Modellierung der angebotenen Managementfähigkeiten und der Betrachtung der verschiedenen Vorgaben im vorgestellten Entwicklungsvorgehen zunächst zu verschiedenen Managementdienstkandidaten, die letztlich die Grundlage bilden, eine Integration der tatsächlich vorhandenen mit den idealerweise benötigen Managementfähigkeiten in Deckung zu bringen (siehe auch Abschnitt 5.1 auf Seite 123). In nachfolgender Abbildung 57 ist dieser Sachverhalt dargestellt.

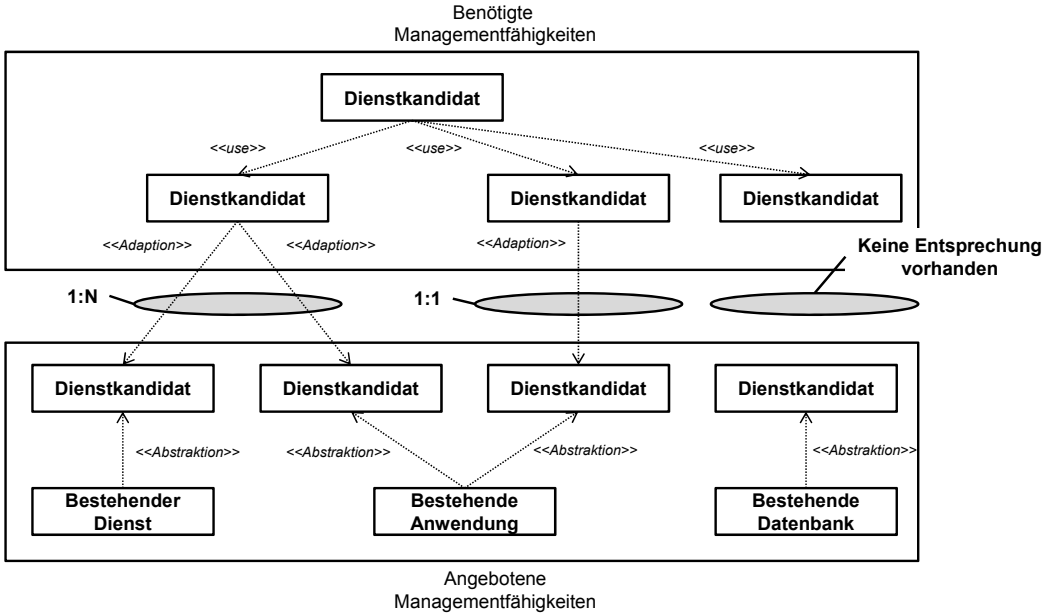

Abbildung 57 Adaption benötigter Managementfähigkeiten

Bei dem betrachteten Entwicklungsvorgehen sind letztlich drei verschiedene Varianten möglich. In Variante 1 können die identifizierten benötigten Managementfähigkeiten auf mehrere angebotene Managementfähigkeiten von bestehenden Managementdiensten, Managementwerkzeugen oder Datenbanken abgebildet werden. Die Umsetzung dieses Dienstkandidaten mündet daher in der Spezifikation eines Managementprozessdienstes, der als Komposition verschiedener weiterer Managementdienste realisiert wird. In Variante 2 kann eine benötigte Managementfähigkeit vollständig durch einen Dienstkandidaten umgesetzt werden, der im Rahmen der Definition der angebotenen Managementfähigkeiten ermittelt wurde. Die Umsetzung dieses Dienstkandidaten mündet in der Spezifikation eines Managementbasisdienstes. Eventuell wird die Konstruktion spezieller Adapter erforderlich, falls die Übersetzung von Dienstnachrichten in unterschiedliche Formate notwendig ist. In Variante 3 kann zu einer benötigten Managementfähigkeit keine Überdeckung mit den durch die Definition der angebotenen Managementfähigkeiten zur Verfügung gestellten Managementdienstkandidaten ermittelt werden. Hierdurch wird die vollständige Neuimplementierung einer realisierenden Dienstlogik erforderlich.

Angebotene Managementfähigkeiten des Trouble-Ticket-Werkzeuges

Das im Rahmen des Szenarios betrachtete bestehende *Trouble-Ticket*-Werkzeug bietet grundlegende Funktionen über eine spezielle Programmierschnittstelle (engl. *Application Programing Interface*, API) an, um auf die durch das Werkzeug angebotene Funktionalität zugreifen zu können. Wie in der Betrachtung der API bei der Beschreibung des Szenarios aufgezeigt, orientiert sich die zur Verfügung gestellte Schnittstelle jedoch nicht direkt an den zu unterstützenden Betreiberprozessen, da die definierten Funktionalitäten nicht direkt in Bezug zu den fachlichen Aspekten des betrachteten Managementbereiches gestellt werden können.

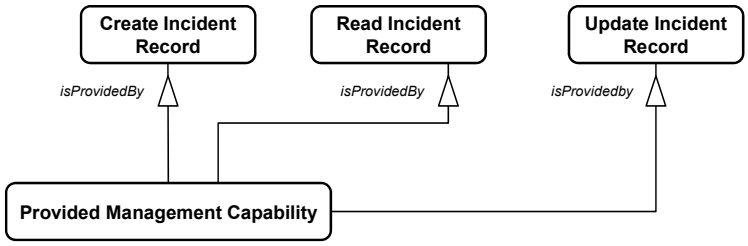

Abbildung 58 Angebotene Managementfähigkeiten des Trouble-Ticket-Werkzeuges

Aus Abbildung 58 wird demnach ersichtlich, dass durch das zu integrierende Werkzeug prinzipiell die Managementfähigkeiten `Create Incident Record`, `Read Incident Record` sowie `Update Incident Record` angeboten werden. Wenngleich die Konstruktion einer zusätzlichen Adapterkomponente im weiteren Verlauf erforderlich wird, um eine an den Domänenmodellen ausgerichtete Benennung der Dienstoperationen umzusetzen, stellt diese Information die Grundlage dar, um die weiteren Schritte beim Entwurf der Managementdienste zu unterstützen.

Angebotene Managementfähigkeiten des Wissensbasiswerkzeuges

Vergleichbar zum bestehenden *Trouble-Ticket*-Werkzeug bietet das in der ATIS eingesetzte Wissensbasiswerkzeug eine einfache Webservice-basierte API an, um die Programmlogik des Werkzeuges durch externe Aufrufe zu nutzen. Die mit dem Fachentwickler gemeinsam durchgeführte Analyse der bei der Beschreibung des betrachteten Szenarios beschriebenen Schnittstelle ergab die Definition der in nachfolgender Abbildung 59 dargestellten Managementfähigkeiten. Der Übersichtlichkeit wegen wird hier direkt die überarbeitete, in englischer Sprache verfasste Definition der Managementfähigkeiten dargestellt.

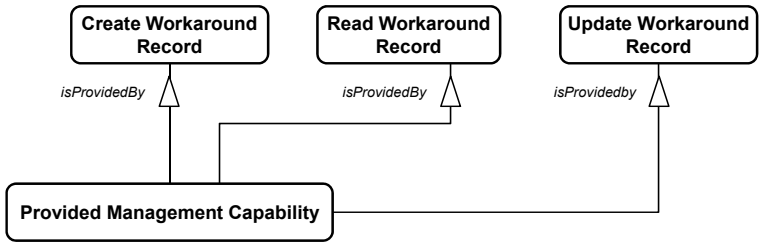

Abbildung 59 Angebotene Managementfähigkeit des Wissensbasiswerkzeuges

In Abbildung 59 sind die vom Wissensbasiswerkzeug angebotenen Managementfähigkeiten dargestellt. Wenngleich die vom betrachteten Werkzeug zur Verfügung gestellte Schnittstelle zunächst keine an den jeweiligen Domänenmodellen ausgerichtete Signatur (Methodennamen, Format der eingehenden und ausgehenden Nachrichten) aufweist, kann die vom Werkzeug angebotene Programmlogik prinzipiell genutzt werden, um die in der Definition der angebotenen Managementfähigkeiten festgelegten Managementaktivitäten zu unterstützen. Dieses Wissen ist insofern hilfreich, als dass es die weiteren Entwurfsentscheidungen bezüglich der prinzipiellen Ausgestaltung der einzelnen Managementdienste beeinflussen kann.

Das Incident-Management-Betreiberprozessmodell

Auf der Grundlage der modellierten strukturellen Zusammenhänge werden im weiteren Verlauf die dynamischen Aspekte in Form von Prozessmodellen festgelegt. Die Modellierung erfolgt, wie in Abschnitt 5.2.2 (Seite 137) beschrieben, auf Basis der erstellten Domänenmodelle. Die Überführung der am Szenario beteiligten Akteure sowie von diesen Akteuren durchgeführten Managementaktivitäten ergibt sich wie im folgenden dargestelltem Modell des *Incident-Management*-Prozesses.

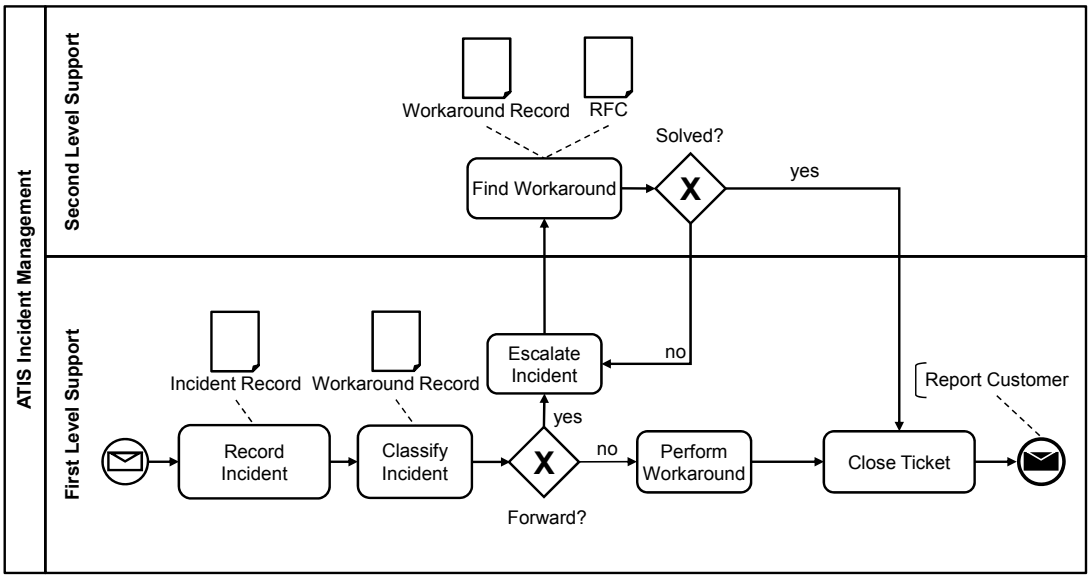

Abbildung 60 Betrachteter Incident-Management-Prozess als BPMN-Modell

Aus diesem Beispiel wird ersichtlich, dass die Definition der einzelnen Aspekte des Prozessmodells zwar auf der Grundlage der definierten Referenzmodelle erfolgt und somit ein gewisses Maß an Vergleichbarkeit über die Grenzen verschiedener Szenarien hinweg aufweisen kann, insgesamt aber durch den Bezug zu einem konkreten Szenario festgelegt wird. Die Modellierung der Betreiberprozesse hat zum Ziel, den Entwurf der einen Prozess automatisierenden Managementprozessdienste zu unterstützen. Die Möglichkeit, die erstellten Modelle bereits in der Analysephase hinsichtlich verschiedener Optimierungsmöglichkeiten hin zu untersuchen, wird im Weiteren nicht weiter betrachtet. Weiterführende Arbeiten hierzu können aber beispielsweise hierzu in [Ha09] gefunden werden.

Der im Szenario auf der Grundlage der erstellten Domänenmodelle ausgearbeitete *Incident-Management*-Prozess (Abbildung 60) bildet die Grundlage, um die Nutzungsprotokolle bei der Definition der konkreten Dienstschnittstellen ermitteln zu können, da die gezeigten Aktivitäten im BPMN-Prozessmodell die Grundlage darstellen, darauf aufbauend den Kontrollfluss und somit den Austausch einzelner Dienstnachrichten zu definieren. Wie in der Abbildung des Prozessmodells dargestellt, werden die identifizierten Managementteilnehmer im Szenario als *Lanes* im BPMN-Modell dargestellt. Die Modellierung der einzelnen Prozessaktivitäten ergibt sich wie in Abschnitt

5.2.2 beschrieben und wird nach der Überführung der einzelnen Aktivitäten durch die Definition des Kontrollflusses vervollständigt.

Dienstkandidatenmodelle

Die Domänenmodelle und die modellierten Betreiberprozesse bilden den Ausgangspunkt, um die Definition von Managementdienstkandidaten zu beschreiben.

Dienstkandidaten für benötigte Managementfähigkeiten

Die identifizierten benötigten Managementfähigkeiten werden im weiteren Verlauf gebündelt und zu Dienstkandidaten zusammengefasst. Dadurch entsteht die Möglichkeit, die resultierenden Artefakte in der Analysephase bereits bezüglich verschiedener Aspekte der Wiederverwendbarkeit hin zu untersuchen. Betrachtet man die dienstorientierte Analyse im Vergleich zu den bekannten Vorgehensmodellen im objektorientierten Softwareentwurf (beispielsweise [Ba00, BD09]), entspricht die Definition von Dienstkandidaten den logisch zusammengehörenden Anwendungskomponenten [Er08, EH+08]. Dienstkandidaten stellen demnach einen Ausgangspunkt für den Entwurf eines dienstorientierten Systems dar, das im weiteren Entwicklungsvorgehen verfeinert wird. Die Dienstkandidaten enthalten noch keine vollständig spezifizierten Signaturen von Dienstoperationen, sondern lediglich zunächst aus fachlicher Sicht benötigte Funktionalität. In den SoaML-Modellen von Dienstkandidaten werden daher auch lediglich die weiter zu entwerfenden Methodennamen definiert. Die Modellierung der einzelnen Dienstkandidaten erfolgt der besseren Übersicht wegen im weiteren Verlauf des Anwendungsbeispieles ausschnittsweise, wobei der Fokus auf die Nachvollziehbarkeit der einzelnen Schritte von der Überführung der Domänenmodelle in die Dienstkandidatenmodelle gelegt wird.

Zunächst wird die Definition der Managementdienstkandidaten der benötigten Managementfähigkeiten betrachtet, die sich aus den erstellten Domänenmodellen für das *Incident Management* ergeben.

In Abbildung 61 ist die Definition der Managementdienstkandidaten dargestellt, die sich aus der Betrachtung des Ausschnitts des in Abbildung 56 dargestellten Teils des Domänenmodells ergeben. Hierbei wird die Anwendung des in Kapitel 5 vorgestellten regelbasierten Entwicklungsvorgehens ersichtlich (Regeln (R1) bis (R6) aus Abschnitt 5.3.1, Seite 145 ff.). Die dargestellten Dienstkandidaten sind bereits bezüglich der Klassifikation benötigter Managementfähigkeiten erweitert (im Modell als `RequiredManagementCapability` dargestellt).

Der identifizierte Dienstkandidat `IncidentManagementService` wird wegen Regel R6 festgelegt und ergibt sich aus der Anforderung, den *Incident-Management*-Prozess (teil-) zu automatisieren. Die einzige festgelegte Operation `ProcessIncidentManagementRequest()` nimmt die im Modell des umzusetzenden Betreiberprozess eingehende Nachrichten entgegen und stellt somit einen zentralen Zugangspunkt dar, um die weiteren Schritte im *Incident-Management*-Prozess zu koordinieren.

Der Dienstkandidat `ClassificationService` definiert die benötigte Managementfähigkeit für die zusammengesetzte Managementaktivität `ClassifyIncident` (Regel R5), ebenso der

Dienstkandidat für die benötigte Managementfähigkeit zur zusammengesetzten Managementaktivität `RecordIncident`.

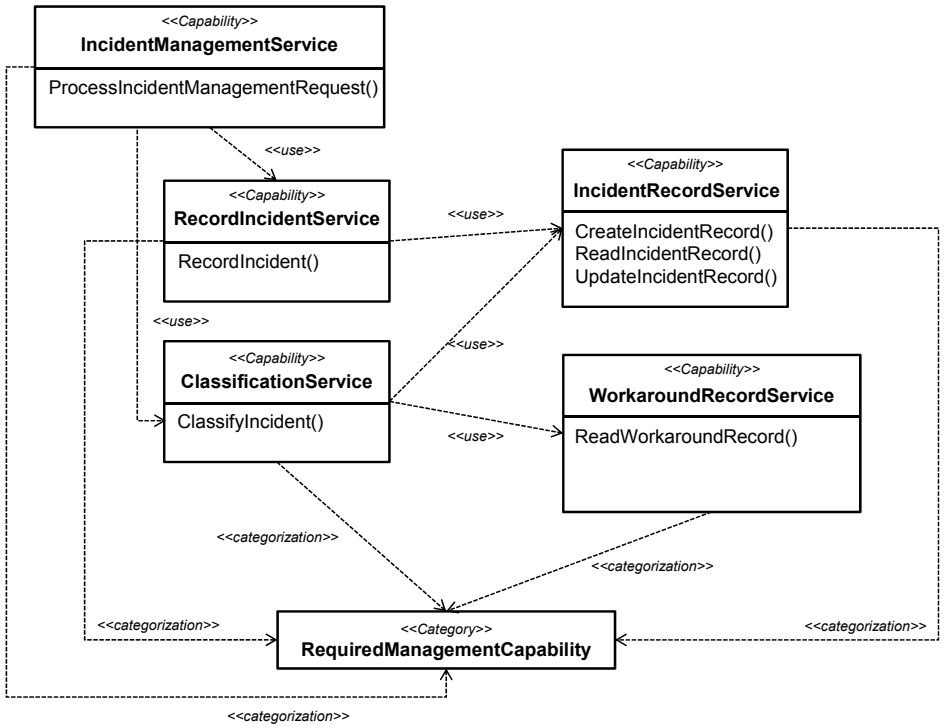

Abbildung 61 Dienstkandidatenmodell benötigter Managementfähigkeiten

Der Dienstkandidat `IncidentRecordService` ergibt sich aus der Bündelung der erforderlichen Managementfähigkeiten für die Managementaktivitäten `CreateIncidentRecord`, `ReadIncidentRecord` und `UpdateIncidentRecord`. Hierbei kommt die Regel R3 zur Anwendung, die die Definition von Managementbasisdienstkandidaten für Managemententitäten beschreibt. Der Name des entsprechenden Dienstkandidaten ergibt sich durch Verbindung der betreffenden Managemententität (in diesem Fall: `IncidentRecord`) sowie dem Suffix `Service`. Wenngleich eine vollständige Annotation der jeweiligen Elemente der Dienstkandidaten- oder Dienstentwurfsmodelle mit den in Kapitel 4 vorgestellten Katalogen durchgeführt werden soll, unterstützt die Einhaltung von Namenskonventionen die Wiederverwendung zusätzlich.

Der Dienstkandidat `WorkaroundRecordService` ergibt sich aus der benötigten Managementfähigkeit, um auf einen `WorkaroundRecord` lesend zugreifen zu müssen. Diese Managementfähigkeit wird von dem identifizierten Managementprozessdienstkandidaten für `ClassifyIncidentService` benötigt.

Die Definition der Abhängigkeiten der einzelnen Managementdienstkandidaten untereinander wird, sofern diese schon bekannt sind, gemäß dem geschilderten Vorgehen durch eine `use`-Relation

dargestellt. Die Ergänzung der einzelnen Kategorien erfolgt gemäß den zugrunde liegenden Elementen der Domänenmodelle.

Dienstkandidaten für angebotene Managementfähigkeit

Die Dienstkandidaten für die angebotenen Managementfähigkeiten der durch die bestehenden Werkzeuge zur Verfügung gestellten Programmlogik wird analog zu den Managementdienstkandidaten für die benötigten Managementfähigkeiten definiert und bezüglich der untersuchten Aspekte der Wiederverwendbarkeit betrachtet. Da Managementdienstkandidaten, die zur Integration eines bestehenden Managementwerkzeuges konstruiert werden, bereits über eine ausführbare Implementierung verfügen, werden diese Dienstkandidaten im weiteren Verlauf automatisch zu Managementbasisdiensten.

In nachfolgender Abbildung 62 ist das Ergebnis der Identifikation von Managementdienstkandidaten für das im betrachteten Szenario vorhandene *Trouble-Ticket*-Werkzeug dargestellt. Der hier dargestellte Managementdienstkandidat ist bereits vollständig bezüglich der zu beachtenden Aspekte der Wiederverwendbarkeit hin definiert worden.

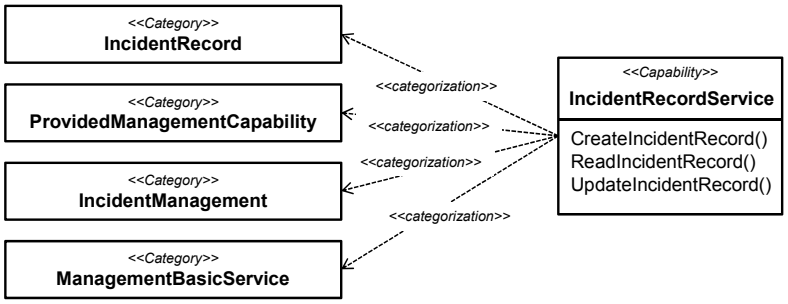

Abbildung 62 Dienstkandidatenmodell angebotener Managementfähigkeiten

Hierbei ist bereits der Vorteil ersichtlich, der durch das verfolgte domänengetriebene Vorgehen erwächst: Die Definition des Dienstkandidaten, der die bestehende Managementfunktionalität abstrahiert, also entgegen den an den Anforderungen abgeleitete Dienstkandidat ausgerichtet ist, ähnelt diesem stark. Lediglich die verschiedene Kategorisierung bezüglich der Managementfähigkeit unterscheidet diesen Dienstkandidaten von denjenigen, die auf Basis der benötigten Managementfähigkeiten identifiziert wurden. Dadurch wird die im folgenden Schritt durchgeführte Abbildung bezüglich benötigter und tatsächlich vorhandener Funktionalitäten wesentlich vereinfacht.

Überarbeitung

Die aus der dienstorientierten Analyse resultierenden Dienstkandidaten gruppieren die logischen Anwendungskomponenten der weiter zu implementierenden Managementdienste und bilden daher als Ausgangspunkt für den späteren Entwurf den Übergang zwischen dienstorientierter Analyse und dienstorientiertem Entwurf. Die Überarbeitung der bislang entstandenen Managementdienstkandidaten für das betrachtete Szenario zur Umsetzung von *Incident-Management*- und *Problem-Management*-Prozessen und der hieraus notwendigen Integration der bestehenden Managementwerkzeuge (*Trouble-*

Ticket-Werkzeug und Wissensbasiswerkzeug) führt zu einer Bewertung der Situation wie in Abbildung 57 dargestellt. Hierbei wird die Abbildung der benötigten auf die tatsächlich vorhandenen Managementdienstkandidaten erforderlich, um festzustellen, welche fehlenden Aspekte in einer iterativen Durchführung des Entwicklungsvorgehens anfallen.

Im betrachteten Beispiel wird konkret in diesem Schritt daher die Zusammenführung der Dienstkandidaten für erforderliche und benötigte Managementfähigkeiten an den beiden identifizierten Dienstkandidaten `IncidentRecordService` und `WorkaroundRecordService` sichtbar.

Bewertung der Wiederverwendbarkeit

Die Zielsetzung des verfolgten systematischen Entwicklungsvorgehens betrachtet explizit die Wiederverwendbarkeit der entstehenden Dienstkandidaten bzw. Dienstschnittstellen. Um frühzeitig im Entwicklungsvorgehen verschiedene Handlungsalternativen eingrenzen zu können, wird daher die Betrachtung der bislang definierten Managementdienstkandidaten erforderlich. Damit kann vermieden werden, dass verschiedene Entwurfsentscheidungen erst spät in der eigentlichen Entwurfsphase getroffen werden und eine Überarbeitung der bislang entstandenen Artefakte einen größeren Aufwand nach sich zieht. Da die definierten Dienstkandidaten die Grundlage bilden, einen ersten Entwurf von logisch zusammengehörenden Anwendungskomponenten der zu entwerfenden Managementdienste darzustellen, kann eine Betrachtung der Wiederverwendbarkeit bereits in der Analysephase einen positiven Einfluss auf das weitere Entwicklungsvorgehen ausüben.

An den im Anwendungsbeispiel aus Abbildung 61 dargestellten Dienstkandidaten werden im Folgenden die betrachteten zentralen Aspekte der Wiederverwendbarkeit, die in den Dienstkandidaten und Dienstentwurfsmodellen festgestellt werden können, angewendet und beispielhaft dargestellt.

Aspekt Klassifikation

Die im ausgewählten Beispiel dargestellten Dienstkandidaten sind nicht optimal bezüglich der erforderlichen Klassifikation. Da in Abbildung 61 bislang lediglich der Typ der zugrunde liegenden Managementfähigkeit als klassifizierendes Element dargestellt ist, ergibt die Anwendung der Berechnungsvorschrift zur Bewertung des Aspektes der Klassifikation bei allen betrachteten Dienstkandidaten, dass weitere Klassifikationselemente festzulegen sind.

Die Anzahl der insgesamt verfügbaren Klassifikationsmöglichkeiten ergibt für den betrachteten Entwicklungsprozess aus den definierten verschiedenen Katalogen $ToCD_{MSC}(MSC_X) = |\{MST, MAT, MCT, ME\}| = 4$, und, wie im Beispiel gezeigt, die Anzahl der tatsächlich vorhandenen Klassifikationselemente bei allen Dienstkandidaten $AoCD_{MSC}(MSC_x) = 1$.

