Ein prozessorientierter, policy–basierter Ansatz für ein integriertes, dienstorientiertes Abrechnungsmanagement

Dissertation

an der

Fakultät für Mathematik, Informatik und Statistik
der
Ludwig-Maximilians-Universität München

vorgelegt von

Igor Radisic

Tag der Einreichung: 10. Januar 2003
Tag der mündlichen Prüfung: 20. Februar 2003

1. Berichterstatter: **Professor Dr. Heinz-Gerd Hegering**, Universität München
2. Berichterstatter: **Professor Dr. Johann Schlichter**, Technische Universität München

Als Buch erhältlich im Verlag Dr. Hut, München, www.dr.hut-verlag.de (ISBN 3-934767-97-4)

Danksagung

Die vorliegende Arbeit entstand während meiner Tätigkeit als wissenschaftlicher Mitarbeiter am Lehrstuhl für Kommunikationssysteme und Systemprogrammierung der Universität München im Rahmen einer mehrjährigen Forschungskooperation mit der Deutschen Telekom Systemlösungen (DeTeSystem).

Mein ganz besonderer Dank gilt meinem Doktorvater, Prof. Dr. Heinz-Gerd Hegering, für die hervorragende und sehr vorbildliche Betreuung während der Erstellung der Arbeit. Sowohl seine wertvolle, stets konstruktive Kritik als auch seine Anleitung während der Erarbeitung von Ergebnissen trug nicht nur maßgeblich zum Gelingen der Arbeit bei, sondern zeigte mir ganz allgemein neue Wege der Erkenntnisgewinnung und des wissenschaftlichen Arbeitens. Herzlich bedanken möchte ich mich auch bei meinem Zweitgutachter, Prof. Dr. Johann Schlichter, für seine äußerst hilfreichen Anmerkungen, die zu einer Erhöhung der Qualität der vorliegenden Arbeit beigetragen haben. Bei beiden Gutachtern möchte ich mich auch für die sehr zügige Bearbeitung früherer Versionen der Arbeit bedanken.

Auf Seiten des Kooperationspartners Deutsche Telekom resp. DeTeSystem will ich mich insbesondere bei den direkten Projektteilnehmern der Forschungskooperation „Abrechnungsmanagement“ bedanken: Dr. Boris Gruschke, Dr. Stephen Heilbronner, Dr. Birgit Kaltenmorgen und Dr. Bernhard Neumair. Sowohl die sehr hilfreichen Diskussionen als auch die mir gewährten Einblicke in reale Szenarios bildeten die entscheidenden Impulse für die in dieser Arbeit entwickelte Lösung.

Herzlicher Dank gebührt auch den aktiven und ehemaligen Kolleginnen und Kollegen des Munich Network Management Teams (MNM Team), die ein herausragendes Arbeitsklima geschaffen haben und damit das Fundament für eine erfolgreiche, wissenschaftliche Arbeit gelegt haben. Insbesondere möchte ich mich bei Michael Brenner, Vitalian Danciu, Dr. Christian Ensel, Markus Garschhammer, Iris Hochstatter, Dr. Rainer Hauck, Bernhard Kempter, Dr. Axel Küpper, Dr. Helmut Reiser, Harald Rölle und Dr. Holger Schmidt für das Lesen von Vorversionen, die zahlreichen Diskussionen und Ratschläge während der Erstellung der Arbeit bedanken. Auch den zahlreichen Studenten, die mich im Rahmen ihrer Diplomarbeiten, Fortgeschrittenenpraktika und Systementwicklungsprojekte unterstützt haben, sei an dieser Stelle gedankt.

Gesondert will ich mich für das Lesen von Vorversionen der Arbeit bei meiner Schwester Angela bedanken, die so manchen Rechtschreib– und Grammatikfehler in dieser Arbeit aufgespürt hat.

Nicht zuletzt danke ich meiner Familie, insbesondere meinen Eltern, meinen Freunden und besonders Silke für die moralische Unterstützung sowie den Rückhalt, auf den ich mich stets verlassen konnte.

München, im Januar 2003

Als Buch erhältlich im Verlag Dr. Hut, München, www.dr.hut-verlag.de (ISBN 3-934767-97-4)

Zusammenfassung

Um die eigene Wettbewerbsfähigkeit gegenüber Konkurrenzunternehmen zu erhöhen, konzentrieren sich heutige Unternehmen zunehmend auf ihre Kernkompetenzen und lagern häufig branchenfremde Teile, wie den Betrieb der Netzinfrastruktur und den darauf realisierten IT–gestützten Diensten, an externe Diensterbringer aus (Outsourcing). Da in diesen Fällen meist eine an den Kundenwünschen orientierte, maßgeschneiderte Diensterbringung erfolgt, wird ebenfalls eine maßgeschneiderte Abrechnung gefordert. Dies beinhaltet neben der Vereinbarung von dienstgüte– und nutzungsorientierten Tarifen beispielsweise auch eine kundenindividuelle Rechnungsstellung. Konsequenterweise erfolgt die Zusammenstellung der Abrechnungslösung, die aus unterschiedlichen HW/SW–Komponenten verschiedener Hersteller besteht, ebenfalls für jeden Kunden maßgeschneidert. Da sich bisher weder Standards für Nutzungs– noch für Managementschnittstellen (sofern überhaupt vorhanden) durchgesetzt haben, erweist sich das Abrechnungsmanagement als besonders aufwändig und fehleranfällig: die an der Abrechnung beteiligten Komponenten werden bisher immer verschieden und isoliert gesteuert und überwacht. Diese Tatsache erweist sich als besonders schwerwiegendes Problem, da häufig auftretende Änderungsaktivitäten nur unzureichend von bestehenden Managementlösungen unterstützt werden und meist manuell von Administratoren durchgeführt werden.

Um diesem Problem beizukommen wird in der vorliegenden Arbeit ein strikt prozessorientierter, auf policy–basierten Konzepten beruhender Ansatz für ein integriertes, dienstorientiertes Abrechnungsmanagement entwickelt. Der grundsätzliche Ansatz sieht vor, dass die vorgangsrealisierenden Komponenten nicht mehr isoliert gemanaged werden, sondern der Vorgang als solcher aus Prozesssicht durch Spezifikation von Managementpolicies gesteuert und überwacht wird. Da der grundlegende Ansatz nicht auf das dienstorientierte Abrechnungsmanagement beschränkt ist, wird bei der Entwicklung der Lösung auf Anwendbarkeit in anderen Managementfunktionsbereichen, wie dem Fehler–, Leistungs–, Sicherheits– und Konfigurationsmanagement, geachtet. Der Abrechnungsvorgang sowie das dazugehörige Abrechnungsmanagement dienen somit lediglich zur vertiefenden Darstellung der Managementproblematik, indem hierzu der Abrechnungsvorgang detailliert analysiert wird, um neben zahlreichen Prozessmodellen auch ein statisches Abrechnungsdienstmodell zu entwickeln. Im Rahmen der entwickelten Lösung wird sowohl eine Policy–Sprache als auch eine geeignete Managementarchitektur entworfen, die es ermöglichen, einen IT Prozess, wie den Abrechnungsvorgang, nach der Spezifikation von Policies automatisiert zu managen. Die entworfene Managementarchitektur stellt sicher, dass existierende Komponenten und Anwendungen integriert werden. Damit gelingt es insbesondere bisher durchgeführte Investitionen in diesem Bereich zu schützen. Ein weiteres Ergebnis der vorliegenden Arbeit ist eine allgemeine Methodik zur Spezifikation von prozessorientierten Managementpolicies, welche den Policy–Ersteller bei dessen Aufgabe anleitet. Eine prototypisch implementierte Managementanwendung dient als Tragfähigkeitsnachweis des entwickelten Lösungsansatzes.