Somit ergeben sich die Wertigkeiten der einzelnen betrachteten Dienstkandidaten wie folgt:

$$RoCD_{MSC}(RecordIncidentService) = \frac{AoCD_{MSC}(RecordIncidentService)}{ToCD_{MSC}} = \frac{1}{4}$$

$$RoCD_{MSC}(ClassificationService) = \frac{AoCD_{MSC}(ClassificationService)}{ToCD_{MSC}} = \frac{1}{4}$$

$$RoCD_{MSC}(IncidentRecordService) = \frac{AoCD_{MSC}(IncidentRecordService)}{ToCD_{MSC}} = \frac{1}{4}$$

$$RoCD_{MSC}(WorkaroundRecordService) = \frac{AoCD_{MSC}(WorkaroundRecordService)}{ToCD_{MSC}} = \frac{1}{4}$$

$$RoCESC(IncidentManagementService) = \frac{AoCD_{MSC}(IncidentManagementService)}{ToCD_{MSC}} = \frac{1}{4}$$

Da für alle Dienstkandidaten ein Wert kleiner 1 berechnet wird, ist ersichtlich, dass alle hier beispielhaft dargestellten Dienstkandidaten keine vollständige Klassifikation aufweisen. Nachfolgende Abbildung 63 zeigt einen Ausschnitt der beiden angepassten Dienstkandidaten `ClassificationService` und `WorkaroundService` mit der erweiterten Klassifikation. In Abbildung 52 sind die entsprechenden erweiterten Dienstkandidaten `RecordIncidentService` sowie `IncidentRecordService` dargestellt.

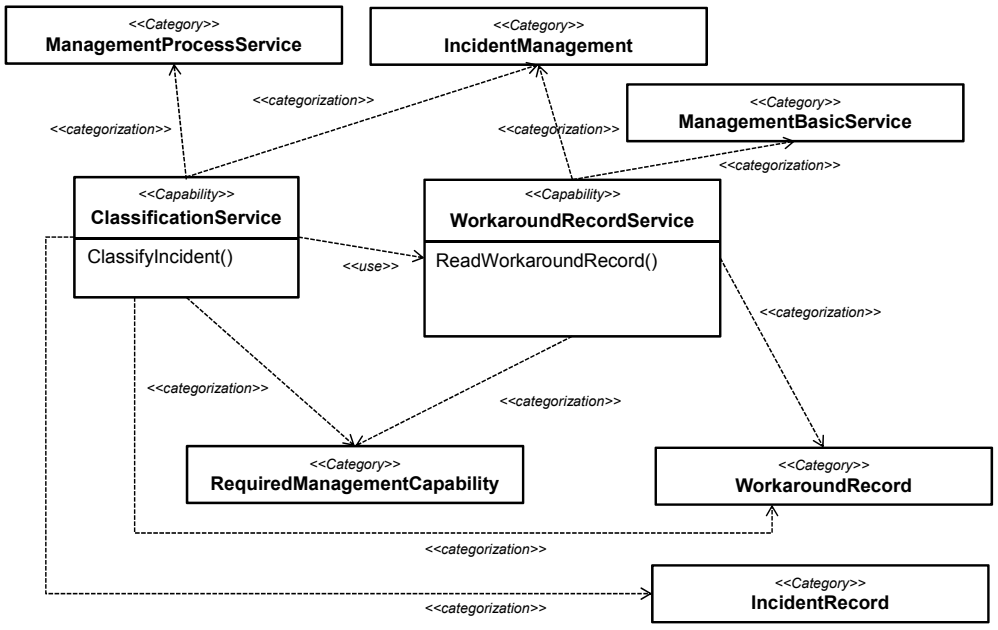

Abbildung 63 Erweiterte Dienstkandidaten mit vollständiger Klassifikation

Abbildung 64 zeigt die Dienstkandidaten für den `RecordIncidentService` sowie für den `IncidentRecordService` nach der Überarbeitung.

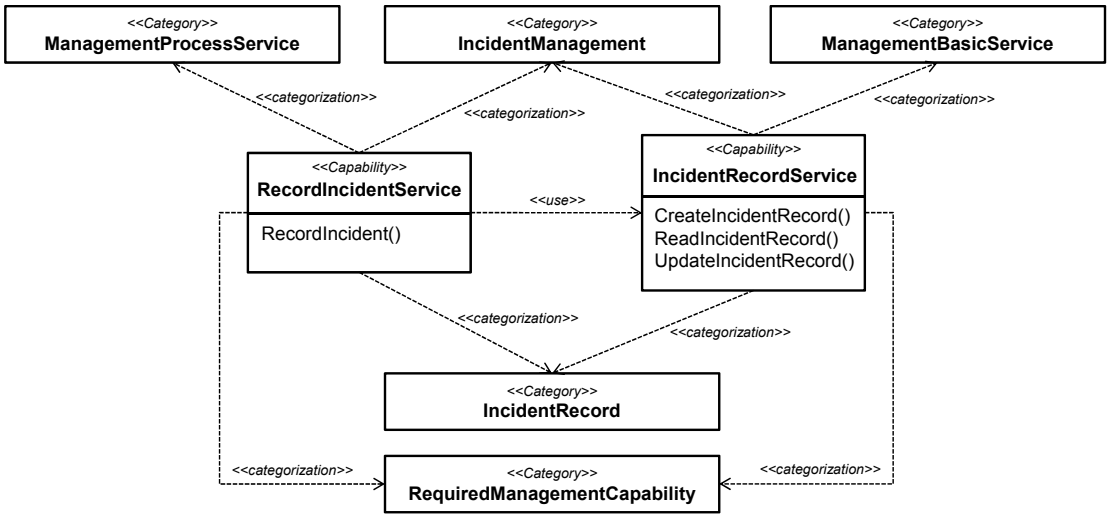

Abbildung 64 Erweiterte Dienstkandidaten mit vollständiger Klassifikation (Fortsetzung)

Die Überarbeitung der zunächst definierten Dienstkandidaten zu den überarbeitenen Kandiaten `IncidentRecordService`, `WorkaroundRecordService`, `ClassifyIncidentService` und `RecordIncidentService` führt insgesamt zu einem gemäß dem betrachteten Aspekt der Klassifikation vollständig erweiterten Analysemodell der zu entwerfenden Managementdienste.

Die Überarbeitung des Dienstkandidaten `IncidentManagementService`, der die grundlegende Umsetzung des zu unterstützenden Betreiberprozesses darstellt, führt nach der Erweiterung der fehlenden Klassifikationselemente jedoch zu einem etwas differenzierten Bild. Da dieser Managementdienstkandidat durch die zugrunde gelegte Komposition aus den Managementdiensten `RecordIncidentService` und `ClassificationService` indirekt auf zwei Managemententitäten operiert, führt die vollständige Klassifikation in diesem Fall zu einem Wert größer 1. Hierbei wird der eigentliche Grund für den betrachteten Aspekt der vollständigen Klassifikation zur Bewertung der Wiederverwendbarkeit sichtbar. Durch die vollständige Klassifikation aller Analysemodelle (und im weiteren Verlauf ebenso aller Entwurfsmodelle) wird die Betrachtung der weiterführenden Aspekte zur Abschätzung der Wiederverwendbarkeit überhaupt erst möglich.

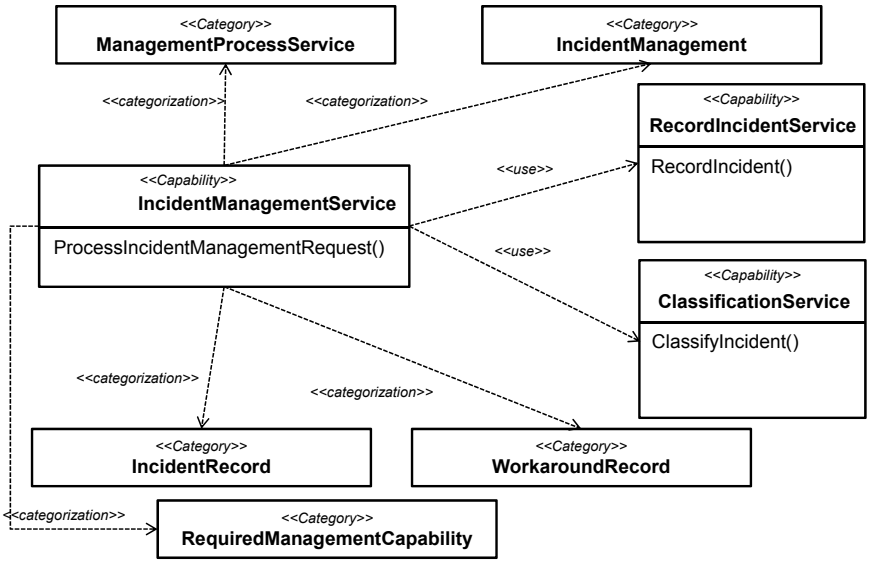

Abbildung 65 Dienstkandidat IncidentManagementService mit vollständiger Klassifikation

Die vollständige Einordnung der erstellten Entwurfsartefakte in das dem betrachteten Entwicklungsprozess zugrunde liegende Klassifikationsschema ermöglicht die Reflexion der bislang getätigten Modellierungsschritte – und hiermit auch verbunden der bislang getätigten Entscheidungen bezüglich einer ersten groben Gruppierung von logisch zueinander gehörenden Managementdiensten – und unterstützt somit direkt die Etablierung eines gemeinsamen Verständnisses zwischen Analysten und Architekten. Da durch die Klassifikationselemente im Wesentlichen ein direkter Bezug zu den entsprechenden Elementen der der Analyse zugrunde liegenden Domänenmodelle hergestellt wird, stellt eine vollständige Klassifikation die Grundlage dar, um die weiteren Aspekte zu betrachten. Hierbei wird ersichtlich, dass dieser Aspekt keinen direkten Einfluss auf die Wiederverwendbarkeit der entstehenden Artefakte ausübt, sondern lediglich indirekt zu beachten ist, da eine Verbindung zu den weiteren betrachteten Aspekten besteht.

Aspekt Vollständigkeit

Die im vorliegenden beispielhaft untersuchten Dienstkandidatenmodell vorhandenen datenzentrierten Managementbasisdienstkandidaten (`IncidentRecordService` und `WorkaroundRecord-Service`) ergeben bei der Anwendung der Berechnungsvorschrift zur Bestimmung der Vollständigkeit $RoC_{MSC}(MSC_x)$ mit $MSC_1 = IncidentRecordService$ und $MSC_2 = WorkaroundRecordService$ folgende Ergebnisse:

$$RoC_{MSC}(MSC_1) = \frac{EX(CO, MSC_1) + EX(RO, MSC_1) + EX(UO, MSC_1)}{3} = \frac{3}{3} = 1$$

sowie für den `WorkaroundRecordService`:

$$RoC_{MSC}(MSC_2) = \frac{EX(CO, MSC_2) + EX(RO, MSC_2) + EX(UO, MSC_2)}{3} = \frac{1}{3}$$

Der entworfene Dienstkandidat für den Basiszugriff auf die Managemententität `IncidentRecord` weist demnach eine für einen datenzentrierten Managementdienst vollständige Definition von weiter zu entwerfenden Dienstoperationen auf. Beim `WorkaroundRecordService` ist dies noch nicht der Fall. Aus diesem Beispiel wird ersichtlich, inwiefern sich nachträgliche Analyseentscheidungen auf das weitere Vorgehen im dienstorientierten Entwurf auswirken können.

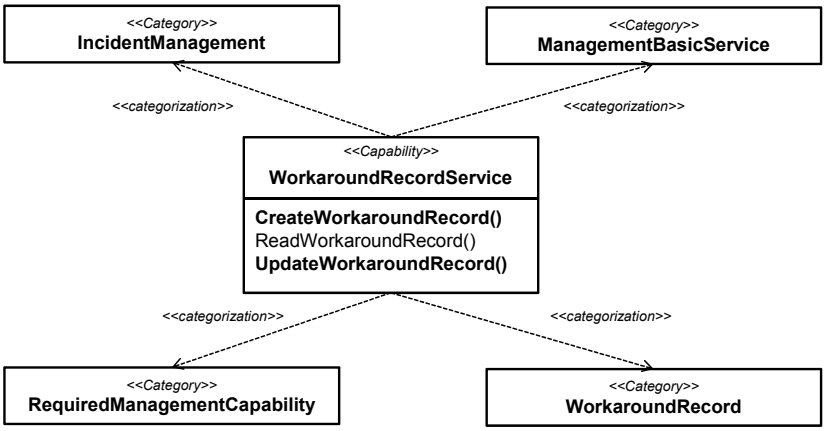

Abbildung 66 Vollständig ergänzter Managementbasisdienstkandidat WorkaroundRecordService

Abbildung 66 zeigt den überarbeiteten und bezüglich des Aspektes der Vollständigkeit ergänzten Managementbasisdienstkandidaten für den `WorkaroundRecordService`. Die beiden ergänzten Operationen (`CreateWorkaroundRecord` und `UpdateWorkaroundRecord`) sind jeweils hervorgehoben.

Aspekt Disjunktheit

Die weitere Analyse der im betrachteten Beispiel dargestellten Managementdienstkandidaten ergibt für den Aspekt der Disjunktheit ein differenziertes Bild.

Disjunktheit des funktionalen Kontextes

Um die Anwendung der Berechnungsvorschriften nachvollziehen zu können, werden zunächst die einzelnen bislang vollständig ergänzten Managementdienstkandidaten betrachtet und wie in nachfolgender Tabelle zusammengefasst zugrunde gelegt.

Dienstkandidat	Klassifikationselemente

`IncidentRecordService`	`IncidentManagement`
	`ManagementBasicService`
	`RequiredManagementCapability`

	ProvidedManagementCapability IncidentRecord
WorkaroundRecordService	ProblemManagement ManagementBasicService RequiredManagementCapability ProvidedManagementCapability WorkaroundRecord
RecordIncidentService	IncidentManagement ManagementProcessService RequiredManagementCapability IncidentRecord
ClassificationService	IncidentManagement ManagementProcessService RequiredManagementCapability IncidentRecord WorkaroundRecord
IncidentManagementService	IncidentManagement ManagementProcessService RequiredManagementCapability IncidentRecord WorkaroundRecord

Tabelle 19 Vollständige Klassifikation der Dienstkandidaten

Auf Basis dieser vollständigen Klassifikationen können die funktionalen Überdeckungen berechnet werden. Der Übersicht halber ist die Angabe der Berechnungsvorschriften im Anschluss an eine zunächst zusammenfassende Tabelle mit den einzelnen Werten dargestellt. An dieser Stelle sei angemerkt, dass die vorgestellten Ergebnisse auf der in Kapitel 4 vorgestellten Klassifikation mit genau diesen definierten Katalogen und Klassifikationselementen beruhen. Durch die Änderung der

Kataloge (z. B. im Falle einer Ergänzung des Metamodells und der hieraus resultierenden Ergänzung der Kataloge) ergeben sich andere Werte, die jedoch relativ gesehen zueinander wieder genormt sind.

Die einzelnen Werte für die verschiedenen Dienstkandidaten ergeben sich zusammenfassend wie folgt:

Dienstkandidat	AoSFCSC(MSC)	RoSFCSC(MSC)
`IncidentRecordService`	= 17	≅ 3,54
`WorkaroundRecordService`	= 9	≅ 1,88
`ClassificationService`	= 12	= 2,5
`RecordIncidentService`	= 14	≅ 2,92
`IncidentManagementService`	= 14	≅ 2,92

Tabelle 20 Zusammenfassung der einzelnen Werte

Die bei der Berechnung der Überdeckung des funktionalen Kontextes zugrunde gelegten Berechnungsvorschriften sind für das betrachtete Szenario aus den fünf identifizierten Dienstkandidaten wie folgt konkretisiert.

Zunächst wird die Anzahl aller im untersuchten Entwurf vorliegenden Klassifikationselemente bestimmt:

$$\alpha = \frac{1}{5} * \sum_{i=0}^{i<5} AoCD_{MSC}(MSC_i) = \frac{1}{5} * (5 + 5 + 4 + 5 + 5) = \frac{24}{5}$$

Mit dem Faktor α können die Werte der absoluten Überdeckungen genormt auf die Anzahl aller vorhandenen Klassifikationselemente angegeben werden.

IncidentRecordService

Die absolute Überdeckung der Klassifikationselemente für den `IncidentRecordService` ist:

$$AoSFC_{MSC}(IncidentRecordService) = \sum_{i=0}^{i<5} \sum_{j=0}^{j<4} \sum_{k=0}^{k<AoCD_{MSC}(MSC_j)} COV(CE_i, CE_k) = 17$$

und somit die relative Überdeckung:

$$RoSFC_{MSC}(IncidentRecordService) = \frac{AoSFC_{MSC}(IncidentRecordService)}{\alpha} = \frac{85}{24} \cong 3{,}54$$

WorkaroundRecordService

Die absolute Überdeckung der Klassifikationselemente für den `WorkaroundRecordService` ist:

$$AoSFC_{MSC}(WorkaroundRecordService) = \sum_{i=0}^{i<5} \sum_{j=0}^{j<4} \sum_{k=0}^{k<AoCD_{MSC}(MSC_j)} COV(CE_i, CE_k) = 9$$

und somit die relative Überdeckung:

$$RoSFC_{MSC}(WorkaroundRecordService) = \frac{AoSFC_{MSC}(WorkaroundRecordService)}{\alpha} = \frac{45}{24} \cong 1{,}88$$

RecordIncident

Die absolute Überdeckung der Klassifikationselemente für den `RecordIncident` ist:

$$AoSFC_{MSC}(RecordIncident) = \sum_{i=0}^{i<4} \sum_{j=0}^{j<4} \sum_{k=0}^{k<AoCD_{MSC}(MSC_j)} COV(CE_i, CE_k) = 12$$

und somit die relative Überdeckung:

$$RoSFC_{MSC}(RecordIncident) = \frac{AoSFC_{MSC}(RecordIncident)}{\alpha} = \frac{60}{24} = 2{,}5$$

ClassificationService

Die absolute Überdeckung der Klassifikationselemente für den `ClassificationService` ist:

$$AoSFC_{MSC}(ClassificationService\ ) = \sum_{i=0}^{i<5} \sum_{j=0}^{j<4} \sum_{k=0}^{k<AoCD_{MSC}(MSC_j)} COV(CE_i, CE_k) = 14$$

und somit die relative Überdeckung:

$$RoSFC_{MSC}(ClassificationService\ ) = \frac{AoSFC_{MSC}(ClassificationService\ )}{\alpha} = \frac{70}{24} \cong 2{,}92$$

IncidentManagementService

Die absolute Überdeckung der Klassifikationselemente für den `IncidentManagementService` ist:

$$AoSFC_{MSC}(IncidentManagementService) = \sum_{i=0}^{i<5} \sum_{j=0}^{j<4} \sum_{k=0}^{k<AoCD_{MSC}(MSC_j)} COV(CE_i, CE_k) = 14$$

und somit die relative Überdeckung:

$$RoSFC_{MSC}(IncidentManagementService) = \frac{AoSFC_{MSC}(IncidentManagementService)}{\alpha} = \frac{70}{24} \cong 2{,}92$$

Disjunktheit von Dienstoperationen

Die im betrachteten Szenario identifizierten Dienstkandidaten weisen keine Überdeckung bezüglich der bei den Dienstkandidaten identifizierten möglichen Dienstoperationen auf. Dies kann damit begründet werden, dass im betrachteten Szenario keine bereits entworfenen Managementdienste zum Einsatz kamen, auf der deren Grundlage sich mögliche überlappende Dienstoperationen identifizieren lassen würden.

Das vorliegende Modell ist demnach optimal bezüglich der Betrachtung der Disjunktheit von Dienstoperationen.

Interpretation der Werte

Der vorgestellte Ansatz, auf Basis eines einheitlichen Metamodells der Domäne die Analyse und den Entwurf von Managementdiensten durch die Definition von Domänenmodellen zu unterstützen stellt den Ausgangspunkt dar, um die drei ausgewählten Aspekte von Wiederverwendbarkeit (Klassifikation, Vollständigkeit und Disjunktheit) direkt in den entstandenen Dienstmodellen aufzeigen zu können.

Die Interpretation der ermittelten Werte (durch die in Abschnitt 4.3.1 angegebenen Berechnungsvorschriften) ist bis auf die Ergebnisse des Aspektes Disjunktheit mehr oder weniger direkt nachvollziehbar. Da die überarbeiteten Dienstkandidaten bezüglich einer vollständigen Klassifikation die definierte Berechnungsvorschrift erfüllen, ist somit die Wiederverwendbarkeit insgesamt wahrscheinlicher, als wenn die identifizierten Managementdienstkandidaten nicht klassifiziert wurden. Dies gilt unter der Annahme, dass die vollständige Klassifikation bei der Definition von abstrakten Dienstschnittstellen mit übernommen wird. Die Erweiterung von konkreten Webservice-Schnittstellen stellt dann letztlich den Ausgangspunkt dar, um die (hier nicht weiter betrachteten) technischen Aspekte der Auffindbarkeit im Kontext der Wiederverwendung umzusetzen. Der Aspekt der Vollständigkeit ist direkt am identifizierten Managementdienstkandidaten für den `WorkaroundRecordService` nachvollziehbar. Da durch die Ergänzung von `Create`- und `Update`-Operationen eine zukünftige Erweiterung vorweggenommen wird, kann der in diesem Szenario betrachtete Dienstkandidat bereits bezüglich einer wahrscheinlichen Wiederverwendung optimiert werden.

Die ermittelten Werte hinsichtlich des Aspektes der Disjunktheit sind zunächst nicht intuitiv nachvollziehbar. Betrachtet man die unterschiedlichen Werte für die beiden Basisdienstkandidaten sowie die im Vergleich hierzu weniger gestreuten Werte für die drei Prozessdienstkandidaten, können

jedoch verschiedene Aussagen aus dem vorliegenden Dienstkandidatenmodell gezogen werden. Der relativ große Wert $RoSFC_{MSC}(IncidentRecordService)$ deutet auf eine große Überdeckung mit den weiteren Dienstkandidaten im betrachteten Szenario im *Incident Management* hin, während hierzu der relativ kleine Wert $RoSFC_{MSC}(WorkaroundRecordService)$ mit der Tatsache begründet ist, dass dieser Dienstkandidat aufgrund der Zuordnung zum Managementbereich *Problem Management* in einem dedizierten Entwicklungsvorgehen für die Betrachtung zusätzlicher Anforderungen genutzt werden wird.

An den ermittelten Werten kann relativ klar nachvollzogen werden, dass der Entwurf bezüglich optimaler Ausgangspositionen zur Wiederverwendung oftmals Gegenläufig zu weiteren erwünschten Eigenschaften ist. Wenngleich eigentlich eine vollständige Disjunktheit bezüglich des funktionalen Kontextes angestrebt wird und somit idealerweise Werte gegen 0 ermittelt werden sollten, deuten die Streuung der Werte der Prozessdienstkandidaten um Werte zwischen 2,7 und der Wert des `IncidentRecordService` von 3,54 auf abgeschlossene und bezüglich des jeweils konkreten betrachteten Szenarios kohärente Dienstkandidaten hin. Für losgelöst von einem komplexen Szenario betrachtete Dienstkandidaten sollten demnach kleine Werte, für Analysemodelle bestehend aus mehreren Dienstkandidaten jedoch große Werte ermittelt werden.

6.1.3 Dienstorientierter Entwurf

Auf Basis der zuvor definierten und bezüglich der Aspekte der Wiederverwendbarkeit überarbeiteten Dienstkandidaten wird im weiteren Verlauf die Entwurfsphase von Managementdiensten betrachtet. Hierbei steht die Spezifikation der abstrakten Dienstschnittstellen mit allen hierzu relevanten Gesichtspunkten im Vordergrund.

Um die grundlegenden Schritte im Entwicklungsvorgehen bezüglich der dienstorientierten Analysephase klar herauszustellen, wurde im weiteren Verlauf angenommen, dass die zu entwerfenden Managementdienste durch einen synchronen Aufruf genutzt werden können. Hierdurch wird der Entwurf benötigter Rückruf-Schnittstellen (engl. *Callback*-Schnittstellen) nicht erforderlich, wodurch die vorliegenden Entwurfsmodelle zunächst weniger komplex werden. Prinzipiell kann das vorgestellte Vorgehen sowie die Bewertung der Wiederverwendbarkeit jedoch auch im Entwurf asynchron-nutzbarer Dienstschnittstellen eingesetzt werden.

Abstrakte Dienstschnittstellen

Die Spezifikation der abstrakten Dienstschnittstellen erfolgt direkt auf Basis der identifizierten Dienstkandidaten. Am Beispiel des betrachteten Szenarios wird die Definition der Dienstschnittstellen zu den Dienstkandidaten `IncidentRecordService`, `RecordIncidentService`, `WorkaroundRecordService` und `ClassificationService` betrachtet. Die Definition der abstrakten Dienstschnittstellen hat zum Ziel, die wesentlichen Merkmale der zu entwerfenden Managementdienste festzulegen, jedoch noch keine Technologie-Entscheidungen vorwegzunehmen. Somit bleibt eine maximale Freiheit für die Ausgestaltung der Implementierung (bei vollständigen Neuimplementierungen) bzw. eine zielführende Festlegung der umzusetzenden Anforderungen bei Integrationsprojekten. Bei Bedarf kann die Entscheidung für die Übernahme einer bestehenden

Implementierung deswegen vollständig losgelöst von der Analyse und dem Entwurf der umzusetzenden Anforderungen betrachtet werden.

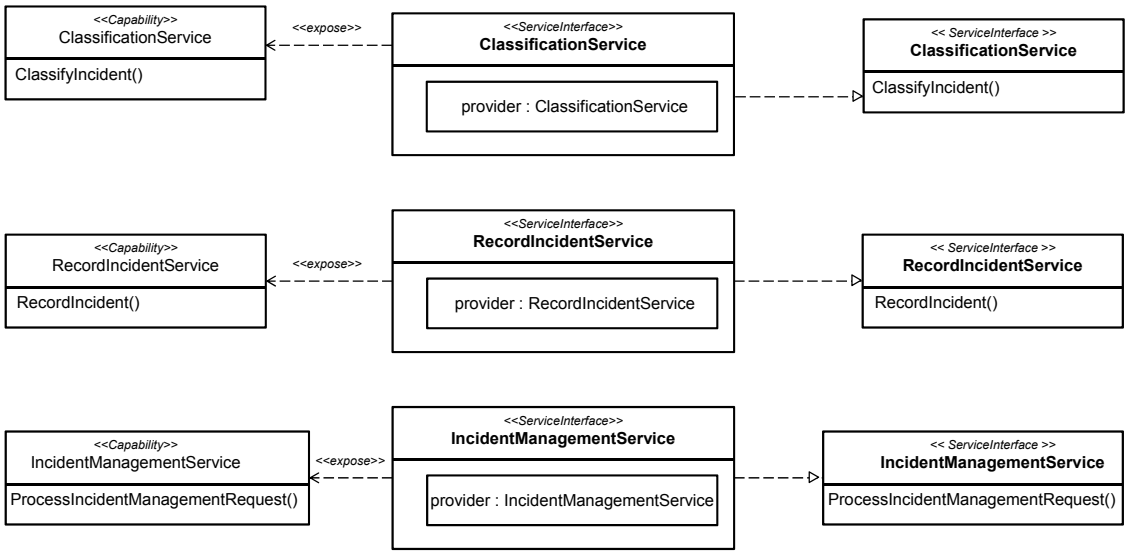

Abbildung 67 Abstrakte Dienstschnittstellen der Managementprozessdienste

Abbildung 67 zeigt der Übersicht wegen zunächst die gemäß dem Vorgehen für den domänengetriebenen Entwurf von Managementdiensten spezifizierten Dienstschnittstellen für die beiden im betrachteten Szenario identifizierten Managementprozessdienstkandidaten.

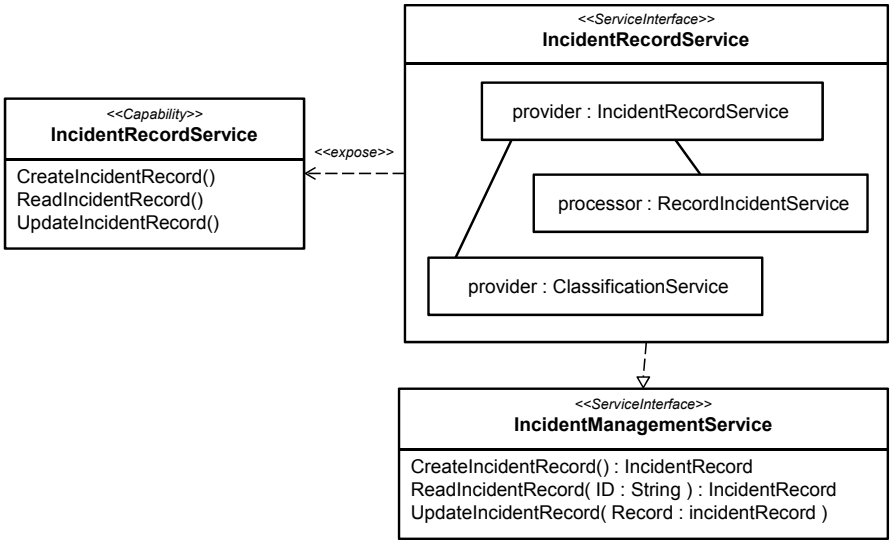

Abbildung 68 Abstrakte Dienstschnittstelle zum IncidentRecordService

Die spezifizierte abstrakte Dienstschnittstelle zum identifizierten Managementbasisdienstkandidaten `WorkaroundRecordService` wird – analog zur Spezifikation der Dienstschnittstelle für den Dienstkandidaten `IncidentRecordService` – durchgeführt und ist in nachfolgender Abbildung 69 dargestellt.

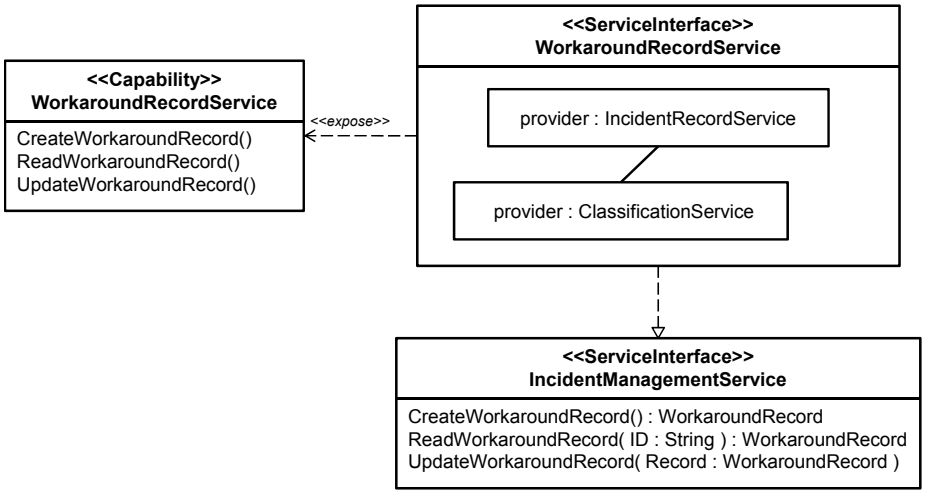

Abbildung 69 Abstrakte Dienstschnittstelle zum WorkaroundRecordService

Die dargestellten abstrakten Dienstschnittstellen sind der Übersicht halber vereinfacht abgebildet, d. h., die dem Entwicklungsvorgehen aus Kapitel 5 zugrunde gelegte Klassifikation der entworfenen Dienstschnittstellen ist in den Schnittstellenmodellen nicht definiert.

Bewertung der Wiederverwendbarkeit

Analog zur Untersuchung der Aspekte der Wiederverwendbarkeit bei den identifizierten Dienstkandidaten wurde im betrachteten Szenario die Wiederverwendbarkeit der entworfenen Dienstschnittstellen untersucht und bewertet. Die Anwendung der in Kapitel 5 vorgestellten Regeln für die schrittweise Verfeinerung von Dienstschnittstellen aus den identifizierten Dienstkandidaten stellt sicher, dass die entworfenen Dienstschnittstellen optimal bezüglich der betrachteten Aspekte von Wiederverwendbarkeit sind, wenn in der Analyse bereits eine Überarbeitung stattgefunden hat.

Es ergeben sich lediglich leicht unterschiedliche Werte für den untersuchten Aspekt der Disjunktheit bei der Betrachtung des funktionalen Kontextes, da hier mit der tatsächlich vorhandenen Anzahl an Klassifikationselementen β ein unterschiedlicher Normungsfaktor angewandt wird, bedingt durch die unterschiedliche Anzahl an möglichen Klassifikationselementen:

$$\beta = \frac{1}{3} * \sum_{i=0}^{i<3} AoCD_{MSI}(MSI_i) = \frac{1}{3} * (3 + 3 + 3 + 4 + 4) = \frac{17}{3}$$

An der grundlegenden Ausrichtung der bereits bestimmten Eigenschaften ändert sich somit jedoch nichts.

Dienstkomponenten

Auf Basis der bislang spezifizierten abstrakten Dienstschnittstellen können weiterhin zusätzliche Details von Managementprozessdiensten, die bereits zur Entwurfszeit bekannt sind, festgelegt werden. Ein wichtiger Aspekt ist die Definition der prinzipiellen Architektur sowie deren grundlegenden Elemente der zu entwerfenden Managementdienste. Hierzu eignet sich, wie im Entwicklungsvorgehen in Abschnitt 5.3.3 (Seite 155 ff.) beschrieben, die Festlegung logischer Komponenten durch die Definition von SoaML-*Participants* (engl. Teilnehmer).

ServiceParticipants

Der Entwurf der logischen Dienstkomponenten für die beiden im betrachteten Szenario identifizierten Managementbasisdienste ergibt sich direkt aus der Verfeinerung der spezifizierten Dienstschnittstellen.

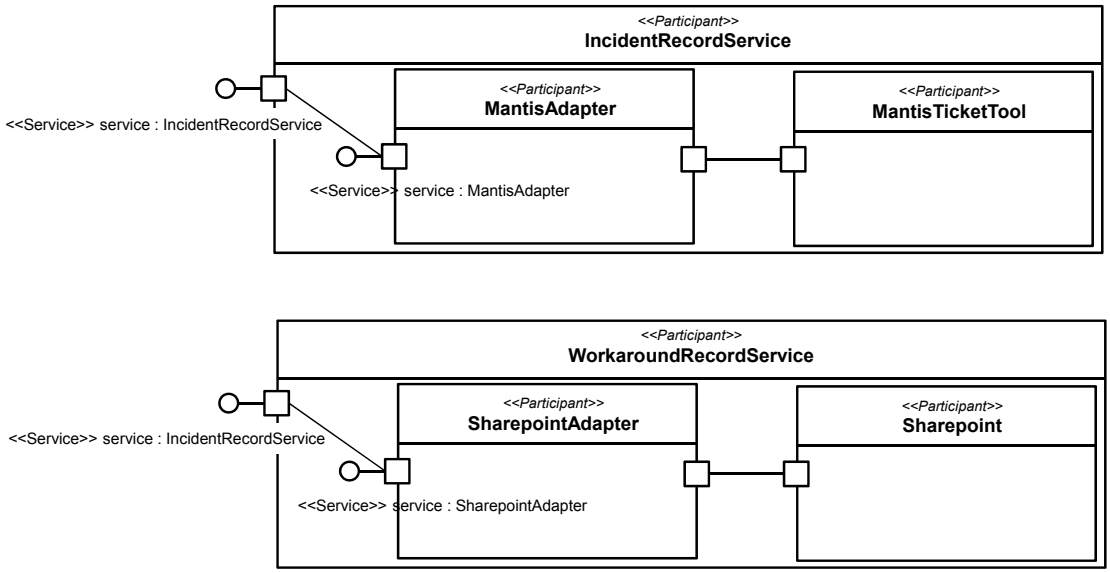

Abbildung 70 Logische Dienstkomponenten der Managementbasisdienste

Die beiden logischen Dienstkomponenten für die Managementbasisdienste sind in Abbildung 70 dargestellt. Hierbei wird ersichtlich, wie im SoaML-Modell die bestehenden Managementwerkzeuge als Teilnehmer dargestellt werden, dabei logisch gesehen jedoch Realisierungen der den Anforderungen entsprechenden Managementbasisdienste darstellen.

Im Gegensatz hierzu sind die logischen Dienstkomponenten der spezifizierten Managementprozessdienste nicht als integrierende Dienste zu bestehenden Managementwerkzeugen definiert, sondern übernehmen die definierten Abhängigkeiten der Schnittstellenmodelle (siehe hierzu auch Abbildung 71).

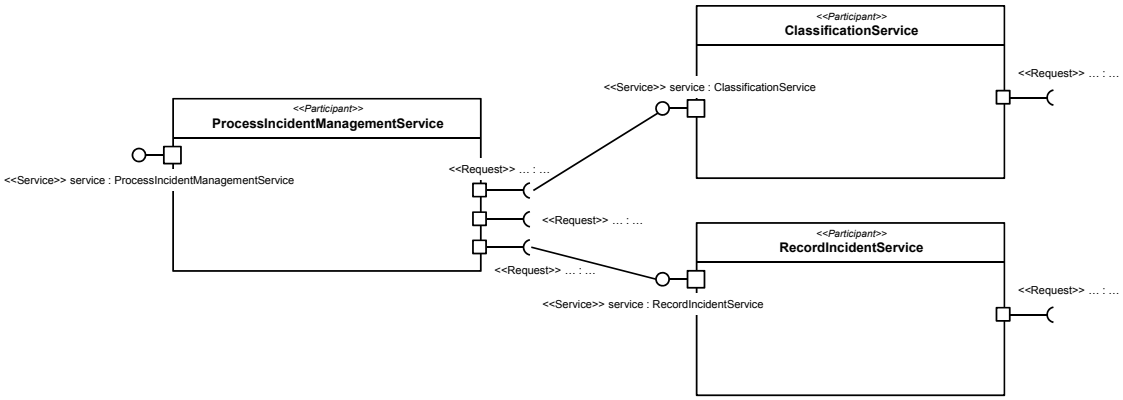

Abbildung 71 Logische Dienstkomponenten der Managementprozessdienste

6.1.4 Konkrete Dienstschnittstellen

Die weiteren Schritte zur Implementierung der Komponenten zur dienstorientierten Integration der im betrachteten Szenario bestehenden Managementwerkzeuge werden im Folgenden ausschnittsweise dargestellt. Insgesamt kann aber festgestellt werden, dass der vergrößerte Aufwand in der dienstorientierten Analyse und im dienstorientierten Entwurf sowie der explizite Einsatz einer dedizierten Ontologie für die Domäne die weitere Implementierung vereinfacht, da in den entstehenden Artefakten jeweils Bezüge zu den relevanten Konzepten hergestellt werden können.

Definition von Webservice-Schnittstellen

Im Folgenden werden beispielhaft an der spezifizierten Webservice-Schnittstelle für den Managementbasisdienst `IncidentRecordService` zur Integration des bestehenden *Trouble-Ticket*-Werkzeuges die weiteren anfallenden Schritte skizziert.

Abbildung 72 gibt zunächst eine Übersicht über die im Entwicklungswerkzeug erstellte WSDL-Datei.

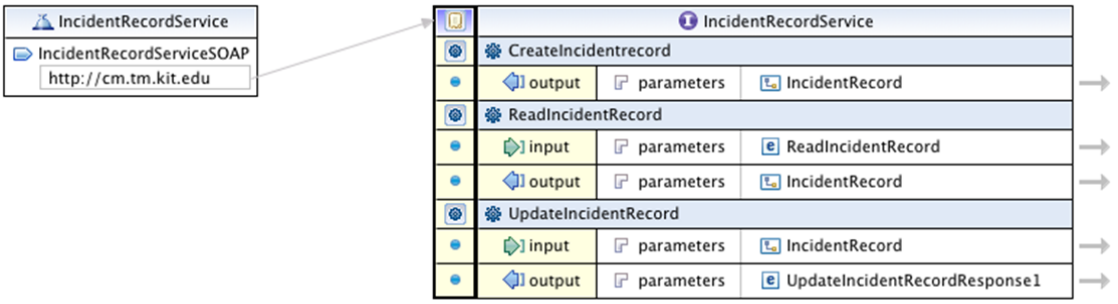

Abbildung 72 Webservice-Schnittstelle von IncidentRecordService

Wie in der SoaML-Spezifikation der abstrakten Dienstschnittstelle festgelegt, verfügt der entworfene Managementdienst über die drei Dienstoperationen `CreateIncidentRecord`, `ReadIncidentRecord` und `UpdateIncidentRecord`. Den Regeln für den Entwurf von

Managementdiensten aus Kapitel 5 folgend sind die Dienstoperationen mit den erforderlichen Parametern ausgestattet. Auf der Grundlage der im Domänenmodell definierten Entitätsstrukturrichtlinien können die einzelnen Dienstnachrichten für die verschiedenen Dienstoperationen spezifiziert werden. Hierbei wird deutlich, dass durch den domänengetriebenen Ansatz eine durchgehende Überführung nicht nur der funktionalen Beziehungen der Dienstoperationen, sondern auch darüber hinausgehend auch die Überführung von erforderlichen Attributen für den nachrichtenbasierten Aufruf einzelner Dienstoperationen möglich ist.

Die zur vorgestellten Definition gehörende WSDL-Datei ist in nachfolgendem Quelltextauszug dargestellt.

```
<?xml version="1.0" encoding="UTF-8" standalone="no"?>
<wsdl:definitions xmlns:soap="http://schemas.xmlsoap.org/wsdl/soap/"
xmlns:tns="http://cm.tm.kit.edu/IncidentRecordService/" xmlns:wsdl="http://schemas.xmlsoap.org/wsdl/"
xmlns:xsd="http://www.w3.org/2001/XMLSchema" name="IncidentRecordService"
targetNamespace="http://cm.tm.kit.edu/IncidentRecordService/">
 <wsdl:types>
  <xsd:schema targetNamespace="http://cm.tm.kit.edu/IncidentRecordService/">
   <xsd:element name="ReadIncidentRecord">
    <xsd:complexType>
     <xsd:sequence>
      <xsd:element name="in" type="xsd:string"/>
     </xsd:sequence>
    </xsd:complexType>
   </xsd:element>
   <xsd:element name="NewOperationResponse">
    <xsd:complexType>
     <xsd:sequence>
      <xsd:element name="out" type="xsd:string"/>
     </xsd:sequence>
    </xsd:complexType>
   </xsd:element>
   <xsd:element name="CreateIncidentrecord">
        <xsd:complexType>
                <xsd:sequence>
                        <xsd:element name="in" type="xsd:string"></xsd:element>
                </xsd:sequence>
        </xsd:complexType>
   </xsd:element>
   <xsd:element name="CreateIncidentrecordResponse">
        <xsd:complexType>
                <xsd:sequence>
                        <xsd:element name="out" type="xsd:string"></xsd:element>
                </xsd:sequence>
        </xsd:complexType>
   </xsd:element>
   <xsd:element name="ReadIncidentRecordResponse1">
        <xsd:complexType>
```

```
                <xsd:sequence>
                        <xsd:element name="out" type="xsd:string"></xsd:element>
                </xsd:sequence>
        </xsd:complexType>
  </xsd:element>
  <xsd:element name="UpdateIncidentRecord">
        <xsd:complexType>
                <xsd:sequence>
                        <xsd:element name="in" type="xsd:string"></xsd:element>
                </xsd:sequence>
        </xsd:complexType>
  </xsd:element>
  <xsd:element name="UpdateIncidentRecordResponse1">
        <xsd:complexType>
                <xsd:sequence>
                        <xsd:element name="out" type="xsd:string"></xsd:element>
                </xsd:sequence>
        </xsd:complexType>
  </xsd:element>

      <xsd:complexType name="IncidentRecord"></xsd:complexType>
    </xsd:schema>
</wsdl:types>
<wsdl:message name="ReadIncidentRecordRequest">
        <wsdl:part element="tns:ReadIncidentRecord" name="parameters" />
</wsdl:message>
<wsdl:message name="NewOperationResponse">
 <wsdl:part element="tns:NewOperationResponse" name="parameters"/>
</wsdl:message>
<wsdl:message name="CreateIncidentrecordRequest">
        <wsdl:part name="parameters" element="tns:CreateIncidentrecord"></wsdl:part>
</wsdl:message>
<wsdl:message name="CreateIncidentrecordResponse">
        <wsdl:part name="parameters" type="tns:IncidentRecord"></wsdl:part>
</wsdl:message>
<wsdl:message name="ReadIncidentRecordResponse1">
        <wsdl:part name="parameters" type="tns:IncidentRecord"></wsdl:part>
</wsdl:message>
<wsdl:message name="UpdateIncidentRecordRequest">
        <wsdl:part name="parameters" type="tns:IncidentRecord"></wsdl:part>
</wsdl:message>
<wsdl:message name="UpdateIncidentRecordResponse1">
        <wsdl:part name="parameters" element="tns:UpdateIncidentRecordResponse1"></wsdl:part>
</wsdl:message>
<wsdl:portType name="IncidentRecordService">
 <wsdl:operation name="CreateIncidentrecord">
        <wsdl:output message="tns:CreateIncidentrecordResponse"></wsdl:output>
 </wsdl:operation>
 <wsdl:operation name="ReadIncidentRecord">
```

```
        <wsdl:input message="tns:ReadIncidentRecordRequest"></wsdl:input>
        <wsdl:output message="tns:ReadIncidentRecordResponse1"></wsdl:output>
  </wsdl:operation>
  <wsdl:operation name="UpdateIncidentRecord">
        <wsdl:input message="tns:UpdateIncidentRecordRequest"></wsdl:input>
        <wsdl:output message="tns:UpdateIncidentRecordResponse1"></wsdl:output>
  </wsdl:operation>
 </wsdl:portType>
 <wsdl:binding name="IncidentRecordServiceSOAP"
        type="tns:IncidentRecordService">
        <soap:binding style="document"
                transport="http://schemas.xmlsoap.org/soap/http" />
        <wsdl:operation name="ReadIncidentRecord">
                <soap:operation
                        soapAction="http://cm.tm.kit.edu/IncidentRecordService/ReadIncidentRecord" />
                <wsdl:input>
                        <soap:body use="literal" />
                </wsdl:input>
                <wsdl:output>
                        <soap:body use="literal" />
                </wsdl:output>
        </wsdl:operation>
        <wsdl:operation name="CreateIncidentrecord">
                <soap:operation
                        soapAction="http://cm.tm.kit.edu/IncidentRecordService/CreateIncidentrecord" />
                <wsdl:output>
                        <soap:body use="literal" />
                </wsdl:output>
        </wsdl:operation>
        <wsdl:operation name="UpdateIncidentRecord">
                <soap:operation
                        soapAction="http://cm.tm.kit.edu/IncidentRecordService/UpdateIncidentRecord" />
                <wsdl:input>
                        <soap:body use="literal" />
                </wsdl:input>
                <wsdl:output>
                        <soap:body use="literal" />
                </wsdl:output>
        </wsdl:operation>
 </wsdl:binding>
 <wsdl:service name="IncidentRecordService">
  <wsdl:port binding="tns:IncidentRecordServiceSOAP" name="IncidentRecordServiceSOAP">
    <soap:address location="http://cm.tm.kit.edu"/>
  </wsdl:port>
 </wsdl:service>
</wsdl:definitions>
```

Quelltext 24 WSDL-Beschreibung des IncidentRecordService

6.1.5 Fortsetzung des Entwicklungsvorgehens

Mit der Definition der konkreten Dienstschnittstellen sowie der vorgehenden Spezifikation der logischen Dienstkomponenten kann die weitere Implementierung der erforderlichen Dienstlogik erfolgen. Die folgenden Schritte werden im Folgenden kurz angerissen und skizziert, da die Implementierung der identifizierten und entworfenen Dienste aktuell Gegenstand im Projekt in der ATIS sind.

Die Implementierung der Managementbasisdienste zur Integration der bestehenden Managementwerkzeuge führt zu einer Implementierung der definierten Adapterkomponenten, in den Teilnehmer-Modellen dargestellt durch die Festlegung der abgebildeten Dienstschnittstellen auf die jeweiligen Adapterkomponenten (`MantisAdapter`-Teilnehmer, `SharepointAdapter`-Teilnehmer). Bei der Implementierung der entsprechenden Adapterkomponenten kann jeweils eine eigene Technologie-Entscheidung getroffen werden, wodurch ein Höchstmaß an Flexibilität erhalten bleibt.

Die Umsetzung der identifizierten und spezifizierten Managementprozessdienste durch Definition von Dienstkompositionen auf Basis der vorhandenen oder identifizierten Managementbasisdienste kann beispielsweise durch die Definition von ausführbaren *Workflows* durch die *Business Process Execution Language for Webservices* (BPEL-WS) erfolgen. Die Umsetzung der Ablauflogik erfolgt auf Basis der modellierten Betreiberprozesse sowie den spezifizierten Interaktionsprotokollen der einzelnen Dienstschnittstellen.

6.1.6 Bewertung

Die iterative Fortsetzung des betrachteten Entwicklungsvorgehens für die vollständige dienstorientierte Integration des bestehenden Wissensbasiswerkzeuges kann auf der Grundlage der erstellten und teilweise bereits identifizierten Dienstkandidaten und spezifizierten Dienstschnittstellen erfolgen. Hierbei wird ersichtlich, dass die Überarbeitung des Dienstkandidaten zur Integration des Wissensbasiswerkzeuges bereits vollständig bezüglich des CRU-Musters erweitert wurde, wodurch spätere oder nachträgliche Änderungen des im *Incident Management* genutzten Managementdienstes vorweggenommen wurden.

Die Ausführungen am betrachteten Anwendungsbeispiel für den Einsatz von Domänenmetamodell, Modellen für die betrachteten Managementbereiche *Incident Management* und *Problem Management* sowie die auf dem domänengetriebenen Entwurf aufbauende explizite Betrachtung verschiedener Aspekte von Wiederverwendbarkeit zeigen die Möglichkeiten auf, die die vorgestellten Beiträge liefern. Die Vorteile von einer Domänenmodellierung, die einem zielführenden Dienstentwurf zugrunde liegt, wurden bereits in bestehenden Arbeiten ausgearbeitet und exemplarisch beschrieben [AG+08, Fa08, EH+08].

Durch den vorgestellten erweiterten Ansatz, auf Basis eines einheitlichen Metamodells der betrachteten Domäne zunächst ein einheitliches Begriffsverständnis zu schaffen und darauf aufbauend dann Wiederverwendbarkeit und Betreiberprozessorientierung des zu entwerfenden Managementsystems zu untersuchen, werden zunächst zusätzliche Schritte in den Entwicklungsprozess eingeführt. Insgesamt kann jedoch festgestellt werden, dass für alle

beschriebenen Beiträge eine hinreichend gute Unterstützung auch in bestehenden Entwicklungswerkzeugen besteht, sodass am hier betrachteten Beispiel festgestellt werden kann, dass der verfolgte Entwicklungsprozess insgesamt zielführend zum Einsatz kommen kann. Die Möglichkeit, die Wiederverwendbarkeit der entstehenden Artefakte (Dienstkandidaten, Dienstschnittstellen) bereits frühzeitig im Entwicklungsprozess zu untersuchen und gegebenenfalls zu entgegen zu wirken, stellt eine Verbesserung gegenüber bestehenden Ansätzen dar.