Als Buch erhältlich im Verlag Dr. Hut, München, www.dr.hut-verlag.de (ISBN 3-934767-97-4)

Summary

IT has become a critical factor for corporate success for companies of all industrial sectors. To achieve cost–efficient and effective provisioning of IT services many companies decide to concentrate on their core business and hand over the operation of their IT infrastructure to third parties (outsourcing). In the same way planing, realizing and providing IT services is customized according to customer needs, a customized accounting management is demanded. This includes among others the support of QoS–dependent, usage–based tariffs as well as customer–individual billing. Consequently, the accounting system, which provides the customized accounting functionality, is individually assembled by combining various HW/SW–components of different manufacturers. As a matter of fact, accounting management proves all but easy-to-handle, as standards for usage/management interfaces are either not yet adopted or not even specified. Thus, accounting management is realized as an error–prone, isolated, component–oriented solution in contrast to a preferable integrated management approach. Especially change management activities, which happen to be carried out very frequently during service operation and have enormous effects on service accounting, are not yet sufficiently supported by any known management solution.

To overcome the afore mentioned problems, this thesis proposes the application of process–oriented, policy–based concepts to design an integrated, service–oriented accounting management. The basic idea is to manage any given system, like an accounting system, along the complete service life cycle by applying a process–oriented view. As the basic approach of applying policies to manage IT processes is not restricted to service oriented accounting management, we ensured the applicability of the proposed solution to other management domains like fault, performance, security and configuration management. Thus, accounting management serves solely for an in–depth process analysis of a given management domain leading to numerous process models as well as a static accounting service model. Afterwards, we develop a policy language as well as a management architecture which enable an automated, process oriented IT management after policy specification. We ensured that the management architecture can easily integrate already existing components. Furthermore, a methodology for declaring both policies as well as meta–policies is specified as user guidance. Finally, a prototypical implementation of the designed management application is presented as a proof–of–concept.

Als Buch erhältlich im Verlag Dr. Hut, München, www.dr.hut-verlag.de (ISBN 3-934767-97-4)

INHALT

 Als Buch erhältlich im Verlag Dr. Hut, München, www.dr.hut-verlag.de (ISBN 3-934767-97-4)

 Als Buch erhältlich im Verlag Dr. Hut, München, www.dr.hut-verlag.de (ISBN 3-934767-97-4)

Kapitel 1

Einleitung

1.1 Motivation

Die zunehmende Vernetzung von Unternehmen erhöht die Relevanz der Verfügbarkeit der damit verbundenen Systeme: Die Wirtschaftlichkeit eines Unternehmens ist in kritischer Art und Weise abhängig vom effizienten und fehlerfreien Betrieb der netzrealisierenden Infrastruktur. Mit der wachsenden Vernetzung steigt aber auch die Komplexität der Aufgabe, dies zu bewerkstelligen. Aus diesem Grund werden Bereitstellung, Unterhalt und Betrieb der Netzinfrastruktur immer häufiger an Dritte ausgelagert (auch als *Outsourcing* bezeichnet), die sich auf diese Aufgaben spezialisiert haben. In entsprechenden Verträgen (*Service Level Agreements, SLAs*) zwischen Auftraggeber und Dienstleister wird festgelegt, welche Leistung zu welchem Preis und zu welcher Dienstgüte zu erbringen ist.

Seit kurzem ist zusätzlich ein weiterer Trend zu beobachten: Die Netzbetreiber übernehmen nicht mehr nur die Vernetzung nationaler und internationaler Unternehmensstandorte für ihre Kunden, sondern auch den Betrieb und das Management von anwendungsorientierten Diensten, wie z.B. DNS Server, Email Server, WWW Server, E-Commerce Server, etc. Neben der damit verbundenen Zunahme der Komplexität des Betriebs [BHP+ 00] werden ebenfalls neue Anforderungen an das Management des Dienstleisters gestellt.

Diese neue Ausrichtung verstärkt die sich abzeichnende Entwicklung, dass die bisher üblichen Tarife für die erbrachte Dienstleistung nicht mehr vertretbar sind. Diese sehen i.d.R. eine pauschale Abrechnung, d.h. einen festen Betrag im Abrechnungsintervall, oder eine volumenorientierte Abrechnung auf sehr niedrigem Niveau vor, wie z.B. pro übertragenem Byte oder pro Zeiteinheit. Statt dessen wird von Kundenseite gefordert, dass die Abrechnung in Abhängigkeit weiterer Leistungsmerkmale erfolgt. Dies kann beispielsweise bedeuten, dass neben dem entstandenen Dienstnutzungsvolumen auch die während der tatsächlichen Dienstnutzung durch den Betreiber zur Verfügung gestellte Dienstgüte einbezogen wird. Allgemein wird gefordert, dass Tarife die in den Verträgen festgesetzten Vereinbarungen sowohl bzgl. der Dienstgüte als auch bzgl. der Dienstfunktionalität in geeigneter Art und Weise widerspiegeln und somit auch Vertragsstrafen und Rabatte miteinbeziehen. Folglich sollen die bisher einfach anwendbaren

Tarife durch aufwändiger strukturierte Tarife ersetzt werden. Weiterhin besteht durch die zunehmende Individualisierung der durch den Dienstleister erbrachten Dienste von Seiten des Kunden ebenfalls der Anspruch, kundenindividuelle Tarife vom Dienstleister zu beziehen. Dies bedeutet, dass unabhängig von den Tarifen, die der Dienstleister mit seinen bisherigen Kunden vereinbart hat, der Kunde an der Tarifgestaltung beteiligt ist und Einfluss auf den zu vereinbarenden Tarif nehmen kann. Schließlich kann festgestellt werden, dass von Seiten des Dienstleisters gewünscht wird, Tarife für bestimmte ausgewählte Dienste kurzfristig auch nur für eine bestimmte Kundengruppe ändern zu können, um sich z.B. dem Markt anzupassen und damit konkurrenzfähig zu bleiben[1].