6.2 Zusammenfassung und erzielter Beitrag

Die Demonstration der Tragfähigkeit des betrachteten Entwicklungsvorgehens ist insgesamt aufgeteilt in zwei verschiedene Teile. Zunächst wird am Beispiel eines konkreten durchgeführten Projektes ein Anwendungsbeispiel für die im Entwurf von Managementdiensten durchgeführten Schritte vorgelegt. Anhand dieses Anwendungsbeispiels werden die einzelnen verschiedenen Aktivitäten der dienstorientierten Analyse und des dienstorientierten Entwurfes sowie darauf aufbauend die für die Implementierung von Managementdiensten erforderlichen Definitionen von konkreten Dienstschnittstellen diskutiert. Hierbei wird anschaulich dargestellt, wie die Modellierung der betrachteten Domäne, die Modellierung der zu unterstützenden Betreiberprozesse und die Modellierung der für die Umsetzung des zu konstruierenden Systems erforderlichen Dienstmodelle durchgeführt und schrittweise verfeinert werden. Das Anwendungsbeispiel wird am Szenario eines *Incident-Management*-Prozesses sowie den hierzu zu integrierenden bestehenden Managementwerkzeugen (*Trouble-Ticket*-Werkzeug, Wissensbasiswerkzeug) beschrieben. Das aufgezeigte Anwendungsbeispiel verdeutlicht das methodische Vorgehen zum Entwurf von Managementdiensten, die bezüglich zweier für die Elemente einer dienstorientierten Architektur grundlegender Eigenschaften dieser Elemente betrachtet werden: die Orientierung der entworfenen Managementdienste an den zu unterstützenden Betreiberprozessen, sowie die Fokussierung des Entwurfes auf Artefakte, die im Resultat eine mögliche Wiederverwendung günstig unterstützen.

Im zweiten Teil der Demonstration der Tragfähigkeit werden daher die im Anwendungsbeispiel entworfenen Managementdienste bezüglich dieser beiden Eigenschaften gezielt untersucht. Die im vierten Kapitel diskutierten Aspekte von Wiederverwendbarkeit, die in den bereits während des Entwurfes zugrunde liegenden Artefakten ausgemacht werden können, werden am Anwendungsbeispiel kritisch untersucht und eingeordnet. Es wird aufgezeigt, inwiefern das betrachtete Entwicklungsvorgehen dazu eingesetzt werden kann, die entstehenden Artefakte überhaupt bezüglich Wiederverwendbarkeit zu untersuchen. Es wird verdeutlicht, dass die explizite Modellierung der zugrunde liegenden Domäne eine wesentliche Grundlage liefert, um ein gemeinsames Verständnis zwischen den beteiligten Akteuren in der Analysephase (Analyst) und in der Entwurfsphase (Architekt) zu etablieren, wodurch letztlich sichergestellt wird, dass die aus dem Entwurf resultierenden Artefakte (spezifizierte Dienstmodelle) an den zu unterstützenden Betreiberprozessen ausgerichtet sind. Darauf aufbauend werden dann die einzelnen Aspekte von Wiederverwendbarkeit in den Analyse- und Entwurfsmodellen hin untersucht.

Mit dem Anwendungsbeispiel des methodischen Vorgehens und der Bewertung der entworfenen Artefakte bezüglich der Wiederverwendbarkeit wird in diesem Kapitel daher insgesamt ein zu den

beiden vorangegangen Kapiteln ergänzender Beitrag geliefert, der den Einsatz der konzeptionellen Beiträge an einem durchgehenden und realen Entwicklungsprojekt aufzeigt.

7 Ergebnisbewertung und Ausblick

In diesem letzten Kapitel werden die erzielten Ergebnisse zusammengefasst und vor dem Hintergrund der in Abschnitt 3.1 definierten Anforderungen bewertet. Anschließend werden mögliche Anknüpfungspunkte für nachfolgende Arbeiten aufgezeigt, die einige der fokussierten Aspekte aufgreifen und vertiefen können.

7.1 Zusammenfassung der Beiträge der Arbeit

Die vorliegende Arbeit hat zum Ziel, die automatisierte Ausführung von Betreiberprozessen auf Basis bestehender Managementwerkzeuge zu ermöglichen. Hierzu wird der Entwurfsprozess einer dienstorientierten Architektur betrachtet, um das hierzu notwendige verteilte, integrierte Informationssystem realisieren zu können. Um ein tragfähiges Gesamtkonzept zu entwickeln, wird der Entwurf von drei unterschiedlichen Perspektiven (Betreiberprozess, Softwarearchitektur und Managementwerkzeug) her motiviert. Insgesamt liefert die Arbeit hierzu zwei Beiträge.

Im Mittelpunkt des ersten Beitrages steht ein Metamodell der Domäne IT-Management, das diese drei Perspektiven vereinigt und im Rahmen einer durchgängigen Entwicklungsmethodik die dienstorientierte Analyse direkt ermöglicht sowie darauf aufbauend den dienstorientierten Entwurf unterstützt. Die unterschiedlichen Aspekte des konzipierten Metamodells werden hierzu auf Basis verschiedener Ansätze zur Konkretisierung umgesetzt. So wird der Teilaspekt des Metamodells, der die strukturelle Analyse der Domäne zum Gegenstand hat, auf Basis einer OWL-Ontologie [W3C-OWL] spezifiziert. Dieser Ansatz bietet den wesentlichen Vorteil, unabhängig von verschiedenen Typen von Anforderungsartefakten zu sein, gleichwohl eine durchgehende modellgestützte, dienstorientierte Analyse zu ermöglichen. Der Teilaspekt des konzipierten Metamodells, der die Identifikation von Managementdienstkandidaten zum Gegenstand hat, wurde dagegen auf die durch die SoaML vorgegebene konkrete Syntax abgebildet. Ein weiterer wesentlicher Aspekt dieses Beitrages ist die konstruktive Bewertung der durch den Ansatz entwickelten Managementdienste bezüglich der Anforderung der Wiederverwendbarkeit. Konkret werden hierzu die notwendigen Modellbildungen betrachtet und in das vorgestellte Entwicklungsvorgehen integriert. So können die Entwurfsentscheidungen verschiedener Fachentwickler frühzeitig durch den Einsatz entsprechender, auf den Modellen zur Bewertung der Wiederverwendbarkeit basierender Werkzeugerweiterungen, unterstützt werden.

Der auf dem vorgestellten Metamodell der Domäne durchgeführte domänengetriebene Entwurf ist Gegenstand des zweiten Beitrages. Während im ersten Beitrag mit der Analyse der wesentlichen strukturellen Elemente der Domäne IT-Management die Grundlage geschaffen wird, um auf des Basis eines einheitlichen Metamodells den Entwurf wiederverwendbarer Managementdienste zu unterstützen, werden im zweiten Beitrag verschiedene Aspekte eines methodischen Vorgehens beschrieben, um den Entwurf von Basisdiensten und Prozessdiensten nachvollziehbar zu gestalten. Mit der detaillierten Betrachtung des Entwurfes von Managementdiensten zeigt der zweite Beitrag daher auf, wie die Integration bestehender Managementwerkzeuge im Kontext eines dienstorientierten Softwareentwicklungsprozesses gelöst werden kann. Das Vorgehen zur Integration gliedert sich nahtlos in den vorgestellten Entwurf von Managementdiensten ein. Auf Basis der im ersten Beitrag

spezifizierten Metamodelle wird gezeigt, wie die Ergänzung eines konkreten vorliegenden Entwurfes von Managementdiensten erweitert werden muss, damit eine syntaktische und semantische Integration bestehender Managementwerkzeuge erfolgen kann. Hierzu wird das im ersten Beitrag durch eine OWL-Ontologie spezifizierte Metamodell der Domäne eingesetzt, um entstandene Werkzeugmodelle und fachfunktionale Modelle in einem integrierten Entwicklungsprozess zu nutzen. Die Abbildung von automatisierbaren Betreiberprozessen wird als direkte Fortführung des Entwurfes der Managementbasisdienste diskutiert. Es wird aufgezeigt, inwiefern die erweiterte Betrachtung von Prozessmodellen genutzt werden kann, um auf Basis der für den Entwurf von Managementbasisdiensten erstellten und genutzten Domänenmodelle die Modellierung der dynamischen Abläufe in komplexen Betriebsszenarien heranziehen zu können. Als ergänzender Beitrag werden verschiedene Referenzmodelle für ausgewählte Betreiberprozesse vorgestellt.

In Kapitel 6 wird die Tragfähigkeit des vorgestellten Ansatzes an einem realen Entwicklungsprojekt demonstriert, das im Kontext einer Kooperation des Lehrstuhls C&M mit der ATIS am KIT durchgeführt wurde. Hierbei wurden verschiedene bestehende Managementwerkzeuge, die im Kontext der Störungsbearbeitung eingesetzt werden, mit dienstorientierten Schnittstellen ausgestattet, um die Abbildung des *Incident-Management*-Prozesses gemäß ISO/IEC20000-1:2005 zu ermöglichen. Die entworfenen Schnittstellen ermöglichen den dienstorientierten Zugriff auf elementare Funktionalität der Managementwerkzeuge und gliedern sich darüber hinausgehend in einen ganzheitlichen dienstorientierten Ansatz ein. Die entworfenen Dienstschnittstellen sind durch die Betrachtung der Wiederverwendbarkeit optimal bezüglich der in Abschnitt 4.3 vorgestellten ausgewählten und formal beschriebenen Aspekte.

7.2 Diskussion und Bewertung der Ergebnisse

Die kritische Bewertung der eigenen Lösungen und Beitrage vor dem Hintergrund der in Kapitel 3 definierten Anforderungen wird im Folgenden detailliert vorgenommen. Hierbei wird aufgezeigt, inwiefern die einzelnen Anforderungen durch die vorgestellte Methode erfüllt und den identifizierten Problemstellungen gerecht werden.

7.2.1 Metamodell für den Entwuf wiederverwendbarer Managementdienste

Der Entwurf wiederverwendbarer Managementdienste auf Basis einer strukturierten Modellierung der Domäne IT-Management stellt die Grundlage dar, um auf Basis bestehender Managementwerkzeuge eine automatisierte Ausführung von Betreiberprozessen er erreichen. Die in Abschnitt 3.1.1 definierten Anforderungen werden – in der relativen Betrachtung zu den untersuchten Arbeiten – erfüllt.

Referenzcharakter des Dienstentwurfes

Die Ausführung automatisierbarer Betreiberprozesse auf Basis bestehender Managementwerkzeuge erfordert die Konstruktion dienstorientierter Schnittstellen. Hierzu ist ein nachvollziehbares, wiederholbares und zielführendes Verfahren notwendig, um den verschiedenen Aspekten

(Betreiberaspekt, Softwarearchitekturaspekt, Softwareentwicklungsaspekt) gerecht zu werden. Die in der vorliegenden Arbeit vorgestellte Methode erfüllt diese Anforderung und liefert hierzu durch die beiden Beiträge eine wesentliche Verbesserung gegenüber bestehenden Arbeiten. Durch die getrennte Betrachtung von strukturellem Modell und Vorgehen kann der Beitrag an eigene Gegebenheiten in konkreten Entwicklungsprojekten angepasst werden. Der Einsatz von OWL zur Spezifikation des Metamodells oder zur Spezifikation einer Ontologie für einen ausgewählten Anwendungsfall ermöglicht die weitergehende Evolution des Ansatzes auf der Grundlage formal definierter Semantik. Unterstützung durch Entwicklungswerkzeuge ist vorhanden (beispielsweise: *Protégé* [Protege]). Für die Modellierung von Dienstkandidaten und abstrakten Dienstschnittstellen wird ebenfalls auf verfügbare Sprachen (SoaML) zurückgegriffen, wofür ebenfalls die Unterstützung in modernen Entwicklungswerkzeugen vorliegt (*Rational Software Architect* [IBM-RSA]). Insgesamt werden die Kriterien hinsichtlich des Referenzcharakters des Dienstentwurfes mit dem vorgestellten Ansatz erfüllt.

Unabhängigkeit von konkreten Managementwerkzeugen

Der Einsatz eines Metamodells, das unabhängig von konkreten Szenarien, Managementwerkzeugen oder Anwendungsfällen sowie vollständig entlang den wesentlichen Eigenschaften der zu unterstützenden Domäne ausgerichtet ist, ermöglicht die Anwendbarkeit des Vorgehens in beliebigen Entwicklungsprojekten. Das in Kapitel 4 vorgestellte Metamodell ist dahingehend abstrahiert ausgelegt, dass eine von konkreten Managementwerkzeugen unabhängige Modellierung von den durch die Werkzeuge angebotenen Managementfunktionen durchgeführt werden kann. Hierdurch können sowohl Aspekte für den technisch-orientierten Betrieb von IT-Infrastrukturen, aber auch Aspekte für die organisatorische Unterstützung komplexer Betriebsabläufe erfasst werden. Insgesamt ist die vorgestellte Methode für den Entwurf wiederverwendbarer Managementdienste unabhängig von konkreten Managementwerkzeugen.

Metamodell der Domäne

Wesentliches Merkmal des ersten Beitrages ist die Konstruktion eines Metamodells der zu unterstützenden Domäne. Somit können fachfunktionale Anforderungen auf eine von konkreten Managementwerkzeugen unabhängige Art und Weise erfasst und in einen zielführenden Softwareentwicklungsprozess überführt werden. Insgesamt ist die in der vorliegenden Arbeit vorgestellte Methode darauf ausgelegt, einen umfassenden Entwurf von Managementdiensten für die Domäne IT-Management zu unterstützen. Sowohl das vorgestellte Metamodell als auch die an den Elementen des Metamodells ausgerichtete systematische Methode für den Entwurf wiederverwendbarer Managementdienste sind spezifisch für die betrachtete Domäne. Prinzipiell kann eine Anwendung der vorgeschlagenen Methodenschritte auch in weiteren Fachbereichen durchgeführt werden, wobei Anpassungen an konkrete Gegebenheiten vorzunehmen sind.

Wiederverwendbarkeit von Managementdiensten

Zentrales Merkmal des vorgestellten Metamodells ist die fokussierte Betrachtung, inwiefern die durch den vorgestellten Ansatz entworfenen Managementdienste gewisse Eigenschaften hinsichtlich eines hohen Grades an Wiederverwendbarkeit aufweisen. Die Wiederverwendbarkeit stellt ein zentrales

Kriterium in dienstorientierten Architekturen dar, da durch die Trennung zwischen fachlicher Schnittstelle und realisierender Implementierung eine Wiederverwendung bestehender Softwareartefakte in vernetzten Systemen angestrebt wird. Wie bei der Betrachtung bestehender Arbeiten aufgezeigt wird, unterstützen bekannte Ansätze diesen Aspekt nur unzureichend, bzw. oftmals gar nicht. Zur Begegnung dieser Problematik wird hierzu ein Katalog von verschiedenen Kriterien aufgestellt, der als Abstraktion bestehender Arbeiten in Bezug auf diese Thematik den Gegebenheiten in der Domäne IT-Management angepasst und erweitert wird. Die verschiedenen Kriterien werden ausführlich beschrieben und in den Kontext der jeweils zugehörigen Modellaspekte eingeordnet. Darüber hinausgehend sind einige wesentliche Kriterien durch die Definition einer formalen Berechnungsvorschrift spezifiziert, wodurch die praktische Anwendbarkeit in realen Entwicklungsprojekten sichergestellt ist, vorausgesetzt, es existiert eine hinreichende Unterstützung in den eingesetzten Entwicklungswerkzeugen.

7.2.2 Domänengetriebener Entwurf wiederverwendbarer Managementdienste

Im zweiten Kernbeitrag wird aufgezeigt, wie auf Basis des in Kapitel 4 vorgestellten Metamodells und der hieraus resultierenden Spezifikation in Form einer OWL-Ontologie der Entwurf wiederverwendbarer Managementdienste systematisch beschrieben werden kann. Wenngleich durch die Komplexität der Fragestellung und der hiermit verbundenen Diversität verschiedener Aspekte keine durchgehende Transformation verschiedener Modelle spezifiziert wurde, ist eine Abbildung dynamischer Aspekte von Betreiberprozessen auf die Elemente einer dienstorientierten Architektur mit der vorgestellten Methode sukzessive möglich.

Flexible Anwendbarkeit der Ableitungsmethode

In der Analyse bestehender Arbeiten wurde festgestellt, dass zwar einzelne Arbeiten eine flexible Anwendung der Methode zur Abbildung von Betreiberprozessen betrachten (z. B. Mayerl et al., Brown et al.), in gleichem Maße jedoch einige der wesentlichen weiteren Anforderungen nicht erfüllen. Die in der vorliegenden Arbeit vorgestellte Methode ist dahingehend flexibel in der Anwendung, als dass ausgerichtet an den wesentlichen Elementen des Metamodells eine grundlegende Trennung einzelner Methodenschritte erfolgt. Insgesamt sind die Methodenschritte jedoch weitestgehend unabhängig voneinander, sowie unabhängig von konkreten eingesetzten Technologien, Managementwerkzeugen oder Modellierungssprachen. Beispielsweise kann der vorgestellte domänengetriebene Entwurf als Verfeinerung des in [EH+08] vorgestellten Vorgehens für den Entwurf von Anwendungsdiensten aufgefasst werden, wobei in der vorgestellten Methode der Entwurf von Managementbasisdiensten und Managementprozessdiensten vergleichbar mit dem Entwurf von Anwendungsdiensten auf Basis formal definierter Domänenmodelle ist.

Einheitliche Begriffe auf Basis eines Domänenmodells

Grundlegende Voraussetzung für die Beteiligung verschiedener Teilnehmer an einem komplexen Softwareentwicklungsprojekt ist die Festlegung auf ein einheitliches Begriffsverständnis. Dies bezieht

sich nicht ausschließlich auf die strukturellen Aspekte der zur unterstützenden Domäne, wie beispielsweise die Bedeutung einzelner Managementteilnehmer, Managemententitäten oder Richtlinien, sondern auch auf die Bedeutung komplexer und zusammengesetzter Managementfunktionen, die beispielsweise bei der Integration bestehender Managementwerkzeuge beachtet werden müssen. Hier ist der Einsatz entsprechender Ansätze erforderlich, um die Semantik der jeweiligen Modellelemente formal ausdrücken zu können, insbesondere derjenigen Modellelemente, die bei Modellierung von Betreiberprozessen erforderlich sind. Im vorgestellten systematischen Vorgehen wird hierzu der Einsatz von einer auf OWL basierenden Ontologie betrachtet und am Beispiel einer Ontologie für die Störungsbearbeitung aufgezeigt. Die Spezifikation einer Ontologie für einen ausgewählten Bereich ermöglicht darüber hinausgehend die Festlegung auf ein einheitliches Begriffsverständnis, damit im Rahmen von Integrationsaufgaben eine fachliche Modellierung der bestehenden Managementwerkzeuge auf der Grundlage der erstellten Ontologie möglich ist. Insgesamt führt die vorgestellte Methode daher ein Vorgehen ein, um zunächst über die strukturelle Modellierung der wesentlichen Aspekte der Domäne den Entwurf von prozessunabhängigen Managementdiensten zu ermöglichen, und anschließend auf Basis der erstellten Modelle die Abbildung von automatisierbaren Betreiberprozessen zu unterstützen.

Formale Modellierung von Managementprozessen

Die Modellierung von Betreiberprozessen wird durch den Einsatz von formal definierten Modellierungssprachen unterstützt. Prinzipiell ist keine Festlegung auf eine konkrete Syntax erforderlich, da die Konkretisierung der im konzeptionellen Metamodell der Domäne identifizierten Elemente der abstrakten Syntax durch verschiedene, ausgereifte und in realen Entwicklungsprozessen eingesetzte Modellierungssprachen möglich ist. Um die vorgestellte Methode in konkreten Entwicklungsprojekten einfach und direkt einsetzen zu können, wird aber über diese Feststellung hinausgehend die Modellierung von Betreiberprozessen mit der *Business Process Model and Notation* (BPMN) [OMG-BPMN] durchgeführt. Es wird aufgezeigt, wie die erstellen Domänenmodelle schrittweise in BPMN-Modelle überführt werden können, um somit den Entwurf der die dynamischen Aspekte unterstützenden Managementprozessdienste zu ermöglichen. Insgesamt genügt die vorgestellte Methode daher dieser Anforderung.

7.3 Ausblick

Die vorliegende Arbeit verbindet grundlegende Fragestellungen aus zwei umfangreichen Teilgebieten der Informatik. Sowohl die Softwareentwicklung als auch der Betrieb komplexer und untereinander vernetzter IT-Infrastrukturen ist in zahlreichen Arbeiten Gegenstand und Thema. Eine auf die in der vorliegenden Arbeit fokussierte Fragestellung findet nach aktuellem Kenntnisstand in vergleichbarer Art und Weise bislang nicht statt. Während der Bearbeitung der Problemstellungen ergaben sich weitere Aspekte, die im Folgenden kurz dargestellt werden sollen, und die als Ausgangspunkt für anknüpfende Fragestellungen dienen kann.

Qualitative Aspekte von Managementdiensten

In der vorliegenden Arbeit wird der Entwurf von Managementdiensten betrachtet und auf die fachfunktionalen Aspekte von umzusetzenden Managementfunktionen eingeschränkt. Dies stellt

zunächst keine Einschränkung dar, da von einem konzeptionellen Standpunkt aus gesehen die Überführung qualitativer Anforderungen durch ergänzende Schritte in der Methode für den Entwurf von Basisdiensten und Prozessdiensten umgesetzt werden kann. Dadurch werden eventuell Änderungen am Metamodell der Domäne erforderlich, die jedoch aufgrund des offenen Konzeptes (Einsatz einer OWL-Ontologie zur Spezifikation der abstrakten Syntax des Metamodells) einfach umgesetzt werden können. Durch die erweiterte Betrachtung der qualitativen Aspekte können zusätzliche Einschränkungen bestehender Managementwerkzeuge erfasst werden, die durch das vorgestellte Metamodell bislang nicht in einer systematischen Art und Weise dargestellt werden können. So ist beispielsweise denkbar, die durch ein Managementwerkzeug angebotene Managementfunktionalität zusätzlich durch leistungsbezogene Einschränkungen der dem Werkzeug zugrunde liegenden Implementierung (z. B. sequenzieller Zugriff auf eine Datenbank, Operationsaufrufe pro Sekunde einer Schnittstelle usw.) zu beschreiben. Diese zusätzliche Information kann genutzt werden, um die bei der Abbildung automatisierbarer Bertreiberprozesse anfallenden qualitativen Anforderungen (z. B. Bearbeitungsdauer einer gemeldeten Störung in weniger als einer Stunde) umzusetzen.

Semantische Integration zur Laufzeit

Die vorliegende Arbeit fokussiert die Fragestellung, inwiefern fachliche Anforderungen an ein Managementsystem zur Abbildung automatisierbarer Betreiberprozesse in einem nachvollziehbaren und systematischen Entwicklungsvorgehen erfasst werden können. Hierbei kommen Ansätze zum Einsatz, die für den Entwurf von Managementdiensten notwendige Information durch die explizite Definition der jeweiligen Semantik festzulegen. Prinzipiell können die entstehenden Modelle genutzt werden, die Schnittstellen der im späteren Verlauf des Entwicklungsprozesses fertig implementierten Managementdienste zu erweitern, um im Falle einer über die eigentliche Zielsetzung hinausgehende Anforderung umzusetzen. Konkret kann die erstellte Ontologie genutzt werden, um die auf Basis von Webservices und der WSDL definierten Webservice-Schnittstellen erstellte Artefakte semantisch zu annotieren. Als konsequente Fortführung dieses Konzeptes ist es denkbar, auf eine Vorab-Spezifikation von zur Abbildung einer Anforderung benötigten Managementdiensten zu verzichten und die jeweils benötigten Dienste zur Laufzeit – unter zu Hilfenahme der jeweiligen semantisch erweiterten Dienstschnittstellen – aufzufinden. Erste Arbeiten betrachten bereits die Integration von Managementinformation durch semantische Ansätze [VV+02].

Vernetztes Management

In der vorliegenden Arbeit wird das integrierte Management bei einem isolierten Betreiber betrachtet. Es werden zwar Schnittstellen an der Betreibergrenze definiert (z. B. `IncidentManagementService`), eine ganzheitliche Betrachtung eines Betriebsprozesses über mehrere Betreibergrenzen hinweg ist allerdings nicht untersucht worden. Jedoch kann der hier vorgestellte Ansatz einfach erweitert werden, wenn alle Betreiber auf Basis des Domänenmetamodells ihre Managementdienste entwerfen. Dies stellt einen ersten gemeinsamen Ausgangspunkt dar, um ein über Betreibergrenzen hinweg einheitliches Dienstverzeichnis zu entwickeln. Auf der Grundlage der vorgestellten Methode können die Inhalte dieses Dienstverzeichnisses – die Managementbasisdienste

und Managementprozessdienste – von den jeweiligen Betreibern entworfen oder aufgefunden werden. Darüber hinausgehend sind aber weitere Fragen zu klären, die vor allem das Verhältnis zwischen den Nutzern und Anbietern von Managementdiensten betreffen.

Durchgehende Modelltransformationen

Der Entwurf verteilter Softwaresysteme erfordert in der Regel eine Vielzahl beteiligter Akteure. Zur Ausblendung von zunächst unwichtigen Details für eine spätere Lösung werden verschiedene Modellabstraktionen vorgenommen, die von den Experten eines jeweiligen Abstraktionsniveaus verstanden und sukzessive mit Details einer jeweils zugrunde liegenden gedachten Plattform angereichert werden. So wird beispielsweise im Verlauf der vorgestellten Methode zunächst die Erfassung fachfunktionaler Anforderungen durch den Einsatz von dedizierten strukturellen Domänenmodellierungssprachen vorgenommen, um die erstellten Abstraktionen in einem nächsten Schritt auf Modelle von Dienstkandidaten abzubilden. Hierbei findet eine Anreicherung mit Aspekten der zugrunde liegenden dienstorientierten Architektur als gedachte Ausführungsplattform statt. Durch den Einsatz von speziellen Transformationen zwischen Quell- und Zielmodell können die in der vorgestellten Methode spezifizierten manuellen Ableitungsschritte automatisiert werden. Dies kann insgesamt zu einer Reduktion von Kosten in der Entwicklung von Managementdiensten führen, wenngleich der erste Aufwand für die Konstruktion der entsprechenden Modelltransformationen größer ist.

Evolution der erstellten Ontologie

Die auf Basis des vorgestellten Metamodells der Domäne erstellten Modelle zielen auf die Umsetzung von fachlichen Anforderungen für konkrete Anwendungsfälle ab. Zur Reduktion wiederkehrender Schritte sowie als Beitrag im Bemühen, ein über die Grenzen von Betreibern einheitliches Begriffsverständnis innerhalb des Problembereiches zu erschaffen, wurde eine Ontologie für die Störungsbearbeitung vorgestellt. Diese Ontologie kann als Ausgangspunkt genutzt werden, um in konkreten Entwicklungsprojekten die Modellierung von fachfunktionalen Anforderungen zu vereinfachen. Beispielsweise können auf der Grundlage der Ontologie zur Störungsbearbeitung verschiedene Muster innerhalb des *Incident-Managemen*-Prozesses vorab erstellt werden, damit diese Muster als Schablone zur Abbildung komplexer Abläufe genutzt werden können. Insgesamt ist die vorgestellte Ontologie auf der Grundlage einer Analyse des Standards ISO/IEC20000-1:2005 entstanden. Da davon auszugehen ist, dass im Laufe der Zeit nicht nur der Standard an sich, sondern auch verschiedene Aspekte innerhalb der Domäne eine Überarbeitung erfahren werden, ist davon auszugehen, dass die vorgestellte Ontologie in der jetzigen Form an die neuen Gegebenheiten angepasst werden muss. Durch das offene Konzept der vorgestellten Methode kann eine Überarbeitung der jeweiligen spezifizierten Modelle erfolgen, wenn in weiteren Betrachtungen die hierfür notwendigen Schritte nachvollziehbar in die Gesamtmethode eingebettet werden.

Anhänge

A. Glossar

Begriff	Begriffserklärung
Analysephase	Die Analysephase stellt in einem Softwareentwicklungsprozess die erste Phase dar und hat die von der zu konstruierenden Softwarelösung nötigen Anforderungen (funktionale, nichtfunktionale Anforderungen) zu erfassen und darzustellen zum Ziel.
Anforderung	Eine Anforderung ist ein Merkmal, das ein System haben muss, oder eine Bedingung, die erfüllt sein muss, um vom Auftraggeber einer zu entwerfenden Anwendung akzeptiert zu werden [BD09].
Anwendungsdienst	Ein Anwendungsdienst (*application service*) ist ein Geschäftsdienst oder ein Teil davon, der mittels IT von der Anwendungslandschaft erbracht wird [EH+08]. Im Kontext der vorliegenden Arbeit sind Anwendungsdienste diejenigen Dienste, die eine Managementfunktionalität durch entsprechende IT-Komponenten realisieren.
Anwendungsdomäne	Unter einer Anwendungsdomäne, häufig auch verkürzt Domäne, versteht man in der Informatik, und insbesondere in der Softwaretechnik, ein abgrenzbares Problemfeld des täglichen Lebens oder – etwas spezieller – einen bestimmten Einsatzbereich für Computersysteme oder Software.
Architektur	Gesamtheit der Strukturen eines Systems, die die einzelnen Bausteine, deren extern sichtbare Eigenschaften sowie die Beziehungen und Interaktionen zwischen diesen umfassen [PM06].
Betreiberprozess	In dieser Arbeit verwendetes Synonym für Managementprozess.
Dienst	Eine strukturierte Sammlung von Fähigkeiten um eine Anwendung zu unterstützen [ITU00]. Ein Dienst definiert Funktions- bzw. Leistungsbereitschaft und verschattet die Komplexität der Realisierung der Eigenschaften [Ma99]. *A service is an offer of value to another through a well-defined interface and available to a community (which may be the general public). A service results in work provided to one by another* [OMG-SoaML].

	Service is defined as an offer of value to another party, enabled by one or more capabilities. Here, the access to the service is provided using a prescribed contract and is exercised consistent with constraints and policies as specified by the service contract. A service is provided by a participant acting as the provider of the service-for use by others. The eventual consumers of the service may not be known to the service provider and may demonstrate uses of the service beyond the scope originally conceived by the provider [OASIS-SOA-RM]
Domäne	*A domain ontology explains the types of things in that domain* [GD+09]. Eine Domäne beschreibt ein begrenztes Wissens- oder Interessensgebiet [SV+07]. Domänen gruppieren die Komponenten einer Anwendungslandschaft. Die Gruppierung erfolgt nach fachlichen Gesichtspunkten. Domänen können hierarchisch geschachtelt sein. Komponenten der Anwendungslandschaft werden jeweils der tiefsten geschachtelten Domäne zugeordnet [EH+08].
Domänenanalyse	Domänenanalyse bezeichnet das Vorgehen zur Identifizierung und Organisation von Wissen bezüglich eines Klasse von Problemen, mit dem Ziel, eine Beschreibung und eine Lösung dieser Probleme zu unterstützen [PA91].
Domänenmodell	Modell einer Domäne. Siehe hierzu auch Modell und Domäne.
Entwurfsparadigma	Entwurfsparadigmen sollen helfen, Lösungen für eine Aufgabenstellung in einem Anwendungsbereich zu finden und in einen realisierbaren Entwurf zu übertragen. Sie sind eine Art konzeptionelle Anleitung für die Softwareentwicklung [RP06].
Fachfunktionales Modell	Modellaspekt eines Domänenmodells, das die Erfassung fachfunktionaler Anforderungen aus der Analyse von Prozessen unterstützt.
Geschäftsdienst	Ein Geschäftsdienst ist ein Element geschäftlichen Verhaltens. Er stellt eine geschäftliche Leistung dar, die ein Dienstgeber gegenüber einem Dienstnehmer erbringt. Jedem Geschäftsdienst liegt ein Vertrag zugrunde [EH+08].

	Ein Geschäftsdienst beschreibt im Kontext der vorliegenden Arbeit eine fachlich-motivierte Außenansicht auf ein System, die nicht zwangsläufig direkt von IT erbracht werden muss. Konkret sind die Geschäftsdienste in der betrachteten Domäne IT-Management diejenigen Managementdienste, die direkt von Kunden eines IT-Dienstleisters genutzte Managementfunktionalität bereitstellen, beispielsweise die Funktionalität, die von einem *Service Desk* erbracht wird.
Integration	Schaffen eines Verbundes von mehreren Informationssystemen mit dem Ziel, Daten der einzelnen Systeme austauschen zu können, um hierdurch neue oder geänderte Anforderungen erfüllen zu können [Li03].
IT-Management	Gesamtheit aller Maßnahmen für einen unternehmenszielorientierten effektiven und effizienten Betrieb eines verteilten Systems mit seinen Ressourcen [HA+99].
Komponente	Eine Softwarekomponente ist ein Softwareelement, das konform zu einem Komponentenmodell ist und gemäß einem Kompositionsstandard ohne Änderungen mit anderen Komponenten verknüpft und ausgeführt werden kann [CH01].
Management	*Management is related to activities which control or monitor the use of resources* [ISO89].
Managementarchitektur	Das Rahmenwerk für managementrelevante Standards in Bezug auf die genannten Aspekte wird Managementarchitektur genannt [HA+99].
Managed Object	*The OSI Management view of a resource within the OSI Environment that may be managed through the use of OSI management protocols* [ISO89].
Managementplattform	Plattformen sind Trägersysteme für Managementanwendungen. Eine Managementplattform stellt Grundfunktionen zur Realisierung von Managementanwendungen zur Verfügung. Folgende Aspekte müssen herstellerübergreifend spezifiziert sein: - Beschreibung von Ressourcen : Informationsmodell - Behandlung und Unterstützung von Organisationsaspekten, Rollen und Kooperationsformen: Organisationsmodell - Beschreibung von Kommunikationsvorgängen zu

	Managementzwecken: Kommunikationsmodell - Strukturierung der Managementfunktionalität: Funktionsmodell
Managementprozess	Ein Managementprozess ist eine spezielle Art eines Geschäftsprozesses mit der Zielsetzung, überwachend oder steuernd auf ein *Managed Object* einzuwirken [Bl91]. Aktivitäten eines Managementprozesses unterstützen Manager in der Planung, Organisation, Überwachung, Kontrolle und Abrechnung von verketteten Diensten [ISO89].
Managementsystem	Ein funktionales System, das das Management von vernetzten Systemen und deren Ressourcen unterstützt, um einen Dienst sicher betreiben zu können [ITU00]. Managementsysteme sind die Hardwarekomponenten, Software-anwendungen und Prozeduren, die benötigt werden um Netzwerk und Systemressourcen zu überwachen, kontrollieren, betreiben, koordinieren, planen, administrieren, berichten und abzurechnen [Gh97].
Managementwerkzeug	Ein Managementwerkzeug ist ein Softwareartefakt, das eine oder mehrere Aktivitäten eines Managementprozesses realisiert.
Metamodell	Ein Metamodell legt die Struktur und die Bedeutung von Elementen eines Modells fest [SV+07].
Modell	Ein Modell stellt ein Abbild eines realen oder abstrakten Systems dar, das Analogien aufweist und durch einen oder mehrere Aspekte eine Abstraktionsbeziehung zum Ausgangsobjekt besitzt [PM06]. Ein Modell ist (...) eine Sammlung von Daten, die einer bestimmten Struktur genügen und die eine bestimmte Bedeutung haben [SV+07].
Ontologie	Eine Ontologie stellt die Spezifikation einer Konzeption dar [Gr93].
Referenzmodell	Ein Referenzmodell stellt – im Sinne einer ersten begrifflichen Annäherung – für die Entwicklung spezifischer Modelle einen Bezugspunkt dar, da es eine Klasse von Anwendungsfällen

	repräsentiert.
Softwarearchitektur	Systematische Einteilung eines Systems in Elemente, deren Schnittstellen, die Prozesse und Abhängigkeiten zwischen ihnen sowie die benötigten Ressourcen [RP06].
Vorgehensmodell	Rahmenkonzept zur Softwareentwicklung, angelehnt an den Lebenszyklus von Software. Dient der zeitlichen und logischen Strukturierung der durchzuführenden Aktivitäten innerhalb der Entwicklungs- und Einsatzphasen von Informationssystemen [TL+09].
Werkzeugmodell	Modellaspekt eines Domänenmodells, das die Managementfähigkeiten eines bestehenden Managementwerkzeuges darstellt.
Wiederverwendung	Bezeichnet den Vorgang, neue oder geänderte Anforderungen an ein Softwaresystem auf Basis bestehender Elemente dieses Systems umzusetzen. Wiederverwendung setzt somit voraus, dass eine möglichst optimale Wiederverwendbarkeit besteht.
Wiederverwendbarkeit	Definiert ein Maß, das angibt, unter welchen Bedingungen ein Softwareartefakt wiederverwendet werden kann.

B. Abkürzungen

Abkürzung	Langbezeichnung
ATIS	*Abteilung technische Infrastruktur* Dienstleister der Institute der Fakultät für Informatik am Karlsruher Institut für Technologie.
BPMN	*Business-Process Model and Notation* Modellierungssprache zur formalisierten Darstellung von geschäftlichen Abläufen.
C&M	*Cooperation & Management* Name einer am Karlsruher Institut für Technologie angesiedelten Forschungsgruppe.
CIM	*Common Information Model* Informationsmodell der DMTF, das die Grundlage für die Beschreibung von Managementinformation der WBEM-Architektur bildet.
CIM	*Computation Independent Model* Abstraktionsgrad innerhalb der MDA, der die systemunabhängige Formulierung von Anforderungen an eine Softwarelösung beschreibt.
CORBA	*Common Objects Request Broker Architecture* Objektorientierter Ansatz zur Gestaltung von verteilten Anwendungen.
DMTF	*Desktop Management Task Force* Zusammenschluss mehrerer Unternehmen mit dem Ziel, einen einheitlichen Satz an Managementfunktionen und unterstützender Managementinformation zu definieren.

DLV *Dienstleistungsvereinbarung*

Deutsches Synonym für SLA.

ISO *International Organization for Standardization*

Internationaler Zusammenschluss nationaler Normierungsorganisationen.

IT Informationstechnik (engl. *Information Technology*)

Bezeichnet die Gesamtheit aller Technologien zur Verarbeitung von Information unter Zuhilfenahme von Hardware und Software.

ITIL *Information Technology Infrastructure Library*

Rahmenwerk zur Spezifikation von ITSM-Prozessen in Form von unverbindlichen *Best Practices.*

ITSM *Information Technology Service Management*

Gesamtheit aller Verfahren zur Überwachung und Steuerung aller Aspekte von IT-Diensten.

ITU *International Telecommunication Union*

Organisation der Vereinten Nationen zur Regelung von globaler Informations- und Kommunikationstechnik-bezogenen Fragen.

OMG *Object Management Group*

Konsortium zur Standardisierung von Ansätzen zur Modellierung wie UML und Spezifikation von Middleware-Ansätzen wie CORBA.

OSI *Open Systems Interconenction*

Ansatz zur Standardisierung vernetzer Systeme, vorangetrieben durch die ISO und durch die ITU.

MDA *Model-driven Architecture*

Ansatz der OMG zur Entwicklung von Softwaresystemen, indem Modelle eines hohen Abstraktiongrades in Modelle eine niedrigeren Abstraktiongrades transformiert werden.

MDSD *Model-driven Software Development*

Modellgetriebene Softwareentwicklung ist ein Oberbegriff für Technicken, die aus formalen Modellen automatisiert lauffähige Software erzeugen.

MOF *Meta Object Facility*

Standard der Object Management Group (OMG) zur Vereinheitlichung der Definition von Metadaten als Grundlage für den Austausch von Datenmodellen. MOF ist Grundlage zur Spezifikation der Metamodelle der UML.

MOF *Managed Object Format*

Ansatz der *Distributed Management Task Fore* (DMTF), eine Syntax zur Definition von CIM-Modellen festzulegen.

MOF *Microsoft Operations Framework*

Rahmenwerk der Firma Microsoft zur Spezifikation eines prozessorientierten Ansatzes für den qualitätsgesicherten Betrieb von IT-Diensten und IT-Infrastrukturen.

MOWS *Management of Web service*

Standardpezifikation der OASIS für funktionale Anforderungen zum Betrieb von Webservices.

MUWS *Management using Web services*

Standardspezifikation der OASIS für funktionale Anforderungen zum Einsatz von Webservices für den Betrieb von IT.

SNMP *Simple Network Management Procotol*

Protokollspezifikation für den verteilten Betrieb vernetzter System- und

Anwendungskomponenten.

UML *Unified Modeling Language*

Standardisierte Modellierungssprache zur Beschreibung statischer und dynamischer Aspekte beim Entwurf von komplexen Systemen.

UMPS *UML Profile and Metamodel for Services*

UML-Profil der OMG zur Modellierung von Aspekten einer dienstorientierten Architektur. Das UPMS ist Ausgangspunkt und wesentlicher Meilenstein für die Entwicklung der SoaML.

PIM *Platform Independent Model*

Abstraktionsgrad innerhalb der MDA, der die plattformunabhängige Modellierung bezeichnet.

PSM *Platform Specific Model*

Abstraktionsgrad innerhalb der MDA, der die plattformspezifische Modellierung bezeichnet.

SOA *Service-oriented Architecture*

Architektonischenes Muster zur Gestaltung eines verteilten Systems, bei dem die wesentlichen Elemente aus (Software-)Diensten bestehen.

SOMA *Service-oriented Modeling and Architecture*

Ein von der Firma IBM konzipiertes Vorgehensmodell, um die Modellierung von Diensten in einem integrierten Entwicklungsvorgehen in den Vordergrund zu stellen.

SLA *Service Level Agreement*

Vereinbarung mit vertraglichem Charakter zwischen einem Dienstnehmer

	und einem Dienstleister. Der Gegenstand eines Service Level Agreements ist die Festlegung von Dienstfunktionalität verknüpft mit qualitativem Aspekt.
SLM	*Service Level Management* Einer der zentralen Betreiberprozesse, um die Qualitätsgüte von IT-Diensten sicherzustellen. Gegenstand des SLM ist u. a. die Aushandlung von SLA oder die Überwachung von Infrastrukturressourcen zur Einhaltung von SLAs.
WBEM	*Web-based Enterprise Management* Verteilte Managementarchitektur der DMTF.
WSDL	*Web Service Description Language* Vom W3C standardisierte Syntax zur Definition von Schnittstellenbeschreibung von Webservices.
XMI	*eXtensible Markup Language Metadata Interchange* Konkrete Syntax zur Beschreibung von Information für den plattformunabhängigen Austausch.
XML	*eXtensible Markup Language* Konkrete Syntax zur strukturierten Auszeichnung von Information.

C. Index

D. Abbildungsverzeichnis

E. Tabellenverzeichnis

F. Quelltextverzeichnis

G. Literaturverzeichnis

[AB+07] N. Ayachitula, M. Buco, Y. Diao, B. Fisher, D. Loewenstern, C. Ward: IT Service Management Automation – An Automation Centric Approach Leveraging Configuration Control, Audit Verification and Process Analytics, in Lecture Notes in Computer Science, Vol. 4785/2007, pp. 195-198, 2007.

[AG+08] A. Arsanjani, S. Gosh, A. Allam, T.Abdollah, S. Ganapathy, K. Holley: SOMA : A method for developing service oriented solutions, in IBM Systems Journal, Vol. 47 (3), pp. 377-396, 2008.

[AH01] G. Aschemann, P. Hasselmeyer: A Loosely Coupled Federation of Distributed Management Services, Journal of Network and Systems Management, Vol 9, No. 1, 2001.

[AL+03] S. Abeck, P. C. Lockemann, J. Schiller, J. Seitz: Verteilte Informationssysteme – Integration von Datenübertragungstechnik und Datenbanktechnik, Dpunkt Verlag, 2003.

[Am10] J. Amsen: Modeling with SoaML, the Service-Oriented Architecture Modeling Language – Part 1- Service Identification, IBM developerWorks 2010.

[An99] N. Anerousis: An Architecture for Building Scalable, Web-Based Management Services, Journal of Network and System Management, Vol. 7, No 1., 1999.

[Ar04] A. Arsanjani: “Service-oriented modeling and architecture”, 2004.

[Ar89] G. Arango: Domain Analysis - From Art Form To Engineering Discipline, ACM SIGSOFT Software Engineering Notes, Volume 14, Issue 3, 1989.

[AS11] W. van der Aalst and C. Stahl: Modeling Business Processes - A Petri Net-Oriented Approach, MIT Press, 2011.

[AZ+06] Uwe Aßmann, Steffen Zschaler , Gerd Wagner: Ontologies, Meta-models and the Model-Driven Paradigm, Ontologies for Software Engineering and Software Technology (2006), pp. 249-273, 2006.

[Ba00] Balzert: Lehrbuch der Softwaretechnik, Spektrum Akademischer Verlag, 2000.

[BB+06] N. Bieberstein, S. Bose, M. Fiammante, K. Jones, R. Shah: Service-Oriented Architecture Compass: Business Value, Planning, and Enterprise Roadmap, Prentice Hall, 2006.

[BD+07] Jörg Becker, Patrick Delfmann, Tobias Rieke: Effiziente Softwareentwicklung mit Referenzmodellen, Physica-Verlag, Auflage: 1, 2007.

[BD09] Bernd Bruegge, Allen H. Dutoit: Object-Oriented Software Engineering Using UML, Patterns and Java, Pearson Education, 2009.

[Be04] A. E. Bell: Death by UML Fever, ACM Queue, Volume 2 Issue 1, March 2004, 2004.

[BK06] A. B. Brown, A. Keller: A Best Practice Approach for Automating IT Management Processes, Proceedings of 10th IEEE/IFIP Network Operations and Management Symposium (NOMS) 2006, pp. 3-44, 2006.

[BK11] C. Baun, M. Kunze, J. Nimis, S. Tai: Cloud Computing – Web-basierte dynamische IT-Services, Springer-Verlag, 2011.

[Bl91] U. D. Black: OSI: a model for computer communications standards, Prentice-Hall, 1991.

[Br06] M. Brenner: Classifying ITIL Processes – A Taxonomy under Tool Support Aspects, Proceedings of First IEEE/IFIP International Workshop on Business–Driven IT Management (BDIM 06), pp. 19–28, 2006.

[Ca03] Nicholas Carr: IT doesn´t matter, Harvard Business Review, 2003; http://www.roughtype.com/archives/2007/01/it_doesnt_matte.php (zuletzt besucht: 19-02-2011).

[CH+05] S. Conrad, W. Hasselbring, A. Koschel, R. Tritsch: Enterprise Application Integration: Grundlagen – Konzepte – Entwurfsmuster – Praxisbeispiele, Spektrum Akademischer Verlag, 2005.

[CH01] W. T. Councill, G. T. Heineman: Component-Based Software Engineering. Addison-Wesley, 2001.

[CJ+99] B. Chandrasekaran, J. R. Josephson, V. R. Benjamins: What are Ontologies, and why do we need them?, IEEE Intelligent Systems and their Applications, Vol. 14, Issue 1, pp. 20-26, 1999.

[CK08] S. W. Choi, S. D. Kim: A Quality Model for Evaluating Reusability of Services in SOA, 10th IEEE Conference on E-Commerce Technology and the 5th IEEE Conference on Enterprise Computing, E-Commerce and Services, pp. 293-298, 2008.

[CN04] G. Cernosek, E. Naiburg: The Value of Modeling, IBM Developer Works Whitepaper, 2004.

[Co07] S. Cohen: Ontology and Taxonomy of Services in a Service-oriented Architecture, Microsoft Architecture Journal Vol. 11, 2007.

[Da06] V. Danciu: Formalism for IT Management Process Representation, Proceedings of First IEEE/IFIP International Workshop on Business–Driven IT Management (BDIM 06), April, 2006.

[De02] V. Devedzic: Understanding Ontological Engineering, Communications of the ACM, Vol. 45 (4), pp. 136-144, 2002.

[De09] M. Deeg: Modellierung von Serices mit der SoaML, OBJEKTspektrum, Ausgabe SOA/2009, 2009.

[DJ+05] W. Dostal, M. Jeckle, I. Melzer, B. Zengler: Service-orientierte Architekturen mit Web Services, Elsevier, Spektrum, Akad. Verl., 2005.

[DK06] Y. Diao, A. Keller: Quantifying the Complexity of IT Service Management Processes., Large Scale Management of Distributed Systems, 17th IFIP/IEEE International Workshop on Distributed Systems: Operations and Management, DSOM 2006, Dublin, Ireland, October 23-25, 2006.

[DK+07] Y. Diao, A. Keller, S. Parekh, V. V. Marinov: Predicting Labor Cost through IT Management Complexity Metrics, Integrated Network Management, IM 2007. 10th IFIP/IEEE International Symposium on Integrated Network Management, Munich, Germany, 2007.

[DMTF-CIM] Distributed Management Task Fore (DMTF): Common Information Model (CIM) Version 2.28.0, 2011.

[DMTF-WS-Management] Distributed Management Task Force (DMTF): Web Services for Management (WS-Management) Specification, Version 1.1.0, DMTF, 2010.

[DSB09] Statistisches Bundesamt, Informationsgesellschaft in Deutschland Ausgabe 2009, Statistisches Bundesamt, Wiesbaden, 2009.

[Du10] Dudenverlag: Die deutsche Rechtschreibung, Band 1, 25. Auflage, 2010.

[EH+08] G. Engels, A. Hess, B. Humm, O. Juwig, M. Lohmann, J.-P. Richter, M. Voß, J. Willkomm: Quasar Enterprise: Anwendungslandschaften serviceorientiert gestalten, dpunkt.verlag, 2008.

[EK+07] C. Emig, K. Krutz, S. Link, C. Momm, S. Abeck: Model-Driven Development of SOA Services, C&M Research Report, Universität Karlsruhe (TH), 2007.

[EL+06] C. Emig, K. Langer, K. Krutz, S. Link, C. Momm, S. Abeck: The SOA‘s Layers, C&M Research Report, Universität Karlsruhe (TH), 2006.

[Em08] C. Emig: Zugriffskontrolle in dienstorientierten Architekturen, Dissertation an der Universität Karlsruhe (TH), 2008.

[Er06] T. Erl: Service-oriented architecture: concepts, technology, and design, Prentice Hall, 2006.

[Er08] T. Erl: SOA: Principles of Service Design, Prentice Hall, 2008.

[Er08b] T. Erl: SOA Design Patterns, Prentice Hall, 2009.

[Er09] T. Erl: Web Service Contract Design and Versioning for SOA, Prentice Hall, 2009.

[Fa08] N. Fareghzadeh: Service Identification Approach to SOA Development, PWASET (Proceedings of World Academy Of Science, Engineering And Technology) Volume 35, 2008.

[FE10] B. Furth, A. Escalante: Handbook of Cloud Computing, Springer-Verlag, 2010.

[FG+02] R. de Almeida Falbo, G. Guizzardi, K. C. Duarte: An Ontological Approach to Domain Engineering, Proceedings of the 14th international conference on Software engineering and knowledge engineering (SEKE2002), 2002.

[FV99] X. Ferré, S.Vegas: An Evaluation of Domain Analysis Methods, In 4th CAiSE/IFIP8.1 International Workshop in Evaluation of Modeling Methods in Systems Analysis and Design (EMMSAD99), 1999.

[GD+09] Dragan Gašević, Dragan Djurić, Vladan Devedžić: Model-Driven Engineering and Ontology Development, Springer, 2009.