1.1.1 Dienst– und Kundenorientierung: Neue Aspekte für die Abrechnung

Mit der zunehmenden Dienst– und Kundenorientierung sieht sich folglich der Dienstleister auf dem Gebiet der Abrechnung vor neue Herausforderungen gestellt. War bisher der Vorgang der Abrechnung[2], der die notwendigen Aktionen und Interaktionen zur erfolgreichen Erstellung der Rechnung zusammenfasst, einfach gehalten, wird dieser durch die sich neu ergebenden Anforderungen wesentlich aufwändiger. In der folgenden Aufstellung sind die wichtigsten neuen Aspekte den bisher üblichen Gegebenheiten gegenübergestellt sowie die sich daraus ergebenden Auswirkungen auf den Abrechnungsprozess angegeben:

- *Nutzungsorientierte, verursachergerechte Abrechnung vs. pauschale Abrechnung*
  Die pauschale Abrechnung wird als ungerecht empfunden. Die Kunden wollen lediglich für diejenige Leistung bezahlen, die sie tatsächlich auch in Anspruch genommen haben [AlCh 01]. Damit muss z.B. das entstandene Dienstnutzungsvolumen gemessen und nach den tatsächlichen Verursachern aufgeschlüsselt werden.
- *Dienstorientierte Abrechnungseinheiten vs. einfach messbare Einheiten*
  Die Einheiten, in denen die Dienstnutzung gemessen wird, müssen der Funktionalität des Dienstes entsprechen. D.h., dass insbesondere die anwendungsorientierten Dienste z.B. nach Transaktionen abgerechnet werden und nicht mehr nur nach verursachtem Datenvolumen. Es werden damit neue Abrechnungseinheiten verwendet, deren Messung und Bewertung ermöglicht werden muss.
- *Einbezug der Dienstgüte vs. dienstgüteunabhängige Abrechnung*
  Da die beim Dienstleister eingekauften Dienste zunehmend „mission critical" für die Kunden sind, ist die tatsächlich erbrachte Dienstgüte im Vergleich zur vereinbarten Dienstgüte von hoher Relevanz für den Kunden. Wird diese wesentlich unterschritten und ist damit die Wirtschaftlichkeit des eigenen Unternehmens in Gefahr, so muss sich dies auch auf

[1] Dieser Sachverhalt kann momentan z.B. bei Mobilfunkbetreibern beobachtet werden, die z.T. im Monatsrhythmus die Mobiltelefontarife ändern.
[2] Im Weiteren wird synonym auch der Begriff *Abrechnungsprozess* verwendet.

 Als Buch erhältlich im Verlag Dr. Hut, München, www.dr.hut-verlag.de (ISBN 3-934767-97-4)

den Preis auswirken. Als Folge muss die während der Dienstnutzung tatsächlich erbrachte Dienstgüte gemessen und in die Abrechnung miteinbezogen werden.

- *Kundenindividuelle Tarife vs. Standardtarife*
  Bislang ist es eine übliche Verfahrensweise, dass die angebotenen Dienste mit Standardtarifen abgerechnet werden [Odly 01]. Demnach kann der Kunde explizit nicht über die Bestandteile des Tarifs mit dem Provider verhandeln (z.B. Abrechnung eines bestimmten Dienstes nach Zeit anstatt nach Volumen, etc.). Desweiteren wird von gegenwärtigen Abrechnungssystemen nur mangelhaft unterstützt, kundenindividuell Rabatte auf Basis anderer Kriterien wie Auftragsvolumen, Nutzungsvolumen, etc. miteinzubeziehen. Insgesamt ist eine in allen Belangen individuelle Vereinbarung von Tarifen kaum vorgesehen.
- *Hohe Änderungsdynamik vs. selten vorkommende Änderungen*
  Eine übliche Vorgehensweise war bislang, dass ab dem Zeitpunkt des Vertragsabschlusses keine Änderungen mehr an abrechnungsrelevanten Daten vorgenommen wurden. Dies ist insbesondere bei Großkundenszenarios nicht mehr vertretbar: Da in diesen Fällen die Auftraggeber (z.B. ein Unternehmen) nicht mit den tatsächlichen Dienstnutzern (z.B. die Mitarbeiter) übereinstimmen, ergeben sich durch Mitarbeiterfluktuation, Abteilungswechsel, etc. erhebliche Auswirkungen auf die Abrechnungsdaten. Weiterhin treten ähnlich hohe Anforderungen an die Änderungsdynamik bzgl. Rechnungsdaten (wechselnde Empfänger, Adressen, Abrechnungsintervalle etc.) und Tarifdaten (Änderung des Gültigkeitszeitraums von Tarifen, etc.) auf.

Bereits anhand dieser wenigen Punkte wird offensichtlich, dass der bisherige, einfach gehaltene Abrechnungsprozess durch die zunehmende Dienst– und Kundenorientierung und die daraus resultierenden Anforderungen wesentlich komplexer wird. *Abrechnungssysteme* setzen den Abrechnungsprozess durch eine geeignete Implementierung um, während das *Abrechnungsmanagement* sich mit der Steuerung des Abrechnungsprozesses und damit des Abrechnungssystems beschäftigt. Dadurch, dass der Abrechnungsprozess durch die Dienstorientierung wesentlich mehr Variationen in dessen Ablauf vorsieht, muss das Abrechnungsmanagement ebenso flexibler in dessen Steuerung werden. Die vorliegende Arbeit beschäftigt sich mit der Konzeption eines Ansatzes für das Abrechnungsmanagement, der den Anforderungen, die aus der Dienst– und Kundenorientierung resultieren, genügt.

1.1.2 Defizite bisheriger Ansätze

Vielfältige Gründe bilden die Ursache dafür, dass es bislang nicht gelungen ist, die Steuerung der am Abrechnungsvorgang beteiligten Komponenten und Systeme aus Sicht des Abrechnungsmanagements in der Weise zu gestalten, dass diese den aufgestellten Anforderungen genügen [Radi 02].

Einerseits vereinen heutige Abrechnungssysteme meist das Management und die Prozessimplementierung in einem. Dadurch, dass bis vor kurzem nur sehr einfache Dienste — mitunter Standarddienste — angeboten wurden, war der Ablauf des Abrechnungsprozesses ebenfalls einfach gehalten: Jedem (Standard)dienst wurde ein Standardtarif zugewiesen, der allen Kunden gleichermaßen angeboten wurde[3]. Sowohl das Abrechnungssystem als auch das dazugehörige Abrechnungsmanagement wurde auf diesen einfach gehaltenen Abrechnungsprozess hin optimiert. Da der Abrechnungsprozess für Standarddienste keine besondere Flexibilität erfordert, ist folglich dessen Steuerungsmöglichkeit und somit auch dessen Management beschränkt. Dies resultiert in der gegenwärtig zu beobachtenden Gegebenheit, dass Abrechnungssysteme nur in eingeschränkter Art und Weise die notwendige, aus der Dienstorientierung resultierende Flexibilität bzgl. der Tarifgestaltung, der Dienstauswahl und den Änderungsmöglichkeiten im laufenden Betrieb unterstützen.