[GD+09b] S. D. Galup, R. Dattero, J. J. Quan, S. Conger: An Overview of IT Service Management, Communications of the ACM, Vol. 52 (5), pp. 124-127, 2009.

[Ge11] M. Gebhart: Qualitätsorientierter Entwurf von Anwendungsdiensten, Dissertation am Karlsruher Institut für Technologie (KIT), 2011.

[Gh97] I. G. Ghetie: Network and Systems Management – Platform Analysis and Evolution, Kluwer Academic Publications, 1997.

[GH+08] E. Gamma, R. Helm, R. Johnson, J. Vlissides: Design Patterns: elements of reusable object-oriented software, Addison-Wesley, 2008.

[GP+08] R. Gupta, K. H. Prasad, M. Mohania: Automating ITSM Incident Management Process, Proceedings of the 2008 International Conference on Autonomic Computing, pp. 141-150, 2008.

[GQ+07] S. D. Galup, R. Dattero, J. J. Quan, S. Conger: Information Technology Service Management: an emerging area for academic research and pedagogical development, Proceedings of the 2007 ACM SIGMIS CPR conference on Computer personnel research: The global information technology workforce, pp.52, 2007.

[Gr93] T. Gruber: A translation approach to portable ontology specifications, Knowledge Acquisition, Vol. 5, Issue 2, pp. 199-220, 1993.

[HA+99] H.-G. Hegering, S. Abeck, B. Neumair: Integriertes Management vernetzter Systeme, dpunkt.verlag, 1999.

[Ha09] M. Hamm: Eine Methode zur Spezifikation der IT-Service-Managementprozesse Verketteter Dienste, Dissertation an der Ludwig-Maximilians-Universität München, 2009.

[He08] M. Henning: The Rise and Fall of CORBA, Communications of the ACM Vol. 51 No. 8, pp. 52-57, 2008.

[HG+09] P. Hoyer, M. Gebhart, I. Pansa, S. Link, A. Dikanski, S. Abeck: A Model-Driven Development Approach for Service-Oriented Integration Scenarios, COMPUTATION WORLD 09: Future Computing, Service Computation, Cognitive, Adaptive, Content, Patterns, 2009.

[HG+10] P. Hoyer, M. Gebhart, I. Pansa, S. Link, A. Dikanski, S. Abeck: Service-Oriented Integration Using a Model-driven Approach, International Journal On Advances in Software, Vol. 3, 2010.

[HL+08] J. S. Her, H. J. La, S. D. Kim: A Formal Approach to Devising a Practical Method for Modeling Reusable Services, 2008 IEEE International Conference on e-Business Engineering, pp. 221-228, 2008.

[HT06] B. Hailpern, P. Tarr: Model-driven Engineering: The Good, The Bad and the Ugly, IBM Systems Journal, Vol. 45, No. 3, 2006.

[IBM-RSA] IBM: Rational Software Architect, Version 7.5.

[ISO89] ISO/IEC 7498-4:1989 : Information processing systems – Open Systems Interconnection – Basic Reference Model – Part4: Management framework, International Standards Organization (ISO), 1989.

[ISO05] ISO/IEC20000-1:2005 : Information technology – Service management – Part1: Specification, International Standards Organization (ISO), 2005.

[ISO05b] ISO/IEC20000-2:2005 : Information technology – Service management – Part2: Code of Practice, International Standards Organization (ISO), 2005.

[ISO09] ISO/IEC20000-3:2009 : Information technology – Service management – Part3: Guidance on scope definition and applicability of ISO/IEC20000-1, International Standards Organization (ISO), 2005.

[ITU00] International Telecommunication Union: SERIES Y: GLOBAL INFORMATION INFRASTRUCTURE AND INTERNET PROTOCOL ASPECTS Global information infrastructure – General: Global Information Infrastructure terminology: Terms and definitions, ITU, 2000.

[IW+94] K. Inoue, A. Watanabe, K. Torii: Modeling Method for Management Process and Its Application to CMM and ISO9000-3, Proceedings of The Third International Conference on The Software Process, pp.85-98, 1994.

[KB+05] D. Krafzig, K. Banke, D. Slama: Enterprise SOA: Service Oriented Architecture Best Practises, Prentice Hall International, 2005.

[Ke98] A. Keller: CORBA-basiertes Enterprise Management: Interoperabilität und Managementinstrumentierung verteilter kooperativer Managementsysteme in heterogener Umgebung, Dissertation an der Technischen Universität

München, 1998.

[KK+06] G. Kappel, E. Kapsammer, H. Kargl, G. Kramler, T. Reiter, W. Retschitzegger, W. Schwinger, M. Wimmer: Lifting Metamodels to Ontologies, Lecture Notes in Computer Science, Volume 4199/2006, pp. 528-542, 2008.

[Kr92] C.W. Krueger: "Software reuse", *ACM Computing Surveys (CSUR)*, vol. 24, no. 2, pp. 131-183, 1992.

[Kr95] P. Kruchten: The 4+1 View Model of Architecture, IEEE Software, vol. 12, no. 6, pp. 42-50, 1995.

[Ku05] P. Kumar: Web Services and IT Management, ACM Queue, Volume 3 Issue 6, 2005.

[Le11] C. Leist: Entwurf wiederverwendbarer Managementdienste auf der Basis von SoaML, Studienarbeit, Karlsruher Institut für Technologie (KIT), C&M (Prof. Abeck), 2011.

[Li03] D. S. Linthicum: Next Generation Application Integration: From Simple Information to Web Services, Addision Wesley Information Technology, 2003.

[Li09] S. Link: Benutzerinteraktion in dienstorientierten Architekturen, Dissertation an der Universität Karlsruhe (TH), 2009.

[LL+08] Z. Lu, Y. Li, J. Wu, S. Thang, Y. Zhong: SOMAS: A Novel SOA-based System and Network Management Model and Scheme, IEEE International Conference on Services Computing, pp. 479 – 480, 2008.

[LW+08] Z. Lu, Y. Wu, C. Xiao, S. Zhang, Y. Zhong: WSDSNM3: A Web Services-based Distributed System and Network Management Middleware Model and Scheme, The 9th International Conference for Young Computer Scientists, pp. 392 – 397, 2008.

[LW+09] Z. Lu, J. Wang, Y. Wu, J. Wu, Y. Zhong: MWS-MCS: A Novel Multi-agent-assisted and WS-Management-based Composite Service Management Scheme, IEEE International Conference on Web Services, pp. 1041 – 1042, 2009.

[LW+09b] Z. Lu, J. Wu, S. Zhang, Y. Zhong: Research on WS-Management-based System and Network Resource Management Middleware Model, IEEE International Conference on Web Services, pp. 1051 – 1053, 2009.

[Ma99] C. Mayerl: Eine integrierte Dienstmanagement-Architektur für die qualitätsgesicherte Bereitstellung von Netz- und Systemdiensten, Dissertation an der Universität Karlsruhe, 1999.

[MantisBT] Mantis Bug Tracker, 2011; http://www.mantisbt.org/ (zuletzt besucht: 19-02-2011).

[MB+04] V. Machiraju, C. Bartolini, F. Casati: Technologies for Business-Driven IT Management, in Extending Web Services Technologies: the Use of Multi-Agent Approaches, Kluwer Academic, 2004.

[Me06] O. Mehl: Modellgetriebene Entwicklung managementfähiger Anwendungssysteme, Dissertation an der Universität Karlsruhe (TH), 2006.

[Mo09] C. Momm: Modellgetriebene Entwicklung überwachter Webservice-Kompositionen, Dissertation an der Universität Karlsruhe (TH), 2009.

[MS+07] A. Moura, J. Sauve, C. Bartolini: Research Challenges of Business-Driven IT Management, 2nd IEEE/IFIP International Workshop on Business-Driven IT Management, pp.19-28, 2007.

[MSSP] Microsoft Sharepoint 2003; http://sharepoint.microsoft.com/de-at/Seiten/default.aspx (zuletzt besucht: 19-02-2011).

[MT+06] C. Mayerl, F. Tröscher, S. Abeck: Process-oriented Integration of Applications for a Service-oriented IT Management, The First International Workshop on Business-Driven IT-Management, BDIM'06, pp. 29-36, 2006.

[MV+05] C. Mayerl, T. Vogel, S. Abeck: SOA-based Integration of IT Service Management Applications, 2005 IEEE International Conference on Web Services (ICWS 2005), 2005.

[Ne80] Jim Neighbors: Software Construction using Components, doctoral dissertation, Information and Computer Science Dept., Univ. of California, Irvine, 1980.

[No09] S. Nolte: QVT - Relations Language: Modellierung mit der Query Views Transformation, Xper.Press, 2009.

[OASIS-UDDI] Organization for the Advancement of Structured Information Standards (OASIS): Universal Description, Discovery and Integration (UDDI),

Version 3.0.2, Oktober 2004.

[OASIS-MUWS] Organization for the Advancement of Structured Information Standards (OASIS): Web Services Distributed Management: MUWS Primer, OASIS, Februar 2006.

[OASIS-SOA-RA] Organization for the Advancement of Structured Information Standards (OASIS): Reference Architecture for Service Oriented Architecture, Version 1.0, OASIS, April 2008.

[OASIS-SOA-RM] Organization for the Advancement of Structured Information Standards (OASIS): Reference Model for Service Oriented Architecture, Version 1.0, OASIS, August 2006.

[OASIS-WSBPEL] Organization for the Advancement of Structured Information Standards (OASIS): Web Services Business Process Execution Language (WS-BPEL), Version 2.0, OASIS, April 2007.

[OGC07] Office of Government Commerce (OGC): “Service Design”, TSO, 2008.

[OGC07b] Office of Government Commerce (OGC): “Service Operation”, TSO, 2008.

[OGC07c] Office of Government Commerce (OGC): “Service Transition”, TSO, 2008.

[OGC08] Office of Government Commerce (OGC): “Continual Service Improvement”, TSO, 2008.

[OGC08b] Office of Government Commerce (OGC): “Service Strategy”, TSO, 2008.

[OMG-BPMN] Object Management Group (OMG): Business Process Model and Notation BPMN, Version 2.0, OMG, Januar 2011.”

[OMG-CORBA] Object Management Group (OMG): Common Object Request Broker Architecture: Core Specification, Version 3.0.3, OMG, März 2004.

[OMG-MDA] Object Management Group (OMG): Model Driven Architecture (MDA) Guide, Version 1.0.1, OMG, Juni 2003.

[OMG-MOF] Object Management Group (OMG): Meta Object Facility (MOF) Core Specification, Version 2.0, OMG, Januar 2006.

[OMG-ODM] Object Management Group (OMG): Ontology Definition Metamodel (ODM). Version 1.0, OMG, Mai 2009.

[OMG-RAS] Object Management Group (RAS): Reuasable Asset Specification, Version 2.2, OMG, November 2005.

[OMG-SoaML] Object Management Group (OMG): Service-oriented architecture Modeling Language (SoaML) - Specification for the UML Profile and Metamodel for Services (UPMS), FTF Beta 1, OMG, April 2009.

[OMG-UML-Super] Object Management Group (OMG): OMG Unified Modeling Language (OMG UML), Superstructure, Version 2.3, OMG, Mai 2010.

[OMG-UML-Infra] Object Management Group (OMG): OMG Unified Modeling Language (OMG UML), Infrastructure, Version 2.3, OMG, Mai 2010.

[OMG-UPMS] Object Management Group (OMG): UML Profile and Metamodel for Services (UPMS), Request for Proposal, OMG, September 2006.

[OMG-XMI] The Object Management Group: MOF 2.0/XMI Mapping Specification, Version 2.1, OMG, September 2009.

[PA91] R. Prieto-Díaz, G. Arango: Domain Analysis and Software Systems Modeling, IEEE Computer Society Press, 1991.

[Pa07] G. Pavlou: On the Evolution of Management Approaches, Frameworks and Protocols: A Historical Perspective, Journal of Network and Systems Managemen, Springer New York, Vol. 15, Issue 4, pp. 425- 445, 2007.

[Pa10] F. Palmen: Integration bestehender Managementwerkzeuge in einen prozessorientierten Betreiberansatz, Diplomarbeit, Karlsruher Institut für Technologie (KIT), C&M (Prof. Abeck), 2010.

[Pi00] M. Pidd: Tools for Thinking – Modeling in Management Science, Wiley, 2000.

[PM06] R. Petrasch, O. Meimberg: Model Driven Architecture Eine praxisorientierte Einführung in die MDA, dpunkt.verlag, 2006.

[Po97] J. Poulin: Measuring Software Reuse, Addison Wesley Publishing Group, 1997.

[PL+11] I. Pansa, L. Leist, M. Reichle, S. Abeck: Designing Reusable Management Services, SERVICE COMPUTATION 2011, erscheint in: The Third International Conferences on Advanced Service Computing, 2011.

[PR+11] I. Pansa, M. Reichle, L. Leist, S. Abeck: A Domain Ontology for Designing

Management Services, SERVICE COMPUTATION 2011, erscheint in: The Third International Conferences on Advanced Service Computing, 2011.

[Protege] Protege, http://protege.stanford.edu

[PW+10] I. Pansa, P. Walter, S. Abeck, K. Scheibenberger: Model-based Integration of Tools Supporting Automatable Management Processes, IEEE/IFIP Network Operations and Management Symposium Workshops (NOMS Wksps), 2010.

[RD05] O. P. Rotaru, M. Dobre: Reusability Metrics for Software Components, The 3rd ACS/IEEE International Conference on Computer Systems and Applications, 2005.

[Re11] M. Reichle: Abbildung automatisierbarer Betreiberprozesse auf eine dienstorientierte Architektur, Diplomarbeit, Karlsruher Institut für Technologie (KIT), C&M (Prof. Abeck), 2011.

[RH06] R. Reussner, W. Hasselbring: Handbuch der Softwarearchitektur, dpunkt.verlag, 2006.

[RP06] P. Rechenberg, G. Pomberger: Informatik-Handbuch, 4. Auflage, Carl Hanser Verlag, 2006.

[SB+07] R. Schmidt, C. Bartsch, R. Oberhauser: Ontology-based representation of compliance requirements for service processes, Proceedings of the workshop on semantic business processes and product lifecycle management (SBPM 2007), 2007.

[SB08] T. Schaff, M. Brenner: On Tool Support for Service Level Management : From requirements to system specifications, 3rd IEEE/IFIP International Workshop on Business-driven IT Management (BDIM 2008), 2008.

[Sc99] R. Schütte: Grundsätze ordnungsmäßiger Referenzmodellierung – Konstruktion konfigurations- und anpassungsorientierter Modelle, Dissertation an der Universität Münster, 1999.

[SM+06] J. Sauvé, A. Moura, M. Sampaio, J. Jornada, E. Radziuk: An Introductory Overview and Survey of Business-Driven IT Management, 1st IEEE/IFIP International Workshop on Business-driven IT Management (BDIM 2006), 2006.

[SS08] A. Sillitti, G Succi: Reuse: From Components to Services, High Confidence Software Reuse in Large Systems (2008), pp. 266-269, 2008.

[SV+07] T. Stahl, M. Völter, S. Efftinge, A. Haase: Modellgetriebene Softwareentwicklung, dpunkt.verlag, 2007.

[SW+10] S. Staab, T. Walter, G. Gröner F. S. Parreiras: Model Driven Engineering with Ontology Technologies, Lecture Notes in Computer Science, Volume 6325/2010, pp. 62-98, 2010.

[TL+09] O. Thomas, K. Leyking, M. Schied: Vorgehensmodelle zur Entwicklung serviceorientierter Softwaresysteme, Business Services: Konzepte, Technologien, Anwendungen. 9. Internationale Tagung Wirtschaftsinformatik, Wien, 25.-27. Februar 2009.

[To06] V. Tosic: The 5 C Challenges of Business-Driven IT Management and the 5 A Approaches to Addressing Them, The First IEEE/IFIP International Workshop on Business-Driven IT Management, 2006.

[TS+06] Marie-Noëlle Terrasse, Marinette Savonnet, Eric Leclercq, and Thierry Grison, George Becker: Do We Need Metamodels AND Ontologies for Engineering Platforms?, Proceedings of the 2006 international workshop on Global integrated model management, 2006.

[TZ05] Gerrit Tamm, Rüdiger Zarnekow: Umsetzung eines ITIL-konformen IT-Service-Support auf der Grundlage von Web-Services, Wirtschaftsinformtik 2005, pp. 647-666, 2005.

[Us10] Thomas Usländer: Service-oriented Design of Environmental Information Systems, Dissertation am Karlsruher Institut für Technologie (KIT), 2010.

[VV+02] J. E. López De Vergara, V. A. Villagrá, J. Berrocal: Semantic Management: advantages of using an ontology-based management information meta-model, Proceedings of the HP Openview University Association Ninth Plenary Workshop (HP-OVUA'2002), 2002.

[VV+03] J. E. López De Vergara, V. A. Villagrá, J. Asensio, J. Berrocal: Ontologies Giving Semantics to Network Management Models, IEEE Network, Vol. 17, pp. 15-21, 2003.

[WY+03] H. Washizaki, H. Yamamoto, Y. Fukazawa: A Metrics Suite for Measuring Reusability of Software Components, Proceedings of the 9th International Symposium on Software Metrics, 2003.

[WY+08] J. Wang, J. Yu, P. Falcarin, Y. Han, M. Morisio: An Approach to Domain-Specific Reuse in Service-Oriented Environments, Proceedings of the 10th

international conference on Software Reuse: High Confidence Software Reuse in Large Systems, 2008.

[W3C-OWL] World Wide Web Consortium (W3C): Web Ontology Language (OWL), W3C Recommendation, 2004.

[W3C-SAWSDL] World Wide Web Consortium (W3C): Semantic Annotations for WSDL and XML Schema, W3C Recommendation, 2007.

[W3C-WSDL] World Wide Web Consortium (W3C): Web Service Description Language (WSDL) Version 2.0 Part1 Core Language, W3C Recommendation, 2007.

[W3C-XML] World Wide Web Consortium (W3C): Extensible Markup Language (XML) 1.0, W3C Recommendation, 2008.

[XL+09] C. Xiao, Z. Lv, S. Zhang: WS-CHMA: A Composite-Pattern Based Hierarchical WS-Management Architecture, Congress on Services - I, pp. 773 – 780, 2009.

H. Artefakte

Dieser Abschnitt enthält die vollständige Spezifikation der im Rahmen der Arbeit erstellten Artefakte.

1 OWL-Ontologie des Domänenmetamodells

Die OWL-Ontologie spezifiziert das konzeptionelle Metamodell der Domäne IT-Management für den Entwurf wiederverwendbarer Managementdienste.

```
<?xml version="1.0"?>


<!DOCTYPE Ontology [
   <!ENTITY xsd "http://www.w3.org/2001/XMLSchema#" >
   <!ENTITY xml "http://www.w3.org/XML/1998/namespace" >
   <!ENTITY rdfs "http://www.w3.org/2000/01/rdf-schema#" >
   <!ENTITY rdf "http://www.w3.org/1999/02/22-rdf-syntax-ns#" >
]>


<Ontology xmlns="http://www.w3.org/2002/07/owl#"
    xml:base="http://www.semanticweb.org/ontologies/2011/1/ITManagement.owl"
    xmlns:rdfs="http://www.w3.org/2000/01/rdf-schema#"
    xmlns:xsd="http://www.w3.org/2001/XMLSchema#"
    xmlns:rdf="http://www.w3.org/1999/02/22-rdf-syntax-ns#"
    xmlns:xml="http://www.w3.org/XML/1998/namespace"
    ontologyIRI="http://www.semanticweb.org/ontologies/2011/1/ITManagement.owl">
   <Prefix name="rdf" IRI="http://www.w3.org/1999/02/22-rdf-syntax-ns#"/>
   <Prefix name="rdfs" IRI="http://www.w3.org/2000/01/rdf-schema#"/>
   <Prefix name="xsd" IRI="http://www.w3.org/2001/XMLSchema#"/>
   <Prefix name="owl" IRI="http://www.w3.org/2002/07/owl#"/>
   <Declaration>
      <Class IRI="#ManagementActivity"/>
   </Declaration>
   <Declaration>
      <Class IRI="#ManagementArea"/>
   </Declaration>
   <Declaration>
      <Class IRI="#ManagementBasicActivity"/>
   </Declaration>
   <Declaration>
      <Class IRI="#ManagementCapability"/>
   </Declaration>
   <Declaration>
      <Class IRI="#ManagementComposedActivity"/>
   </Declaration>
   <Declaration>
      <Class IRI="#ManagementEntity"/>
   </Declaration>
   <Declaration>
      <Class IRI="#ManagementEntityStructurePolicy"/>
```

```
    </Declaration>
    <Declaration>
        <Class IRI="#ManagementParticipant"/>
    </Declaration>
    <Declaration>
        <Class IRI="#ManagementPolicy"/>
    </Declaration>
    <Declaration>
        <Class IRI="#ManagementProvidedCapability"/>
    </Declaration>
    <Declaration>
        <Class IRI="#ManagementRequiredCapability"/>
    </Declaration>
    <Declaration>
        <ObjectProperty IRI="#contains"/>
    </Declaration>
    <Declaration>
        <ObjectProperty IRI="#defines"/>
    </Declaration>
    <Declaration>
        <ObjectProperty IRI="#hasParticipant"/>
    </Declaration>
    <Declaration>
        <ObjectProperty IRI="#isComposedOf"/>
    </Declaration>
    <Declaration>
        <ObjectProperty IRI="#isDefinedBy"/>
    </Declaration>
    <Declaration>
        <ObjectProperty IRI="#isOperatedBy"/>
    </Declaration>
    <Declaration>
        <ObjectProperty IRI="#isPartOf"/>
    </Declaration>
    <Declaration>
        <ObjectProperty IRI="#isRequiredBy"/>
    </Declaration>
    <Declaration>
        <ObjectProperty IRI="#isUsedBy"/>
    </Declaration>
    <Declaration>
        <ObjectProperty IRI="#operatesOn"/>
    </Declaration>
    <Declaration>
        <ObjectProperty IRI="#participatesIn"/>
    </Declaration>
    <Declaration>
        <ObjectProperty IRI="#requires"/>
    </Declaration>
```

```
<Declaration>
  <DataProperty IRI="#hasDescription"/>
</Declaration>
<Declaration>
  <DataProperty IRI="#hasIdentifier"/>
</Declaration>
<EquivalentClasses>
  <Class IRI="#ManagementActivity"/>
  <ObjectIntersectionOf>
    <ObjectSomeValuesFrom>
      <ObjectProperty IRI="#hasParticipant"/>
      <Class IRI="#ManagementParticipant"/>
    </ObjectSomeValuesFrom>
    <ObjectSomeValuesFrom>
      <ObjectProperty IRI="#operatesOn"/>
      <Class IRI="#ManagementEntity"/>
    </ObjectSomeValuesFrom>
    <ObjectSomeValuesFrom>
      <ObjectProperty IRI="#requires"/>
      <Class IRI="#ManagementCapability"/>
    </ObjectSomeValuesFrom>
    <ObjectAllValuesFrom>
      <ObjectProperty IRI="#hasParticipant"/>
      <Class IRI="#ManagementParticipant"/>
    </ObjectAllValuesFrom>
    <ObjectAllValuesFrom>
      <ObjectProperty IRI="#operatesOn"/>
      <Class IRI="#ManagementEntity"/>
    </ObjectAllValuesFrom>
    <ObjectAllValuesFrom>
      <ObjectProperty IRI="#requires"/>
      <Class IRI="#ManagementCapability"/>
    </ObjectAllValuesFrom>
  </ObjectIntersectionOf>
</EquivalentClasses>
<EquivalentClasses>
  <Class IRI="#ManagementCapability"/>
  <ObjectUnionOf>
    <Class IRI="#ManagementProvidedCapability"/>
    <Class IRI="#ManagementRequiredCapability"/>
  </ObjectUnionOf>
</EquivalentClasses>
<EquivalentClasses>
  <Class IRI="#ManagementComposedActivity"/>
  <ObjectIntersectionOf>
    <ObjectSomeValuesFrom>
      <ObjectProperty IRI="#isComposedOf"/>
      <Class IRI="#ManagementActivity"/>
    </ObjectSomeValuesFrom>
    <ObjectAllValuesFrom>
      <ObjectProperty IRI="#isComposedOf"/>
```

```
            <Class IRI="#ManagementActivity"/>
        </ObjectAllValuesFrom>
    </ObjectIntersectionOf>
</EquivalentClasses>
<EquivalentClasses>
    <Class IRI="#ManagementEntity"/>
    <ObjectIntersectionOf>
        <ObjectSomeValuesFrom>
            <ObjectProperty IRI="#isOperatedBy"/>
            <Class IRI="#ManagementActivity"/>
        </ObjectSomeValuesFrom>
        <ObjectAllValuesFrom>
            <ObjectProperty IRI="#isDefinedBy"/>
            <Class IRI="#ManagementEntityStructurePolicy"/>
        </ObjectAllValuesFrom>
        <ObjectAllValuesFrom>
            <ObjectProperty IRI="#isOperatedBy"/>
            <Class IRI="#ManagementActivity"/>
        </ObjectAllValuesFrom>
        <ObjectExactCardinality cardinality="1">
            <ObjectProperty IRI="#isDefinedBy"/>
            <Class IRI="#ManagementEntityStructurePolicy"/>
        </ObjectExactCardinality>
    </ObjectIntersectionOf>
</EquivalentClasses>
<EquivalentClasses>
    <Class IRI="#ManagementEntityStructurePolicy"/>
    <ObjectIntersectionOf>
        <ObjectAllValuesFrom>
            <ObjectProperty IRI="#defines"/>
            <Class IRI="#ManagementEntity"/>
        </ObjectAllValuesFrom>
        <ObjectExactCardinality cardinality="1">
            <ObjectProperty IRI="#defines"/>
            <Class IRI="#ManagementEntity"/>
        </ObjectExactCardinality>
    </ObjectIntersectionOf>
</EquivalentClasses>
<EquivalentClasses>
    <Class IRI="#ManagementParticipant"/>
    <ObjectIntersectionOf>
        <ObjectSomeValuesFrom>
            <ObjectProperty IRI="#participatesIn"/>
            <Class IRI="#ManagementActivity"/>
        </ObjectSomeValuesFrom>
        <ObjectAllValuesFrom>
            <ObjectProperty IRI="#participatesIn"/>
            <Class IRI="#ManagementActivity"/>
        </ObjectAllValuesFrom>
```

```
    </ObjectIntersectionOf>
  </EquivalentClasses>
  <EquivalentClasses>
    <Class IRI="#ManagementRequiredCapability"/>
    <ObjectIntersectionOf>
      <ObjectSomeValuesFrom>
        <ObjectProperty IRI="#isRequiredBy"/>
        <Class IRI="#ManagementActivity"/>
      </ObjectSomeValuesFrom>
      <ObjectAllValuesFrom>
        <ObjectProperty IRI="#isRequiredBy"/>
        <Class IRI="#ManagementActivity"/>
      </ObjectAllValuesFrom>
    </ObjectIntersectionOf>
  </EquivalentClasses>
  <SubClassOf>
    <Class IRI="#ManagementActivity"/>
    <ObjectIntersectionOf>
      <ObjectAllValuesFrom>
        <ObjectProperty IRI="#isPartOf"/>
        <Class IRI="#ManagementArea"/>
      </ObjectAllValuesFrom>
      <ObjectAllValuesFrom>
        <ObjectProperty IRI="#isUsedBy"/>
        <Class IRI="#ManagementComposedActivity"/>
      </ObjectAllValuesFrom>
      <ObjectExactCardinality cardinality="1">
        <ObjectProperty IRI="#isPartOf"/>
        <Class IRI="#ManagementArea"/>
      </ObjectExactCardinality>
      <DataSomeValuesFrom>
        <DataProperty IRI="#hasDescription"/>
        <Datatype abbreviatedIRI="xsd:string"/>
      </DataSomeValuesFrom>
      <DataAllValuesFrom>
        <DataProperty IRI="#hasDescription"/>
        <Datatype abbreviatedIRI="xsd:string"/>
      </DataAllValuesFrom>
    </ObjectIntersectionOf>
  </SubClassOf>
  <SubClassOf>
    <Class IRI="#ManagementActivity"/>
    <ObjectUnionOf>
      <Class IRI="#ManagementBasicActivity"/>
      <Class IRI="#ManagementComposedActivity"/>
    </ObjectUnionOf>
  </SubClassOf>
  <SubClassOf>
    <Class IRI="#ManagementBasicActivity"/>
    <Class IRI="#ManagementActivity"/>
  </SubClassOf>
```

```
  <SubClassOf>
    <Class IRI="#ManagementCapability"/>
    <ObjectIntersectionOf>
      <ObjectAllValuesFrom>
        <ObjectProperty IRI="#isPartOf"/>
        <Class IRI="#ManagementArea"/>
      </ObjectAllValuesFrom>
      <ObjectExactCardinality cardinality="1">
        <ObjectProperty IRI="#isPartOf"/>
        <Class IRI="#ManagementArea"/>
      </ObjectExactCardinality>
      <DataSomeValuesFrom>
        <DataProperty IRI="#hasDescription"/>
        <Datatype abbreviatedIRI="xsd:string"/>
      </DataSomeValuesFrom>
      <DataAllValuesFrom>
        <DataProperty IRI="#hasDescription"/>
        <Datatype abbreviatedIRI="xsd:string"/>
      </DataAllValuesFrom>
    </ObjectIntersectionOf>
  </SubClassOf>
  <SubClassOf>
    <Class IRI="#ManagementComposedActivity"/>
    <Class IRI="#ManagementActivity"/>
  </SubClassOf>
  <SubClassOf>
    <Class IRI="#ManagementEntity"/>
    <ObjectIntersectionOf>
      <ObjectAllValuesFrom>
        <ObjectProperty IRI="#isPartOf"/>
        <Class IRI="#ManagementArea"/>
      </ObjectAllValuesFrom>
      <ObjectExactCardinality cardinality="1">
        <ObjectProperty IRI="#isPartOf"/>
        <Class IRI="#ManagementArea"/>
      </ObjectExactCardinality>
      <DataSomeValuesFrom>
        <DataProperty IRI="#hasIdentifier"/>
        <Datatype abbreviatedIRI="xsd:string"/>
      </DataSomeValuesFrom>
      <DataAllValuesFrom>
        <DataProperty IRI="#hasIdentifier"/>
        <Datatype abbreviatedIRI="xsd:string"/>
      </DataAllValuesFrom>
    </ObjectIntersectionOf>
  </SubClassOf>
  <SubClassOf>
    <Class IRI="#ManagementEntityStructurePolicy"/>
    <Class IRI="#ManagementPolicy"/>
```

```
  </SubClassOf>
  <SubClassOf>
    <Class IRI="#ManagementParticipant"/>
    <ObjectIntersectionOf>
      <ObjectSomeValuesFrom>
        <ObjectProperty IRI="#isPartOf"/>
        <Class IRI="#ManagementArea"/>
      </ObjectSomeValuesFrom>
      <ObjectAllValuesFrom>
        <ObjectProperty IRI="#isPartOf"/>
        <Class IRI="#ManagementArea"/>
      </ObjectAllValuesFrom>
      <DataSomeValuesFrom>
        <DataProperty IRI="#hasIdentifier"/>
        <Datatype abbreviatedIRI="xsd:string"/>
      </DataSomeValuesFrom>
      <DataAllValuesFrom>
        <DataProperty IRI="#hasIdentifier"/>
        <Datatype abbreviatedIRI="xsd:string"/>
      </DataAllValuesFrom>
    </ObjectIntersectionOf>
  </SubClassOf>
  <SubClassOf>
    <Class IRI="#ManagementPolicy"/>
    <ObjectIntersectionOf>
      <ObjectAllValuesFrom>
        <ObjectProperty IRI="#isPartOf"/>
        <Class IRI="#ManagementArea"/>
      </ObjectAllValuesFrom>
      <DataSomeValuesFrom>
        <DataProperty IRI="#hasDescription"/>
        <Datatype abbreviatedIRI="xsd:string"/>
      </DataSomeValuesFrom>
      <DataAllValuesFrom>
        <DataProperty IRI="#hasDescription"/>
        <Datatype abbreviatedIRI="xsd:string"/>
      </DataAllValuesFrom>
    </ObjectIntersectionOf>
  </SubClassOf>
  <SubClassOf>
    <Class IRI="#ManagementProvidedCapability"/>
    <Class IRI="#ManagementCapability"/>
  </SubClassOf>
  <SubClassOf>
    <Class IRI="#ManagementRequiredCapability"/>
    <Class IRI="#ManagementCapability"/>
  </SubClassOf>
  <DisjointClasses>
    <Class IRI="#ManagementBasicActivity"/>
    <Class IRI="#ManagementComposedActivity"/>
  </DisjointClasses>
  <InverseObjectProperties>
```

```
    <ObjectProperty IRI="#contains"/>
    <ObjectProperty IRI="#isPartOf"/>
  </InverseObjectProperties>
  <InverseObjectProperties>
    <ObjectProperty IRI="#defines"/>
    <ObjectProperty IRI="#isDefinedBy"/>
  </InverseObjectProperties>
  <InverseObjectProperties>
    <ObjectProperty IRI="#hasParticipant"/>
    <ObjectProperty IRI="#participatesIn"/>
  </InverseObjectProperties>
  <InverseObjectProperties>
    <ObjectProperty IRI="#isUsedBy"/>
    <ObjectProperty IRI="#isComposedOf"/>
  </InverseObjectProperties>
  <InverseObjectProperties>
    <ObjectProperty IRI="#isOperatedBy"/>
    <ObjectProperty IRI="#operatesOn"/>
  </InverseObjectProperties>
  <InverseObjectProperties>
    <ObjectProperty IRI="#requires"/>
    <ObjectProperty IRI="#isRequiredBy"/>
  </InverseObjectProperties>
  <FunctionalObjectProperty>
    <ObjectProperty IRI="#defines"/>
  </FunctionalObjectProperty>
  <FunctionalObjectProperty>
    <ObjectProperty IRI="#isDefinedBy"/>
  </FunctionalObjectProperty>
  <TransitiveObjectProperty>
    <ObjectProperty IRI="#isComposedOf"/>
  </TransitiveObjectProperty>
  <TransitiveObjectProperty>
    <ObjectProperty IRI="#isUsedBy"/>
  </TransitiveObjectProperty>
  <ObjectPropertyDomain>
    <ObjectProperty IRI="#contains"/>
    <Class IRI="#ManagementArea"/>
  </ObjectPropertyDomain>
  <ObjectPropertyDomain>
    <ObjectProperty IRI="#defines"/>
    <Class IRI="#ManagementEntityStructurePolicy"/>
  </ObjectPropertyDomain>
  <ObjectPropertyDomain>
    <ObjectProperty IRI="#hasParticipant"/>
    <Class IRI="#ManagementActivity"/>
  </ObjectPropertyDomain>
  <ObjectPropertyDomain>
    <ObjectProperty IRI="#isComposedOf"/>
    <Class IRI="#ManagementComposedActivity"/>
```

```
  </ObjectPropertyDomain>
  <ObjectPropertyDomain>
    <ObjectProperty IRI="#isDefinedBy"/>
    <Class IRI="#ManagementEntity"/>
  </ObjectPropertyDomain>
  <ObjectPropertyDomain>
    <ObjectProperty IRI="#isOperatedBy"/>
    <Class IRI="#ManagementEntity"/>
  </ObjectPropertyDomain>
  <ObjectPropertyDomain>
    <ObjectProperty IRI="#isPartOf"/>
    <ObjectUnionOf>
      <Class IRI="#ManagementActivity"/>
      <Class IRI="#ManagementCapability"/>
      <Class IRI="#ManagementEntity"/>
      <Class IRI="#ManagementParticipant"/>
      <Class IRI="#ManagementPolicy"/>
    </ObjectUnionOf>
  </ObjectPropertyDomain>
  <ObjectPropertyDomain>
    <ObjectProperty IRI="#isRequiredBy"/>
    <Class IRI="#ManagementCapability"/>
  </ObjectPropertyDomain>
  <ObjectPropertyDomain>
    <ObjectProperty IRI="#isUsedBy"/>
    <Class IRI="#ManagementActivity"/>
  </ObjectPropertyDomain>
  <ObjectPropertyDomain>
    <ObjectProperty IRI="#operatesOn"/>
    <Class IRI="#ManagementActivity"/>
  </ObjectPropertyDomain>
  <ObjectPropertyDomain>
    <ObjectProperty IRI="#participatesIn"/>
    <Class IRI="#ManagementParticipant"/>
  </ObjectPropertyDomain>
  <ObjectPropertyDomain>
    <ObjectProperty IRI="#requires"/>
    <Class IRI="#ManagementActivity"/>
  </ObjectPropertyDomain>
  <ObjectPropertyRange>
    <ObjectProperty IRI="#contains"/>
    <ObjectUnionOf>
      <Class IRI="#ManagementActivity"/>
      <Class IRI="#ManagementCapability"/>
      <Class IRI="#ManagementEntity"/>
      <Class IRI="#ManagementParticipant"/>
      <Class IRI="#ManagementPolicy"/>
    </ObjectUnionOf>
  </ObjectPropertyRange>
  <ObjectPropertyRange>
    <ObjectProperty IRI="#defines"/>
    <Class IRI="#ManagementEntity"/>
```

```
    </ObjectPropertyRange>
    <ObjectPropertyRange>
        <ObjectProperty IRI="#hasParticipant"/>
        <Class IRI="#ManagementParticipant"/>
    </ObjectPropertyRange>
    <ObjectPropertyRange>
        <ObjectProperty IRI="#isComposedOf"/>
        <Class IRI="#ManagementActivity"/>
    </ObjectPropertyRange>
    <ObjectPropertyRange>
        <ObjectProperty IRI="#isDefinedBy"/>
        <Class IRI="#ManagementEntityStructurePolicy"/>
    </ObjectPropertyRange>
    <ObjectPropertyRange>
        <ObjectProperty IRI="#isOperatedBy"/>
        <Class IRI="#ManagementActivity"/>
    </ObjectPropertyRange>
    <ObjectPropertyRange>
        <ObjectProperty IRI="#isPartOf"/>
        <Class IRI="#ManagementArea"/>
    </ObjectPropertyRange>
    <ObjectPropertyRange>
        <ObjectProperty IRI="#isRequiredBy"/>
        <Class IRI="#ManagementActivity"/>
    </ObjectPropertyRange>
    <ObjectPropertyRange>
        <ObjectProperty IRI="#isUsedBy"/>
        <Class IRI="#ManagementComposedActivity"/>
    </ObjectPropertyRange>
    <ObjectPropertyRange>
        <ObjectProperty IRI="#operatesOn"/>
        <Class IRI="#ManagementEntity"/>
    </ObjectPropertyRange>
    <ObjectPropertyRange>
        <ObjectProperty IRI="#participatesIn"/>
        <Class IRI="#ManagementActivity"/>
    </ObjectPropertyRange>
    <ObjectPropertyRange>
        <ObjectProperty IRI="#requires"/>
        <Class IRI="#ManagementCapability"/>
    </ObjectPropertyRange>
    <FunctionalDataProperty>
        <DataProperty IRI="#hasIdentifier"/>
    </FunctionalDataProperty>
    <DataPropertyRange>
        <DataProperty IRI="#hasDescription"/>
        <Datatype abbreviatedIRI="xsd:string"/>
    </DataPropertyRange>
    <DataPropertyRange>
        <DataProperty IRI="#hasIdentifier"/>
```

```
      <Datatype abbreviatedIRI="xsd:string"/>
   </DataPropertyRange>
</Ontology>



<!-- Generated by the OWL API (version 3.0.0.1451) http://owlapi.sourceforge.net -->
```

Quelltext 25 OWL-Ontologie für das Domänenmetamodell

2 OWL-Ontologie Incident Management

Die OWL-Ontologie für das *Incident Management* spezifiziert ein Referenzmodell als Ausgangspunkt für den Entwurf von Managementdiensten für die schnelle Störungsbehebung. Nachfolgend ist die konkrete Syntax der Ontologie für das Domänenmetamodell definiert.

```
<?xml version="1.0"?>


<!DOCTYPE Ontology [
   <!ENTITY xsd "http://www.w3.org/2001/XMLSchema#" >
   <!ENTITY xml "http://www.w3.org/XML/1998/namespace" >
   <!ENTITY rdfs "http://www.w3.org/2000/01/rdf-schema#" >
   <!ENTITY rdf "http://www.w3.org/1999/02/22-rdf-syntax-ns#" >
]>


<Ontology xmlns="http://www.w3.org/2002/07/owl#"
    xml:base="http://www.cm-tm.uni-karlsruhe.de/ontologies/2011/1/17/IncidentManagement.owl"
    xmlns:rdfs="http://www.w3.org/2000/01/rdf-schema#"
    xmlns:xsd="http://www.w3.org/2001/XMLSchema#"
    xmlns:rdf="http://www.w3.org/1999/02/22-rdf-syntax-ns#"
    xmlns:xml="http://www.w3.org/XML/1998/namespace"
    ontologyIRI="http://www.cm-tm.uni-karlsruhe.de/ontologies/2011/1/17/IncidentManagement.owl">
   <Prefix name="rdf" IRI="http://www.w3.org/1999/02/22-rdf-syntax-ns#"/>
   <Prefix name="rdfs" IRI="http://www.w3.org/2000/01/rdf-schema#"/>
   <Prefix name="xsd" IRI="http://www.w3.org/2001/XMLSchema#"/>
   <Prefix name="owl" IRI="http://www.w3.org/2002/07/owl#"/>
   <Import>http://www.semanticweb.org/ontologies/2011/1/ITManagement.owl</Import>
   <Declaration>
      <Class IRI="#ClassifyIncident"/>
   </Declaration>
   <Declaration>
      <Class IRI="#CreateIncidentRecord"/>
   </Declaration>
   <Declaration>
      <Class IRI="#DetermineBusinessImpact"/>
   </Declaration>
   <Declaration>
      <Class IRI="#EscalateIncident"/>
   </Declaration>
```

```
    <Declaration>
        <Class IRI="#IncidentRecord"/>
    </Declaration>
    <Declaration>
        <Class IRI="#IncidentRecordStructurePolicy"/>
    </Declaration>
    <Declaration>
        <Class IRI="#InformCustomer"/>
    </Declaration>
    <Declaration>
        <Class IRI="#PrioritizeIncident"/>
    </Declaration>
    <Declaration>
        <Class IRI="#ReadIncidentRecord"/>
    </Declaration>
    <Declaration>
        <Class IRI="#RecordIncident"/>
    </Declaration>
    <Declaration>
        <Class IRI="#ResolveIncident"/>
    </Declaration>
    <Declaration>
        <Class IRI="#UpdateIncidentRecord"/>
    </Declaration>
    <SubClassOf>
        <Class IRI="#ClassifyIncident"/>
        <Class
IRI="http://www.semanticweb.org/ontologies/2011/1/ITManagement.owl#ManagementComposedActivity"/>
    </SubClassOf>
    <SubClassOf>
        <Class IRI="#CreateIncidentRecord"/>
        <Class
IRI="http://www.semanticweb.org/ontologies/2011/1/ITManagement.owl#ManagementBasicActivity"/>
    </SubClassOf>
    <SubClassOf>
        <Class IRI="#DetermineBusinessImpact"/>
        <Class
IRI="http://www.semanticweb.org/ontologies/2011/1/ITManagement.owl#ManagementComposedActivity"/>
    </SubClassOf>
    <SubClassOf>
        <Class IRI="#EscalateIncident"/>
        <Class
IRI="http://www.semanticweb.org/ontologies/2011/1/ITManagement.owl#ManagementComposedActivity"/>
    </SubClassOf>
    <SubClassOf>
        <Class IRI="#IncidentRecord"/>
        <Class IRI="http://www.semanticweb.org/ontologies/2011/1/ITManagement.owl#ManagementEntity"/>
    </SubClassOf>
    <SubClassOf>
```

```
        <Class IRI="#IncidentRecordStructurePolicy"/>
        <Class
IRI="http://www.semanticweb.org/ontologies/2011/1/ITManagement.owl#ManagementEntityStructurePolicy"/>
    </SubClassOf>
    <SubClassOf>
        <Class IRI="#InformCustomer"/>
        <Class
IRI="http://www.semanticweb.org/ontologies/2011/1/ITManagement.owl#ManagementComposedActivity"/>
    </SubClassOf>
    <SubClassOf>
        <Class IRI="#PrioritizeIncident"/>
        <Class
IRI="http://www.semanticweb.org/ontologies/2011/1/ITManagement.owl#ManagementComposedActivity"/>
    </SubClassOf>
    <SubClassOf>
        <Class IRI="#ReadIncidentRecord"/>
        <Class
IRI="http://www.semanticweb.org/ontologies/2011/1/ITManagement.owl#ManagementBasicActivity"/>
    </SubClassOf>
    <SubClassOf>
        <Class IRI="#RecordIncident"/>
        <Class
IRI="http://www.semanticweb.org/ontologies/2011/1/ITManagement.owl#ManagementComposedActivity"/>
    </SubClassOf>
    <SubClassOf>
        <Class IRI="#ResolveIncident"/>
        <Class
IRI="http://www.semanticweb.org/ontologies/2011/1/ITManagement.owl#ManagementComposedActivity"/>
    </SubClassOf>
    <SubClassOf>
        <Class IRI="#UpdateIncidentRecord"/>
        <Class
IRI="http://www.semanticweb.org/ontologies/2011/1/ITManagement.owl#ManagementBasicActivity"/>
    </SubClassOf>
    <DisjointClasses>
        <Class IRI="#ClassifyIncident"/>
        <Class IRI="#DetermineBusinessImpact"/>
        <Class IRI="#EscalateIncident"/>
        <Class IRI="#InformCustomer"/>
        <Class IRI="#PrioritizeIncident"/>
        <Class IRI="#ResolveIncident"/>
    </DisjointClasses>
    <DisjointClasses>
        <Class IRI="#CreateIncidentRecord"/>
        <Class IRI="#ReadIncidentRecord"/>
        <Class IRI="#UpdateIncidentRecord"/>
    </DisjointClasses>
</Ontology>
```

```
<!-- Generated by the OWL API (version 3.0.0.1451) http://owlapi.sourceforge.net -->
```