Zudem kann beobachtet werden, dass von den gegenwärtigen Abrechnungssystemen immer jeweils nur ein Teil des Abrechnungsprozesses umgesetzt wird. Meist beschränkt man sich auf die Berechnung des in Rechnung zu stellenden Betrags und auf die Rechnungsausstellung und –versendung. Die Ermittlung des Dienstnutzungsvolumens wird vorausgesetzt und demnach nicht berücksichtigt. Hier setzen weitere, eigenständige Produkte auf, die auf das Messen der Dienstnutzung sowie das Sammeln und Aggregieren der Daten spezialisiert sind. Damit wird also derzeitig der Abrechnungsprozess von unterschiedlichen, verteilten Komponenten realisiert. Die Tatsache, dass bislang weder Schnittstellen zum Austausch der Daten noch Datenformate für die Abrechnung standardisiert bzw. existierende Standardisierungen von den Herstellern nicht übernommen worden sind, weist auf ein weiteres Problem hin: Die Integration der Komponenten zu einer Gesamtlösung für die Abrechnung wird momentan durch Unterstützung proprietärer Schnittstellen erreicht bzw. durch Implementierung geeigneter Gateways. Insbesondere wird das Management dieser Komponenten bisher nur unzureichend betrachtet, so dass bspw. keinerlei Vereinheitlichung zur Steuerung der Komponenten existiert. Folglich werden die Komponenten bisher einzeln und unabhängig von den anderen am Abrechnungsprozess beteiligten Komponenten über deren proprietäre Schnittstellen gesteuert. Somit ist v.a. aus Managementsicht eine integrative Architektur erforderlich.

Weiterhin fehlt es insgesamt an einer umfassenden Unterstützung des Administrators beim Management des Abrechnungsvorgangs: Aus der oben bereits angesprochenen Problematik der nicht einheitlichen Steuerung der Abrechnungskomponenten resultiert zusätzlich eine fehlende Automatisierung von wiederkehrenden Managementvorgängen. Eine derartige Automatisierung ist gerade dann notwendig, wenn der Abrechnungsprozess, wie im Falle der Dienstorientierung, aufwändiger wird. Insbesondere nimmt durch die steigende Änderungsdynamik die Anzahl der immer wiederkehrenden Vorgänge in der Weise zu, dass die Automatisierung von Managementvorgängen Voraussetzung dafür ist, die Komplexität des Abrechnungsmanagements handhabbar zu machen.

Zusammengefasst besteht die Notwendigkeit nach einem flexiblen, integrierten, möglichst automatisierten Abrechnungsmanagement, das, unabhängig von Verlauf und Variation des Abrechnungsprozesses und dessen Implementierung, diesen zu steuern vermag.

[3] Historisch liegt dies u.a. darin begründet, dass die Abrechnungssysteme bis vor kurzem v.a. im Telekommunikationsmarkt eingesetzt wurden, in dem meist nur der Telefoniedienst als Abrechnungsleistung vorgesehen war.

 Als Buch erhältlich im Verlag Dr. Hut, München, www.dr.hut-verlag.de (ISBN 3-934767-97-4)

Ziel dieser Arbeit ist es, ein solches flexibles, integriertes Abrechnungsmanagement zu konzipieren.

1.2 Fragestellung und Lösungsansätze

Wie bereits im vorhergehenden Abschnitt skizziert wurde, erfüllen die gegenwärtigen Lösungen und Ansätze für das Abrechnungsmanagement nicht die Anforderungen, die aus der zunehmenden Dienstorientierung resultieren. Es wird einerseits ein, bezogen auf die prozessimplementierenden Komponenten, *integrativer Ansatz* und andererseits ein, bezogen auf die Unterstützung vielfältiger Abrechnungsmöglichkeiten, *flexibler Ansatz* benötigt. Desweiteren sollen wiederkehrende Managementvorgänge möglichst automatisiert ablaufen, um den Manager zusätzlich zu entlasten. Ziel ist es, jede beim Abrechnungsvorgang auftretende Aktivität durch geeignete Managementaktionen so gut wie möglich durch steuernde Eingriffe zu unterstützen.

Wie im Laufe dieser Arbeit noch gezeigt wird, erweist sich in den heutigen Szenarios v.a. die hohe Änderungsdynamik bezogen auf die abrechnungsrelevanten Daten als größte Problematik im Abrechnungsmanagement. Dies gründet sich darauf, dass nicht davon ausgegangen werden kann, dass die Implementierung des Abrechnungsprozesses aus einem Guss ist. Statt dessen wird eine Abrechnungslösung durch die Integration von verteilten Komponenten von z.T. verschiedenen Herstellern verwirklicht, um u.a. hohe Investitionen in bereits bestehende Legacy–Systeme und –Anwendungen zu sichern. Somit ist nach vorgenommenen Änderungen u.a. zu prüfen, ob eine Synchronisation der komponentenlokalen Datenbestände erforderlich ist, um Inkonsistenzen zu vermeiden. Zusammenfassend müssen demnach insbesondere die Auswirkungen von durchgeführten Änderungsaktivitäten von einem geeigneten Abrechnungsmanagement unterstützt werden.

Beispiel: In einem Unternehmen wird der Internetzugang von einem Internet Service Provider (ISP) betrieben. Den Mitarbeitern wird die private Nutzung des Internets grundsätzlich erlaubt, allerdings auf eigene Kosten. Um die entstandenen Kosten verursachergerecht auf die Mitarbeiter zu verteilen, müssen dem ISP die Nutzer bekannt gemacht werden. Bei einem Mitarbeiterneuzugang muss eine Aktualisierung der Daten nicht nur bei der Komponente, die das Nutzungsvolumen erfasst, stattfinden, sondern ebenfalls auch bei den Komponenten, die die Kostenberechnung und Rechnungstellung durchführen. Wird die Konsistenz der Daten nicht gewahrt, können z.B. Probleme bei der Zuordnung des Dienstnutzungsvolumens auf Kostenstellen entstehen. Eine denkbar ungünstige Auswirkung wäre, wenn das durch den neuen Mitarbeiter entstandene Nutzungsvolumen anderen Mitarbeitern unberechtigterweise zugeordnet wird.

Um den Forderungen nach Integration und Flexibilität nachzukommen, wird in der vorliegenden Arbeit die Anwendung einer strikt *prozessorientierten Sicht* in Kombination mit *policy–basierten Konzepten* auf das Abrechnungsmanagement als Lösungsansatz vorgeschlagen. Die prozessorientierte Sicht auf die Abrechnung und das Abrechnungsmanagement bietet für den Manager eine auf das Wesentliche reduzierte, integrative Darstellung der Managementaufgabe. Um neben einer reinen Darstellung auch die tatsächliche Integration verschiedener, an der Abrechnung beteiligter Komponenten zu erreichen, wird eine *Integrationsschicht* zwischen Managementanwendung und zu managenden Entitäten eingeführt, die eine, aus Managementsicht, Vereinheitlichung der Schnittstellen schafft. Der eben skizzierte Lösungsansatz ist grundsätzlich *nicht* auf das dienstorientierte Abrechnungsmanagement beschränkt, sondern kann ebenso in anderen Managementfunktionsbereichen, wie dem Fehler–, Leistungs–, Sicherheits– und Konfigurationsmanagement eingesetzt werden. Auf diesen Sachverhalt wird bei der Entwicklung der tatsächlichen Lösung innerhalb der Arbeit geachtet.

Für die Formulierung von Managementaktivitäten innerhalb des Abrechnungsmanagements werden *Policies*, die man sich als regelähnliche Ausdrücke in einer auf das Management spezialisierten (Skript–)Sprache vorstellen kann, verwendet. Hierbei stellt sich die implizite Abstraktion der Policies als weiterer Vorteil heraus. Dem Manager wird zum Ausdruck von Managementaktivitäten eine einzige, policy–basierte Schnittstelle auf das Abrechnungsmanagement geboten statt wie bisher viele, durch die Einzelkomponenten realisierte Schnittstellen. Ein *Policy–Interpreter* realisiert autark die Abbildung der Policies auf die einzelnen Managementschnittstellen der Komponenten, so dass der eigentliche Managementvorgang vollkommen automatisiert abläuft. Zudem bietet das Policy–Konzept die Möglichkeit, Managementwissen explizit und in formaler Art und Weise zu speichern. Im Laufe dieser Arbeit wird noch gezeigt, dass über die abrechnungsrealisierenden Komponenten hinweg auch ein einheitlicher Rollen– und Domänenbegriff benötigt wird. Auch in diesem Fall kann auf bereits existierende Ansätze im policy–basierten Management zurückgegriffen werden. Werden von einer an der Abrechnung beteiligten Komponente gewisse, aus der Dienstorientierung geforderten Konzepte nicht implementiert (wie z.B. das Domänenkonzept), so ergibt sich ein weiterer Vorteil des Einsatzes von Policies: Die Managementanwendung, die die Policies durchsetzt, kann die fehlende Unterstützung durch eine eigene Implementierung ersetzen. Der geforderte flexible Charakter der Lösung ist ebenfalls gewährleistet und ist in diesem Fall direkt von der Mächtigkeit der Policy–Sprache und der Anzahl der durch Policies gesteuerten Teilprozesse abhängig.

Um die Automatisierung der Managementvorgänge zu erreichen, werden die Abhängigkeiten von Managementaktivitäten untereinander untersucht und anschließend formal durch Spezifikation einer Policy beschrieben. In der vorliegenden Arbeit wird eine strukturierte Identifizierung dieser Abhängigkeiten dadurch erreicht, dass die Verknüpfungen der Managementaktivitäten in der Dimension des Dienstlebenszyklus und in der Dimension der Abrechnungsprozesse betrachtet werden. Um eine noch höhere Automatisierung der Durchführung von Managementaktivitäten zu erreichen, wird zusätzlich das Konzept von *Meta–Policies* eingesetzt. Wie noch im Laufe dieser Arbeit gezeigt wird, kann die Auswirkung einer Änderungsaktivität in einer Respezifikation von existierenden, bereits spezifizierten Policies resultieren. Da Meta–Policies

 Als Buch erhältlich im Verlag Dr. Hut, München, www.dr.hut-verlag.de (ISBN 3-934767-97-4)

in deren allgemeinster Definition die Erstellung, Aktivierung, Ausführung, Deaktivierung und das Löschen von Policies steuern, kann das Konzept der Meta–Policies insbesondere eingesetzt werden, um derartige Änderungsaktivitäten möglichst automatisiert von der zu entwickelnden Managementlösung zu unterstützen.

Damit ergeben sich insgesamt die folgenden Teilfragestellungen, die in dieser Arbeit behandelt werden:

- Was bedeutet dienstorientiertes Abrechnungsmanagement? In diesem Zusammenhang muss zunächst geklärt werden, welche Funktion das Abrechnungsmanagement einnimmt und wie dessen Abhängigkeit zu anderen Managementdisziplinen ist.
- Welche Anforderungen werden an das dienstorientierte Abrechnungsmanagement gestellt? Um dies möglichst umfassend zu beantworten, müssen u.a. reale Szenarios unter dem Abrechnungsgesichtspunkt betrachtet werden.
- Wie sieht der Status Quo auf dem Gebiet des Abrechnungsmanagements sowohl in der Forschung als auch in der Industrie aus? Welche Ansätze und Konzepte können übernommen werden? Warum erfüllen bisherige Ansätze nicht die Anforderungen, die aus der Dienst– und Kundenorientierung resultieren?
- Wie bereits in diesem Abschnitt erläutert wurde, wird eine prozessorientierte Sicht auf die Abrechnung und das Abrechnungsmanagement gewählt, um eine integrative Sicht aus dem Blickwinkel des Managements zu erreichen. Daraus ergeben sich die Fragen: Welche Teilprozesse realisieren den Gesamtprozess Abrechnung? Wie sind die Verbindungen zwischen den Teilprozessen und welche Auswirkungen hat dies auf das Management? Wie sieht der Dienstlebenszyklus aus Sicht des Abrechnungsmanagements aus?
- In diesem Abschnitt wurde bereits kurz der Lösungsansatz skizziert, der auf policy–basierten Konzepten beruht. Ziel ist es, ganz allgemein IT Prozesse durch Policies zu steuern, um damit einerseits dem Manager eine einheitliche Managementschnittstelle zur Spezifikation von Managementaktivitäten zu bieten und andererseits Managementwissen explizit und in formaler Art und Weise zu notieren (im Sinne einer Wissensdatenbank). Daraus ergibt sich folgende Teilfragestellung: Wie sieht eine Sprache für Policies zur Steuerung von IT Prozessen aus? Können bereits vorhandene Sprachen verwendet bzw. erweitert werden?
- Es werden Meta–Policies verwendet, um eine zusätzliche Automatisierung von notwendigen Managementaktivitäten zu erreichen. Wie sieht eine geeignete Sprache für Meta–Policies aus? Können damit alle notwendigen Abhängigkeiten ausgedrückt werden? Wo liegen die Grenzen der Automatisierung?
- Kann eine Methodik zur Spezifikation von Policies und Meta–Policies für das prozessorientierte IT Management angegeben werden? Falls ja, kann diese durch geeignete Mechanismen und Werkzeuge unterstützt werden?
- Bezogen auf das konkrete Einsatzgebiet des dienstorientierten Abrechnungsmanagements stellt sich die Frage, welche Teilprozesse des Abrechnungsvorgangs auf welche Weise durch Policies gesteuert werden können.