Quelltext 26 OWL-Ontologie für das Incident Management

3 OWL-Ontologie Problem Management

Die OWL-Ontologie für das Problem Management spezifiziert ein Referenzmodell als Ausgangspunkt für den Entwurf von Managementdiensten für reaktives und proaktives Problemmanagement. Nachfolgend ist die konkrete Syntax der Ontologie für das Problem Management definiert.

```
<?xml version="1.0"?>


<!DOCTYPE Ontology [
    <!ENTITY xsd "http://www.w3.org/2001/XMLSchema#" >
    <!ENTITY xml "http://www.w3.org/XML/1998/namespace" >
    <!ENTITY rdfs "http://www.w3.org/2000/01/rdf-schema#" >
    <!ENTITY rdf "http://www.w3.org/1999/02/22-rdf-syntax-ns#" >
]>


<Ontology xmlns="http://www.w3.org/2002/07/owl#"
     xml:base="http://www.cm-tm.uni-karlsruhe.de/ontologies/2011/1/17/ProblemManagement.owl"
     xmlns:rdfs="http://www.w3.org/2000/01/rdf-schema#"
     xmlns:xsd="http://www.w3.org/2001/XMLSchema#"
     xmlns:rdf="http://www.w3.org/1999/02/22-rdf-syntax-ns#"
     xmlns:xml="http://www.w3.org/XML/1998/namespace"
     ontologyIRI="http://www.cm-tm.uni-karlsruhe.de/ontologies/2011/1/17/ProblemManagement.owl">
    <Prefix name="rdf" IRI="http://www.w3.org/1999/02/22-rdf-syntax-ns#"/>
    <Prefix name="rdfs" IRI="http://www.w3.org/2000/01/rdf-schema#"/>
    <Prefix name="xsd" IRI="http://www.w3.org/2001/XMLSchema#"/>
    <Prefix name="owl" IRI="http://www.w3.org/2002/07/owl#"/>
    <Import>http://www.semanticweb.org/ontologies/2011/1/ITManagement.owl</Import>
    <Declaration>
        <Class IRI="#ClassifyProblem"/>
    </Declaration>
    <Declaration>
        <Class IRI="#CreateKnownErrorRecord"/>
    </Declaration>
    <Declaration>
        <Class IRI="#CreateKnownProblemRecord"/>
    </Declaration>
    <Declaration>
        <Class IRI="#CreateProblemRecord"/>
    </Declaration>
    <Declaration>
        <Class IRI="#CreateProblemResolutionRecord"/>
    </Declaration>
    <Declaration>
        <Class IRI="#EscalateProblem"/>
```

```
  </Declaration>
  <Declaration>
    <Class IRI="#IdentifyImpact"/>
  </Declaration>
  <Declaration>
    <Class IRI="#KnownErrorRecord"/>
  </Declaration>
  <Declaration>
    <Class IRI="#KnownErrorRecordStructurePolicy"/>
  </Declaration>
  <Declaration>
    <Class IRI="#KnownProblemRecord"/>
  </Declaration>
  <Declaration>
    <Class IRI="#KnownProblemRecordStructurePolicy"/>
  </Declaration>
  <Declaration>
    <Class IRI="#ProblemResolutionRecord"/>
  </Declaration>
  <Declaration>
    <Class IRI="#ProblemResolutionRecordStructurePolicy"/>
  </Declaration>
  <Declaration>
    <Class IRI="#ReadKnownErrorRecord"/>
  </Declaration>
  <Declaration>
    <Class IRI="#ReadKnownProblemRecord"/>
  </Declaration>
  <Declaration>
    <Class IRI="#ReadProblemRecord"/>
  </Declaration>
  <Declaration>
    <Class IRI="#ReadProblemResolutionRecord"/>
  </Declaration>
  <Declaration>
    <Class IRI="#RecordProblem"/>
  </Declaration>
  <Declaration>
    <Class IRI="#ResolveProblem"/>
  </Declaration>
  <Declaration>
    <Class IRI="#UpdateKnownErrorRecord"/>
  </Declaration>
  <Declaration>
    <Class IRI="#UpdateKnownProblemRecord"/>
  </Declaration>
  <Declaration>
    <Class IRI="#UpdateProblemRecord"/>
  </Declaration>
  <Declaration>
    <Class IRI="#UpdateProblemResolutionRecord"/>
  </Declaration>
```

```
    <SubClassOf>
        <Class IRI="#ClassifyProblem"/>
        <Class
IRI="http://www.semanticweb.org/ontologies/2011/1/ITManagement.owl#ManagementComposedActivity"/>
    </SubClassOf>
    <SubClassOf>
        <Class IRI="#CreateKnownErrorRecord"/>
        <Class
IRI="http://www.semanticweb.org/ontologies/2011/1/ITManagement.owl#ManagementBasicActivity"/>
    </SubClassOf>
    <SubClassOf>
        <Class IRI="#CreateKnownProblemRecord"/>
        <Class
IRI="http://www.semanticweb.org/ontologies/2011/1/ITManagement.owl#ManagementBasicActivity"/>
    </SubClassOf>
    <SubClassOf>
        <Class IRI="#CreateProblemRecord"/>
        <Class
IRI="http://www.semanticweb.org/ontologies/2011/1/ITManagement.owl#ManagementBasicActivity"/>
    </SubClassOf>
    <SubClassOf>
        <Class IRI="#CreateProblemResolutionRecord"/>
        <Class
IRI="http://www.semanticweb.org/ontologies/2011/1/ITManagement.owl#ManagementBasicActivity"/>
    </SubClassOf>
    <SubClassOf>
        <Class IRI="#EscalateProblem"/>
        <Class
IRI="http://www.semanticweb.org/ontologies/2011/1/ITManagement.owl#ManagementComposedActivity"/>
    </SubClassOf>
    <SubClassOf>
        <Class IRI="#IdentifyImpact"/>
        <Class
IRI="http://www.semanticweb.org/ontologies/2011/1/ITManagement.owl#ManagementComposedActivity"/>
    </SubClassOf>
    <SubClassOf>
        <Class IRI="#KnownErrorRecord"/>
        <Class IRI="http://www.semanticweb.org/ontologies/2011/1/ITManagement.owl#ManagementEntity"/>
    </SubClassOf>
    <SubClassOf>
        <Class IRI="#KnownErrorRecordStructurePolicy"/>
        <Class
IRI="http://www.semanticweb.org/ontologies/2011/1/ITManagement.owl#ManagementEntityStructurePolicy"/>
    </SubClassOf>
    <SubClassOf>
        <Class IRI="#KnownProblemRecord"/>
        <Class IRI="http://www.semanticweb.org/ontologies/2011/1/ITManagement.owl#ManagementEntity"/>
    </SubClassOf>
    <SubClassOf>
```

```
      <Class IRI="#KnownProblemRecordStructurePolicy"/>
      <Class
IRI="http://www.semanticweb.org/ontologies/2011/1/ITManagement.owl#ManagementEntityStructurePolicy"/>
   </SubClassOf>
   <SubClassOf>
      <Class IRI="#ProblemResolutionRecord"/>
      <Class IRI="http://www.semanticweb.org/ontologies/2011/1/ITManagement.owl#ManagementEntity"/>
   </SubClassOf>
   <SubClassOf>
      <Class IRI="#ProblemResolutionRecordStructurePolicy"/>
      <Class
IRI="http://www.semanticweb.org/ontologies/2011/1/ITManagement.owl#ManagementEntityStructurePolicy"/>
   </SubClassOf>
   <SubClassOf>
      <Class IRI="#ReadKnownErrorRecord"/>
      <Class
IRI="http://www.semanticweb.org/ontologies/2011/1/ITManagement.owl#ManagementBasicActivity"/>
   </SubClassOf>
   <SubClassOf>
      <Class IRI="#ReadKnownProblemRecord"/>
      <Class
IRI="http://www.semanticweb.org/ontologies/2011/1/ITManagement.owl#ManagementBasicActivity"/>
   </SubClassOf>
   <SubClassOf>
      <Class IRI="#ReadProblemRecord"/>
      <Class
IRI="http://www.semanticweb.org/ontologies/2011/1/ITManagement.owl#ManagementBasicActivity"/>
   </SubClassOf>
   <SubClassOf>
      <Class IRI="#ReadProblemResolutionRecord"/>
      <Class
IRI="http://www.semanticweb.org/ontologies/2011/1/ITManagement.owl#ManagementBasicActivity"/>
   </SubClassOf>
   <SubClassOf>
      <Class IRI="#RecordProblem"/>
      <Class
IRI="http://www.semanticweb.org/ontologies/2011/1/ITManagement.owl#ManagementComposedActivity"/>
   </SubClassOf>
   <SubClassOf>
      <Class IRI="#ResolveProblem"/>
      <Class
IRI="http://www.semanticweb.org/ontologies/2011/1/ITManagement.owl#ManagementComposedActivity"/>
   </SubClassOf>
   <SubClassOf>
      <Class IRI="#UpdateKnownErrorRecord"/>
      <Class
IRI="http://www.semanticweb.org/ontologies/2011/1/ITManagement.owl#ManagementBasicActivity"/>
   </SubClassOf>
   <SubClassOf>
      <Class IRI="#UpdateKnownProblemRecord"/>
```

```
        <Class
IRI="http://www.semanticweb.org/ontologies/2011/1/ITManagement.owl#ManagementBasicActivity"/>
    </SubClassOf>
    <SubClassOf>
        <Class IRI="#UpdateProblemRecord"/>
        <Class
IRI="http://www.semanticweb.org/ontologies/2011/1/ITManagement.owl#ManagementBasicActivity"/>
    </SubClassOf>
    <SubClassOf>
        <Class IRI="#UpdateProblemResolutionRecord"/>
        <Class
IRI="http://www.semanticweb.org/ontologies/2011/1/ITManagement.owl#ManagementBasicActivity"/>
    </SubClassOf>
    <DisjointClasses>
        <Class IRI="#ClassifyProblem"/>
        <Class IRI="#EscalateProblem"/>
        <Class IRI="#IdentifyImpact"/>
        <Class IRI="#RecordProblem"/>
        <Class IRI="#ResolveProblem"/>
    </DisjointClasses>
    <DisjointClasses>
        <Class IRI="#CreateKnownErrorRecord"/>
        <Class IRI="#CreateKnownProblemRecord"/>
        <Class IRI="#CreateProblemRecord"/>
        <Class IRI="#CreateProblemResolutionRecord"/>
        <Class IRI="#ReadKnownErrorRecord"/>
        <Class IRI="#ReadKnownProblemRecord"/>
        <Class IRI="#ReadProblemRecord"/>
        <Class IRI="#ReadProblemResolutionRecord"/>
        <Class IRI="#UpdateKnownErrorRecord"/>
        <Class IRI="#UpdateKnownProblemRecord"/>
        <Class IRI="#UpdateProblemRecord"/>
        <Class IRI="#UpdateProblemResolutionRecord"/>
    </DisjointClasses>
    <DisjointClasses>
        <Class IRI="#KnownErrorRecord"/>
        <Class IRI="#KnownProblemRecord"/>
        <Class IRI="#ProblemResolutionRecord"/>
    </DisjointClasses>
    <DisjointClasses>
        <Class IRI="#KnownErrorRecordStructurePolicy"/>
        <Class IRI="#KnownProblemRecordStructurePolicy"/>
        <Class IRI="#ProblemResolutionRecordStructurePolicy"/>
    </DisjointClasses>
</Ontology>



<!-- Generated by the OWL API (version 3.0.0.1451) http://owlapi.sourceforge.net -->
```

Quelltext 27 OWL-Ontologie für das Problem Management

4 OWL-Ontologie Change Management

Die OWL-Ontologie für das Change Management spezifiziert ein Referenzmodell als Ausgangspunkt für den Entwurf von Managementdiensten zur Unterstützung von Änderungsplanungen. Nachfolgend ist die konkrete Syntax der Ontologie für das Change Management definiert.

```
<?xml version="1.0"?>


<!DOCTYPE Ontology [
   <!ENTITY xsd "http://www.w3.org/2001/XMLSchema#" >
   <!ENTITY xml "http://www.w3.org/XML/1998/namespace" >
   <!ENTITY rdfs "http://www.w3.org/2000/01/rdf-schema#" >
   <!ENTITY rdf "http://www.w3.org/1999/02/22-rdf-syntax-ns#" >
]>


<Ontology xmlns="http://www.w3.org/2002/07/owl#"
    xml:base="http://www.cm-tm.uni-karlsruhe.de/ontologies/2011/1/17/ChangeManagement.owl"
    xmlns:rdfs="http://www.w3.org/2000/01/rdf-schema#"
    xmlns:xsd="http://www.w3.org/2001/XMLSchema#"
    xmlns:rdf="http://www.w3.org/1999/02/22-rdf-syntax-ns#"
    xmlns:xml="http://www.w3.org/XML/1998/namespace"
    ontologyIRI="http://www.cm-tm.uni-karlsruhe.de/ontologies/2011/1/17/ChangeManagement.owl">
   <Prefix name="rdf" IRI="http://www.w3.org/1999/02/22-rdf-syntax-ns#"/>
   <Prefix name="rdfs" IRI="http://www.w3.org/2000/01/rdf-schema#"/>
   <Prefix name="xsd" IRI="http://www.w3.org/2001/XMLSchema#"/>
   <Prefix name="owl" IRI="http://www.w3.org/2002/07/owl#"/>
   <Import>http://www.semanticweb.org/ontologies/2011/1/ITManagement.owl</Import>
   <Declaration>
      <Class IRI="#ApproveRequestForChange"/>
   </Declaration>
   <Declaration>
      <Class IRI="#AssessRequestForChange"/>
   </Declaration>
   <Declaration>
      <Class IRI="#ChangePlanStructurePolicy"/>
   </Declaration>
   <Declaration>
      <Class IRI="#ChangePlantRecord"/>
   </Declaration>
   <Declaration>
      <Class IRI="#ClassifyRequestForChange"/>
   </Declaration>
   <Declaration>
      <Class IRI="#CreateChangePlanRecord"/>
   </Declaration>
   <Declaration>
      <Class IRI="#CreateRequestForChangeRecord"/>
   </Declaration>
```

```
    <Declaration>
        <Class IRI="#ReadChangePlanRecord"/>
    </Declaration>
    <Declaration>
        <Class IRI="#ReadRequestForChangeRecord"/>
    </Declaration>
    <Declaration>
        <Class IRI="#RecordRequestForChange"/>
    </Declaration>
    <Declaration>
        <Class IRI="#RequestForChangeRecord"/>
    </Declaration>
    <Declaration>
        <Class IRI="#RequestForChangeStructurePolicy"/>
    </Declaration>
    <Declaration>
        <Class IRI="#ReverseRequestForChange"/>
    </Declaration>
    <Declaration>
        <Class IRI="#ReviewRequestForChange"/>
    </Declaration>
    <Declaration>
        <Class IRI="#UpdateChangePlanRecord"/>
    </Declaration>
    <Declaration>
        <Class IRI="#UpdateRequestForChangeRecord"/>
    </Declaration>
    <SubClassOf>
        <Class IRI="#ApproveRequestForChange"/>
        <Class
IRI="http://www.semanticweb.org/ontologies/2011/1/ITManagement.owl#ManagementComposedActivity"/>
    </SubClassOf>
    <SubClassOf>
        <Class IRI="#AssessRequestForChange"/>
        <Class
IRI="http://www.semanticweb.org/ontologies/2011/1/ITManagement.owl#ManagementComposedActivity"/>
    </SubClassOf>
    <SubClassOf>
        <Class IRI="#ChangePlanStructurePolicy"/>
        <Class
IRI="http://www.semanticweb.org/ontologies/2011/1/ITManagement.owl#ManagementEntityStructurePolicy"/>
    </SubClassOf>
    <SubClassOf>
        <Class IRI="#ChangePlantRecord"/>
        <Class IRI="http://www.semanticweb.org/ontologies/2011/1/ITManagement.owl#ManagementEntity"/>
    </SubClassOf>
    <SubClassOf>
        <Class IRI="#ClassifyRequestForChange"/>
        <Class
IRI="http://www.semanticweb.org/ontologies/2011/1/ITManagement.owl#ManagementComposedActivity"/>
```

```
    </SubClassOf>
    <SubClassOf>
        <Class IRI="#CreateChangePlanRecord"/>
        <Class
IRI="http://www.semanticweb.org/ontologies/2011/1/ITManagement.owl#ManagementBasicActivity"/>
    </SubClassOf>
    <SubClassOf>
        <Class IRI="#CreateRequestForChangeRecord"/>
        <Class
IRI="http://www.semanticweb.org/ontologies/2011/1/ITManagement.owl#ManagementBasicActivity"/>
    </SubClassOf>
    <SubClassOf>
        <Class IRI="#ReadChangePlanRecord"/>
        <Class
IRI="http://www.semanticweb.org/ontologies/2011/1/ITManagement.owl#ManagementBasicActivity"/>
    </SubClassOf>
    <SubClassOf>
        <Class IRI="#ReadRequestForChangeRecord"/>
        <Class
IRI="http://www.semanticweb.org/ontologies/2011/1/ITManagement.owl#ManagementBasicActivity"/>
    </SubClassOf>
    <SubClassOf>
        <Class IRI="#RecordRequestForChange"/>
        <Class
IRI="http://www.semanticweb.org/ontologies/2011/1/ITManagement.owl#ManagementComposedActivity"/>
    </SubClassOf>
    <SubClassOf>
        <Class IRI="#RequestForChangeRecord"/>
        <Class IRI="http://www.semanticweb.org/ontologies/2011/1/ITManagement.owl#ManagementEntity"/>
    </SubClassOf>
    <SubClassOf>
        <Class IRI="#RequestForChangeStructurePolicy"/>
        <Class
IRI="http://www.semanticweb.org/ontologies/2011/1/ITManagement.owl#ManagementEntityStructurePolicy"/>
    </SubClassOf>
    <SubClassOf>
        <Class IRI="#ReverseRequestForChange"/>
        <Class
IRI="http://www.semanticweb.org/ontologies/2011/1/ITManagement.owl#ManagementComposedActivity"/>
    </SubClassOf>
    <SubClassOf>
        <Class IRI="#ReviewRequestForChange"/>
        <Class
IRI="http://www.semanticweb.org/ontologies/2011/1/ITManagement.owl#ManagementComposedActivity"/>
    </SubClassOf>
    <SubClassOf>
        <Class IRI="#UpdateChangePlanRecord"/>
        <Class
IRI="http://www.semanticweb.org/ontologies/2011/1/ITManagement.owl#ManagementBasicActivity"/>
    </SubClassOf>
    <SubClassOf>
```

```
      <Class IRI="#UpdateRequestForChangeRecord"/>
      <Class
IRI="http://www.semanticweb.org/ontologies/2011/1/ITManagement.owl#ManagementBasicActivity"/>
   </SubClassOf>
   <DisjointClasses>
      <Class IRI="#ApproveRequestForChange"/>
      <Class IRI="#AssessRequestForChange"/>
      <Class IRI="#ClassifyRequestForChange"/>
      <Class IRI="#RecordRequestForChange"/>
      <Class IRI="#ReverseRequestForChange"/>
      <Class IRI="#ReviewRequestForChange"/>
   </DisjointClasses>
   <DisjointClasses>
      <Class IRI="#ChangePlanStructurePolicy"/>
      <Class IRI="#RequestForChangeStructurePolicy"/>
   </DisjointClasses>
   <DisjointClasses>
      <Class IRI="#ChangePlantRecord"/>
      <Class IRI="#RequestForChangeRecord"/>
   </DisjointClasses>
   <DisjointClasses>
      <Class IRI="#CreateChangePlanRecord"/>
      <Class IRI="#CreateRequestForChangeRecord"/>
      <Class IRI="#ReadChangePlanRecord"/>
      <Class IRI="#ReadRequestForChangeRecord"/>
      <Class IRI="#UpdateChangePlanRecord"/>
      <Class IRI="#UpdateRequestForChangeRecord"/>
   </DisjointClasses>
</Ontology>



<!-- Generated by the OWL API (version 3.0.0.1451) http://owlapi.sourceforge.net -->
```

Quelltext 28 OWL-Ontologie für das Change Management

5 OWL-Ontologie Release Management

Die OWL-Ontologie für das Release Management spezifiziert ein Referenzmodell als Ausgangspunkt für den Entwurf von Managementdiensten für die Durchführung von Änderungen.

Nachfolgend ist die konkrete Syntax der Ontologie für das Release Management definiert.

```
<?xml version="1.0"?>


<!DOCTYPE Ontology [
   <!ENTITY xsd "http://www.w3.org/2001/XMLSchema#" >
   <!ENTITY xml "http://www.w3.org/XML/1998/namespace" >
   <!ENTITY rdfs "http://www.w3.org/2000/01/rdf-schema#" >
```

```
   <!ENTITY rdf "http://www.w3.org/1999/02/22-rdf-syntax-ns#" >
]>


<Ontology xmlns="http://www.w3.org/2002/07/owl#"
    xml:base="http://www.cm-tm.uni-karlsruhe.de/ontologies/2011/1/17/ReleaseManagement.owl"
    xmlns:rdfs="http://www.w3.org/2000/01/rdf-schema#"
    xmlns:xsd="http://www.w3.org/2001/XMLSchema#"
    xmlns:rdf="http://www.w3.org/1999/02/22-rdf-syntax-ns#"
    xmlns:xml="http://www.w3.org/XML/1998/namespace"
    ontologyIRI="http://www.cm-tm.uni-karlsruhe.de/ontologies/2011/1/17/ReleaseManagement.owl">
   <Prefix name="rdf" IRI="http://www.w3.org/1999/02/22-rdf-syntax-ns#"/>
   <Prefix name="rdfs" IRI="http://www.w3.org/2000/01/rdf-schema#"/>
   <Prefix name="xsd" IRI="http://www.w3.org/2001/XMLSchema#"/>
   <Prefix name="owl" IRI="http://www.w3.org/2002/07/owl#"/>
   <Import>http://www.semanticweb.org/ontologies/2011/1/ITManagement.owl</Import>
   <Declaration>
      <Class IRI="#AssessReleaseImpact"/>
   </Declaration>
   <Declaration>
      <Class IRI="#BuildRelease"/>
   </Declaration>
   <Declaration>
      <Class IRI="#CreateReleasePlanRecord"/>
   </Declaration>
   <Declaration>
      <Class IRI="#PlanRelease"/>
   </Declaration>
   <Declaration>
      <Class IRI="#ReadReleasePlanRecord"/>
   </Declaration>
   <Declaration>
      <Class IRI="#ReleasePlanRecord"/>
   </Declaration>
   <Declaration>
      <Class IRI="#ReleasePlanStructurePolicy"/>
   </Declaration>
   <Declaration>
      <Class IRI="#ReportReleaseStatistics"/>
   </Declaration>
   <Declaration>
      <Class IRI="#ReverseRelease"/>
   </Declaration>
   <Declaration>
      <Class IRI="#TestRelease"/>
   </Declaration>
   <Declaration>
      <Class IRI="#UpdateConfigurationInformation"/>
   </Declaration>
   <Declaration>
      <Class IRI="#UpdateReleasePlanRecord"/>
```

```
    </Declaration>
    <SubClassOf>
        <Class IRI="#AssessReleaseImpact"/>
        <Class
IRI="http://www.semanticweb.org/ontologies/2011/1/ITManagement.owl#ManagementComposedActivity"/>
    </SubClassOf>
    <SubClassOf>
        <Class IRI="#BuildRelease"/>
        <Class
IRI="http://www.semanticweb.org/ontologies/2011/1/ITManagement.owl#ManagementComposedActivity"/>
    </SubClassOf>
    <SubClassOf>
        <Class IRI="#CreateReleasePlanRecord"/>
        <Class
IRI="http://www.semanticweb.org/ontologies/2011/1/ITManagement.owl#ManagementBasicActivity"/>
    </SubClassOf>
    <SubClassOf>
        <Class IRI="#PlanRelease"/>
        <Class
IRI="http://www.semanticweb.org/ontologies/2011/1/ITManagement.owl#ManagementComposedActivity"/>
    </SubClassOf>
    <SubClassOf>
        <Class IRI="#ReadReleasePlanRecord"/>
        <Class
IRI="http://www.semanticweb.org/ontologies/2011/1/ITManagement.owl#ManagementBasicActivity"/>
    </SubClassOf>
    <SubClassOf>
        <Class IRI="#ReleasePlanRecord"/>
        <Class IRI="http://www.semanticweb.org/ontologies/2011/1/ITManagement.owl#ManagementEntity"/>
    </SubClassOf>
    <SubClassOf>
        <Class IRI="#ReleasePlanStructurePolicy"/>
        <Class
IRI="http://www.semanticweb.org/ontologies/2011/1/ITManagement.owl#ManagementEntityStructurePolicy"/>
    </SubClassOf>
    <SubClassOf>
        <Class IRI="#ReportReleaseStatistics"/>
        <Class
IRI="http://www.semanticweb.org/ontologies/2011/1/ITManagement.owl#ManagementComposedActivity"/>
    </SubClassOf>
    <SubClassOf>
        <Class IRI="#ReverseRelease"/>
        <Class
IRI="http://www.semanticweb.org/ontologies/2011/1/ITManagement.owl#ManagementComposedActivity"/>
    </SubClassOf>
    <SubClassOf>
        <Class IRI="#TestRelease"/>
        <Class
IRI="http://www.semanticweb.org/ontologies/2011/1/ITManagement.owl#ManagementComposedActivity"/>
```

```
    </SubClassOf>
    <SubClassOf>
        <Class IRI="#UpdateConfigurationInformation"/>
        <Class
IRI="http://www.semanticweb.org/ontologies/2011/1/ITManagement.owl#ManagementComposedActivity"/>
    </SubClassOf>
    <SubClassOf>
        <Class IRI="#UpdateReleasePlanRecord"/>
        <Class
IRI="http://www.semanticweb.org/ontologies/2011/1/ITManagement.owl#ManagementBasicActivity"/>
    </SubClassOf>
    <DisjointClasses>
        <Class IRI="#AssessReleaseImpact"/>
        <Class IRI="#BuildRelease"/>
        <Class IRI="#PlanRelease"/>
        <Class IRI="#ReportReleaseStatistics"/>
        <Class IRI="#ReverseRelease"/>
        <Class IRI="#TestRelease"/>
        <Class IRI="#UpdateConfigurationInformation"/>
    </DisjointClasses>
    <DisjointClasses>
        <Class IRI="#CreateReleasePlanRecord"/>
        <Class IRI="#ReadReleasePlanRecord"/>
        <Class IRI="#UpdateReleasePlanRecord"/>
    </DisjointClasses>
</Ontology>



<!-- Generated by the OWL API (version 3.0.0.1451) http://owlapi.sourceforge.net -->
```

Quelltext 29 OWL-Ontologie für das Release Management

6 OWL-Ontologie Configuration Management

Die OWL-Ontologie für das Configuration Management spezifiziert ein Referenzmodell als Ausgangspunkt für den Entwurf von Managementdiensten für die querschnittlich genutzten Funktionen im Konfigurationsmanagement.

Nachfolgend ist die konkrete Syntax der Ontologie für das Configuration Management definiert.

```
<?xml version="1.0"?>


<!DOCTYPE Ontology [
    <!ENTITY xsd "http://www.w3.org/2001/XMLSchema#" >
    <!ENTITY xml "http://www.w3.org/XML/1998/namespace" >
    <!ENTITY rdfs "http://www.w3.org/2000/01/rdf-schema#" >
    <!ENTITY rdf "http://www.w3.org/1999/02/22-rdf-syntax-ns#" >
]>
```

```
<Ontology xmlns="http://www.w3.org/2002/07/owl#"
    xml:base="http://www.cm-tm.uni-karlsruhe.de/ontologies/2011/1/17/ConfigurationManagement.owl"
    xmlns:rdfs="http://www.w3.org/2000/01/rdf-schema#"
    xmlns:xsd="http://www.w3.org/2001/XMLSchema#"
    xmlns:rdf="http://www.w3.org/1999/02/22-rdf-syntax-ns#"
    xmlns:xml="http://www.w3.org/XML/1998/namespace"
    ontologyIRI="http://www.cm-tm.uni-karlsruhe.de/ontologies/2011/1/17/ConfigurationManagement.owl">
   <Prefix name="rdf" IRI="http://www.w3.org/1999/02/22-rdf-syntax-ns#"/>
   <Prefix name="rdfs" IRI="http://www.w3.org/2000/01/rdf-schema#"/>
   <Prefix name="xsd" IRI="http://www.w3.org/2001/XMLSchema#"/>
   <Prefix name="owl" IRI="http://www.w3.org/2002/07/owl#"/>
   <Import>http://www.semanticweb.org/ontologies/2011/1/ITManagement.owl</Import>
   <Declaration>
      <Class IRI="#ConfigurationItem"/>
   </Declaration>
   <Declaration>
      <Class IRI="#ConfigurationItemStructurePolicy"/>
   </Declaration>
   <Declaration>
      <Class IRI="#CreateConfigurationItem"/>
   </Declaration>
   <Declaration>
      <Class IRI="#DetermineRelationship"/>
   </Declaration>
   <Declaration>
      <Class IRI="#ReadUpdateConfigurationItem"/>
   </Declaration>
   <Declaration>
      <Class IRI="#TakeBaseline"/>
   </Declaration>
   <SubClassOf>
      <Class IRI="#ConfigurationItem"/>
      <Class IRI="http://www.semanticweb.org/ontologies/2011/1/ITManagement.owl#ManagementEntity"/>
   </SubClassOf>
   <SubClassOf>
      <Class IRI="#ConfigurationItemStructurePolicy"/>
      <Class
IRI="http://www.semanticweb.org/ontologies/2011/1/ITManagement.owl#ManagementEntityStructurePolicy"/>
   </SubClassOf>
   <SubClassOf>
      <Class IRI="#CreateConfigurationItem"/>
      <Class
IRI="http://www.semanticweb.org/ontologies/2011/1/ITManagement.owl#ManagementBasicActivity"/>
   </SubClassOf>
   <SubClassOf>
      <Class IRI="#DetermineRelationship"/>
      <Class
IRI="http://www.semanticweb.org/ontologies/2011/1/ITManagement.owl#ManagementComposedActivity"/>
   </SubClassOf>
```

```
    <SubClassOf>
        <Class IRI="#ReadUpdateConfigurationItem"/>
        <Class
IRI="http://www.semanticweb.org/ontologies/2011/1/ITManagement.owl#ManagementBasicActivity"/>
    </SubClassOf>
    <SubClassOf>
        <Class IRI="#TakeBaseline"/>
        <Class
IRI="http://www.semanticweb.org/ontologies/2011/1/ITManagement.owl#ManagementComposedActivity"/>
    </SubClassOf>
    <DisjointClasses>
        <Class IRI="#CreateConfigurationItem"/>
        <Class IRI="#ReadUpdateConfigurationItem"/>
    </DisjointClasses>
    <DisjointClasses>
        <Class IRI="#DetermineRelationship"/>
        <Class IRI="#TakeBaseline"/>
    </DisjointClasses>
</Ontology>



<!-- Generated by the OWL API (version 3.0.0.1451) http://owlapi.sourceforge.net -->
```

Quelltext 30 OWL-Ontologie für das Configuration Management