- Wie sieht die Implementierung eines derartigen, auf Policy–Konzepten und einer Integrationsschicht basierenden Abrechnungsmanagementsystems aus? Wie kann in dieser Konstellation ein Interpreter für die Policies und Meta–Policies realisiert werden? Welche Auswirkungen hat dies auf die Architektur des Gesamtsystems?

1.3 Vorgehensmodell und Ergebnisse der Arbeit

Um einen möglichst weitreichenden Ansatz für das dienstorientierte Abrechnungsmanagement zu entwickeln, wird ein Top–Down–Vorgehen gewählt, das im Nachfolgenden beschrieben und in Abbildung 1.1 grafisch visualisiert ist. Die wichtigsten Ergebnisse der einzelnen Kapitel sind in der Abbildung jeweils grau hinterlegt.

In Kapitel 2 erfolgt eine umfassende Analyse von Anforderungen an das dienstorientierte Abrechnungsmanagement, die zum größten Teil aus realen Szenarios abgeleitet werden. Um die abrechnungsspezifischen Details des in Kapitel 2 vorgestellten Szenarios erklären zu können, wird zunächst eine einheitliche Nomenklatur eingeführt. Die einzelnen Begriffe werden unter anderem anhand des Dienstlebenszyklus sowie anhand des generischen MNM Dienstmodells [GHH+ 01] erläutert. Ergebnis dieses Kapitels ist ein *strukturierter Anforderungskatalog*, der als Bewertungsinstrument für die noch folgenden Kapitel verwendet werden kann.

In Kapitel 3 werden die bisherigen Arbeiten aus Forschung und Industrie untersucht. Hierzu werden neben wissenschaftlichen Arbeiten insbesondere relevante Dokumente aller einschlägigen Standardisierungsgremien analysiert. Jede Arbeit wird anhand der in Kapitel 2 identifizierten Teilfunktionen des Abrechnungsmanagements eingeordnet. Zum Schluss des Kapitels erfolgt durch Anwendung des Anforderungskatalogs aus Kapitel 2 eine Übersicht der bisher in diesem Bereich erreichten Ergebnisse.

Da in Kapitel 3 festgestellt wird, dass insbesondere die Anforderungen an das Abrechnungsmanagement bzgl. der Dynamik und Flexibilität nicht erfüllt werden, werden diese Sachverhalte in Kapitel 4 ausführlich untersucht. Hierzu wird ein *Prozessmodell* entworfen, das alle für das Abrechnungsmanagement relevanten Teilprozesse und die darin eingebetteten Aktivitäten enthält. Auf Basis der identifizierten Teilprozesse werden ausführende Akteure und Ein–/Ausgabeentitäten bestimmt, die letztendlich die Objekte des Managements bilden. Diese werden zu einem *abrechnungsorientierten Dienstmodell* zusammengeführt, das eine Erweiterung des generischen MNM Dienstmodells aus Kapitel 2 darstellt.

Kapitel 5 beschäftigt sich mit der Entwicklung eines neuen Ansatzes für das dienstorientierte Abrechnungsmanagement. Wie in Kapitel 3 gezeigt wird, fehlt neben der Unterstützung der Dynamik und Flexibilität eine einheitliche, umfassende Sicht auf das Abrechnungsmanagement. Bisher werden die in Kapitel 4 identifizierten Teilprozesse isoliert betrachtet und gesteuert, ohne das Zusammenspiel mit anderen Teilprozessen explizit zu berücksichtigen. Hierfür wird in

 Als Buch erhältlich im Verlag Dr. Hut, München, www.dr.hut-verlag.de (ISBN 3-934767-97-4)

Kapitel 5 ein neuer, auf bewährten policy–basierten Konzepten beruhender Ansatz entwickelt, der diese Anforderung erfüllt. Policies werden in dieser Arbeit eingesetzt, um einzelne Teilprozesse sowie Ketten von Teilprozessen zu steuern. Sofern Policies nicht mehr ausreichen, um eine Managementaufgabe auszudrücken, werden Meta–Policies eingesetzt. Dies ist insbesondere immer dann der Fall, wenn beispielsweise neue Policies kreiert bzw. bestehende gelöscht werden sollen. Da der grundlegende Ansatz des prozessorientierten IT Managements nicht auf das Abrechnungsmanagement beschränkt ist, wurde bei der Entwicklung der Lösung auf die Anwendbarkeit in anderen Managementfunktionsbereichen geachtet. In Kapitels 5 wird hierfür eine *Sprache für Policies und Meta–Policies* entwickelt. Um die Elemente der Sprache festzulegen, werden auf Basis bereits entwickelter Policy–Sprachen die Ergebnisse der Untersuchungen des vorangegangenen Kapitel 4 verwendet. Zudem wird eine *Methodik zur Spezifikation von Policies* im prozessorientierten IT Management entwickelt. Die Methodik basiert auf der Verwendung von Prozess– und Dienstmodellen, wie sie in Kapitel 4 für das Abrechnungsmanagement angefertigt wurden. Insbesondere Prozessaktivitäten dienen als Grundlage, um Policy–Muster für das Management von Teilprozessen zu spezifizieren. Zum Abschluss des Kapitels wird der *objektorientierte Entwurf* des policy–basierten Managementsystems beschrieben, welches u.a. sowohl die Verwaltung als auch automatische Durchsetzung von prozessorientierten Managementpolicies realisiert. Um eine Integration existierender (Abrechnungs–)Komponenten zu erreichen, sieht die Architektur eine *Integrationsschicht* vor, welche die Heterogenität der vorgangsrealisierenden Infrastruktur für die policy–basierte Managementanwendung transparent erscheinen läßt. Ergebnis ist die im Rahmen einer Package–Struktur durchgeführte Spezifikation von Schnittstellen und Klassen sowohl für die Managementanwendung als auch für die Integrationsschicht.

In Kapitel 6 wird der entwickelte Ansatz prototypisch implementiert. Wie die Anforderungsanalyse aus Kapitel 2 gezeigt hat, muss die Implementierung insbesondere sicher, flexibel und skalierbar sein. Der Einsatz der am Lehrstuhl Prof. Dr. H.-G. Hegering entwickelten Plattform für Mobile Agenten *Mobile Agent System Architecture (MASA)* [GHR 99] eignet sich hervorragend für dieses Vorhaben. Für die in Kapitel 5 entwickelte Sprache wird ein Interpreter auf Basis eines XML–Parsers implementiert. Um Komponenten außerhalb der MASA–Umgebung einzubinden, werden Integrationsagenten als Gateway realisiert. Hierfür sind jeweils generische Agenten verwirklicht worden, die für spezielle Anpassungen erweitert werden können.

In Kapitel 7 erfolgt schließlich eine Zusammenfassung der wichtigsten Ergebnisse der Arbeit und eine Darstellung weiterführender Fragestellungen, die sich aus den Untersuchungen ergeben haben.

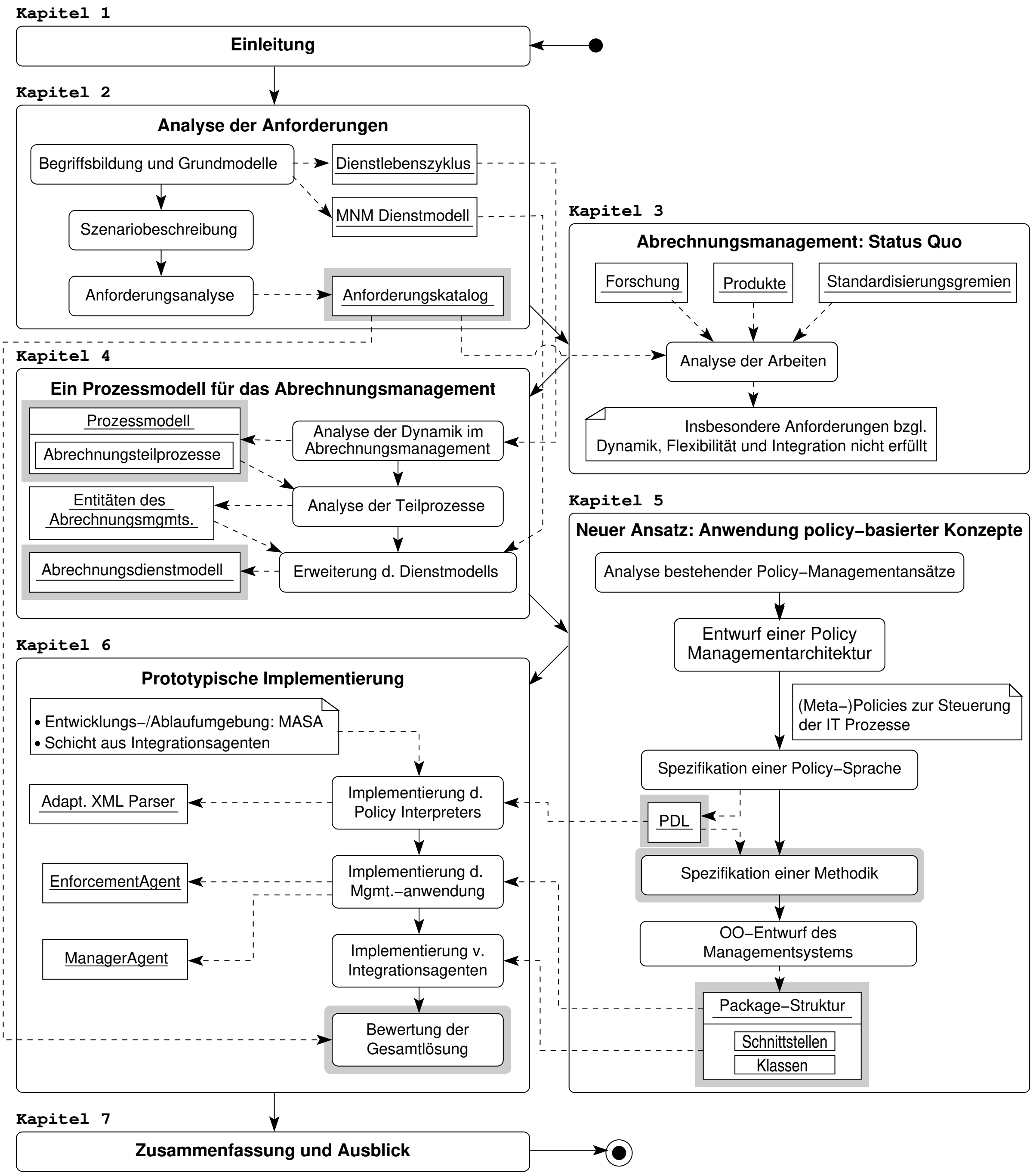

Abbildung 1.1: Vorgehensmodell und Ergebnisse dieser Arbeit

 Als Buch erhältlich im Verlag Dr. Hut, München, www.dr.hut-verlag.de (ISBN 3-934767-97-4)

Kapitel 2

Analyse der Anforderungen

Da das Dienstmanagement sowie das Abrechnungsmanagement noch Gegenstand der aktuellen Forschung sind, hat sich bisher noch keine einheitliche Terminologie in diesen Bereichen durchgesetzt. Aus diesem Grund erfolgt in diesem Kapitel zunächst eine Begriffsbildung. Es werden insbesondere all diejenigen Begriffe eingeführt und erläutert, die für die Anforderungsanalyse benötigt werden. Anschließend erfolgt eine Szenariodarstellung, um einerseits daran die Problematik des Abrechnungsmanagements vertiefend zu erklären und andererseits, um dies als Grundlage für die Herleitung der Anforderungen an das dienstorientierte Abrechnungsmanagement zu verwenden. Ergebnis dieses Kapitels ist ein Katalog, der die identifizierten Anforderungen in strukturierter Art und Weise enthält. Dieser Anforderungskatalog wird als Instrument zur Bewertung existierender Ansätze in Kapitel 3 eingesetzt.

2.1 Begriffsbildung und Grundmodelle

Zunehmend wird in der Fachpresse sowie bei Produktbeschreibungen von „Dienstmanagement“, „Service (Level) Management“, usw. geschrieben, ohne dass diesen Begriffen eine einheitliche Bedeutung zugeordnet wird. Dieser Umstand erschwert nicht unerheblich u.a. den Vergleich von existierenden Ansätzen und Produkten. Im Bereich des Abrechnungsmanagements vollzieht sich bzgl. der Terminologie eine ähnliche Beobachtung. Da sich das dienstorientierte Abrechnungsmanagement, wie es in dieser Arbeit verstanden wird, in das Dienstmanagement einbettet, erfolgt zunächst eine Einführung in das Dienstmanagement, welche mit einer Erläuterung der damit verbundenen Terminologie einhergeht. Darauffolgend werden Begriffe des Abrechnungsmanagements definiert und erklärt.

2.1.1 Einführung in das Dienstmanagement

Der Problematik der nicht einheitlichen Begriffsbildung im Forschungsbereich des Dienstmanagements widmet sich die in [GHH+ 01] beschriebene Arbeit. Hierbei wurde ein *Dienstmodell* (das sog. *MNM Dienstmodell*) entwickelt, das alle üblicherweise im Kontext des Dienstmanagements verwendeten Begriffe enthält und über Assoziationen miteinander in Beziehung setzt. Mit Hilfe dieses Dienstmodells ist es gelungen, jeden einzelnen Begriff unter Einbeziehung der in Relation stehenden Begriffe zu definieren und zu erklären und damit schließlich eine einheitliche Terminologie festzusetzen.

Im Folgenden wird analog zu [GHH+ 01] zunächst der *Dienstlebenszyklus* erklärt, der sich hervorragend dazu eignet, um im Rahmen des Abrechnungsmanagements im besonderen und im Rahmen des Dienstmanagements im Allgemeinen, als Strukturierungsmittel von Managementaufgaben, –rollen und –objekten zu dienen. Anschließend werden die Fachtermini in diesem Bereich anhand des Dienstmodells erklärt. Im Anschluss daran werden die wesentlichen Merkmale des *Dienstmanagements* erläutert, in das sich schließlich auch das dienstorientierte Abrechnungsmanagement einbettet.

Der Dienstlebenszyklus

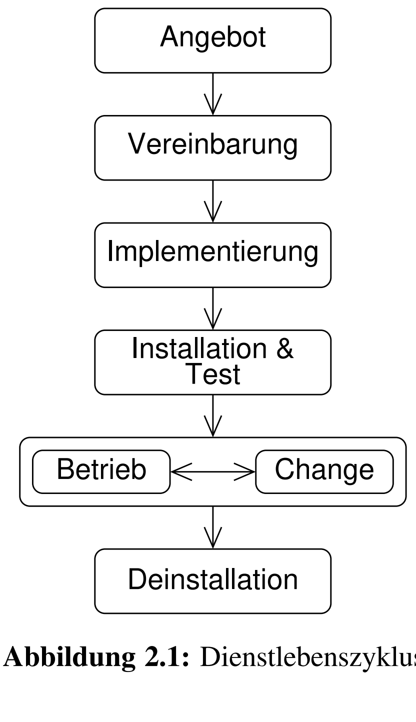

Abbildung 2.1: Dienstlebenszyklus

Wie bereits erwähnt, wird der Dienstlebenszyklus resp. dessen Phasen innerhalb dieser Arbeit als Mittel zur Strukturierung und Identifizierung von managementrelevanten Entitäten verwendet. Der Dienstlebenszyklus stellt allgemein anwendbar die grundsätzlichen Phasen eines beliebigen Dienstes dar, die dieser vom ersten Entwurf während der Planung bis zur vollständigen Entfernung durchlebt. Die gewählten Lebenszyklusphasen sind eine Verfeinerung der in [HAN 99a] vorgeschlagenen Phasen und finden sich in ähnlicher Art und Weise sowohl in TMForums Telecom Operations Map (siehe hierzu Abschnitt 3.2.4) als auch in der Dienstarchitektur des TINA Consortiums [TINA 97] wieder.

Im Weiteren werden insgesamt 7 *Lebenszyklusphasen* vorgestellt (siehe auch Abbildung 2.1). Der Lebenszyklus beginnt mit der *Angebotsphase*, in welcher der Dienstanbieter dem Dienstkunden ein Angebot über die Diensterbringung und seine Leistung unterbreitet. Im Allgemeinen prüft der Dienstkunde mehrere Angebote von verschiedenen Dienstanbietern. In der darauffolgenden *Vereinbarungsphase* wird ein Vertrag zwischen Kunde und Provider ausgehandelt. In der Regel endet diese Phase mit dem Abschluss eines Vertrags. Ein Vertrag enthält in aller Regel eine detaillierte Beschreibung der Dienstfunktionalität, der Dienstgüte und der anzuwendenden Tarife

 Als Buch erhältlich im Verlag Dr. Hut, München, www.dr.hut-verlag.de (ISBN 3-934767-97-4)

respektive Vertragsstrafen. Im Anschluss daran wird in der *Implementierungsphase* die vereinbarte Dienstfunktionalität durch den Provider realisiert. In der darauffolgenden *Installations– und Testphase* werden alle diejenigen Ressourcen, wie Geräte, Endsysteme, Applikationen, etc. in der Art und Weise installiert und getestet, dass der Dienst nach den vereinbarten Grundsätzen betrieben werden kann. Nachdem der Kunde den Dienst nach intensiver Prüfung abgenommen hat, wird der Dienst in *Betrieb* genommen. Neben der reinen Dienstbereitstellung werden vom Diensterbringer auch Managementaufgaben wie der Betrieb einer Hotline, eines User Help Desks, etc. übernommen. Hierzu zählt auch die Abrechnung des Dienstes nach den in der Vereinbarung getroffenen Richtlinien. Das heißt, dass z.B. die Dienstnutzung gemessen und einem Verursacher zugeordnet sowie die Gebühr für ein gemessenes Nutzungsvolumen berechnet wird. Anschließend muss eine Rechnung erstellt und an den Dienstnehmer verschickt werden. Die *Change–Phase* läuft parallel zur *Betriebsphase* und fasst alle Aktivitäten zusammen, die mit Änderungen der Dienstfunktionalität, der Dienstimplementierung und des Dienstmanagements einhergehen. Der Dienstlebenszyklus endet schließlich mit der *Deinstallationsphase*, in der die durch die Implementierung belegten Ressourcen freigegeben und die Konfiguration innerhalb des Dienstmanagement rückgängig gemacht werden.

Das MNM Dienstmodell

Mit dem in [GHH+ 01] eingeführten und in [GHK+ 01] weiterentwickelten Dienstmodell werden zwei grundsätzliche Ziele verfolgt: Einerseits sollen die Begriffe, die im Umfeld der Dienstorientierung verwendet werden, möglichst durch eine klare Herleitung definiert werden. Hierzu wurden Entitäten, Rollen und Interaktionen durch Anwendung des Dienstlebenszyklus strukturiert untersucht und in Beziehung zueinander gesetzt. Andererseits soll das Dienstmodell zur Modellierung von konkreten Szenarios eingesetzt werden.

In Abbildung 2.2 ist das Dienstmodell abgebildet, dessen Elemente nachfolgend erklärt werden:

- *Dienst*
  Unter einem Dienst soll im Weiteren Funktionalität verstanden werden, die einem *Dienstnehmer* an einer Schnittstelle mit gewissen *Dienstgütemerkmalen* von einem *Dienstleister* zur Verfügung gestellt wird. Die angebotene Funktionalität umfaßt dabei sowohl Nutzungs- als auch Managementfunktionalität.

- *Funktionalität*
  Die Funktionalität eines Dienstes wird als Menge von *Interaktionen* verstanden, die zwischen der Dienstleister– und Dienstnehmerseite ausgetauscht werden. Die Interaktionen können z.B. auf Anwendungstransaktionen, Protokolltransaktionen, aber auch auf Workflows abgebildet werden. Die Dienstfunktionalität wird in die Nutzungs– und Managementfunktionalität aufgeteilt, die nachfolgend erklärt werden:

  – *Nutzungsfunktionalität*
    Die Nutzungsfunktionalität umfaßt alle Interaktionen zwischen der Dienstleister– und

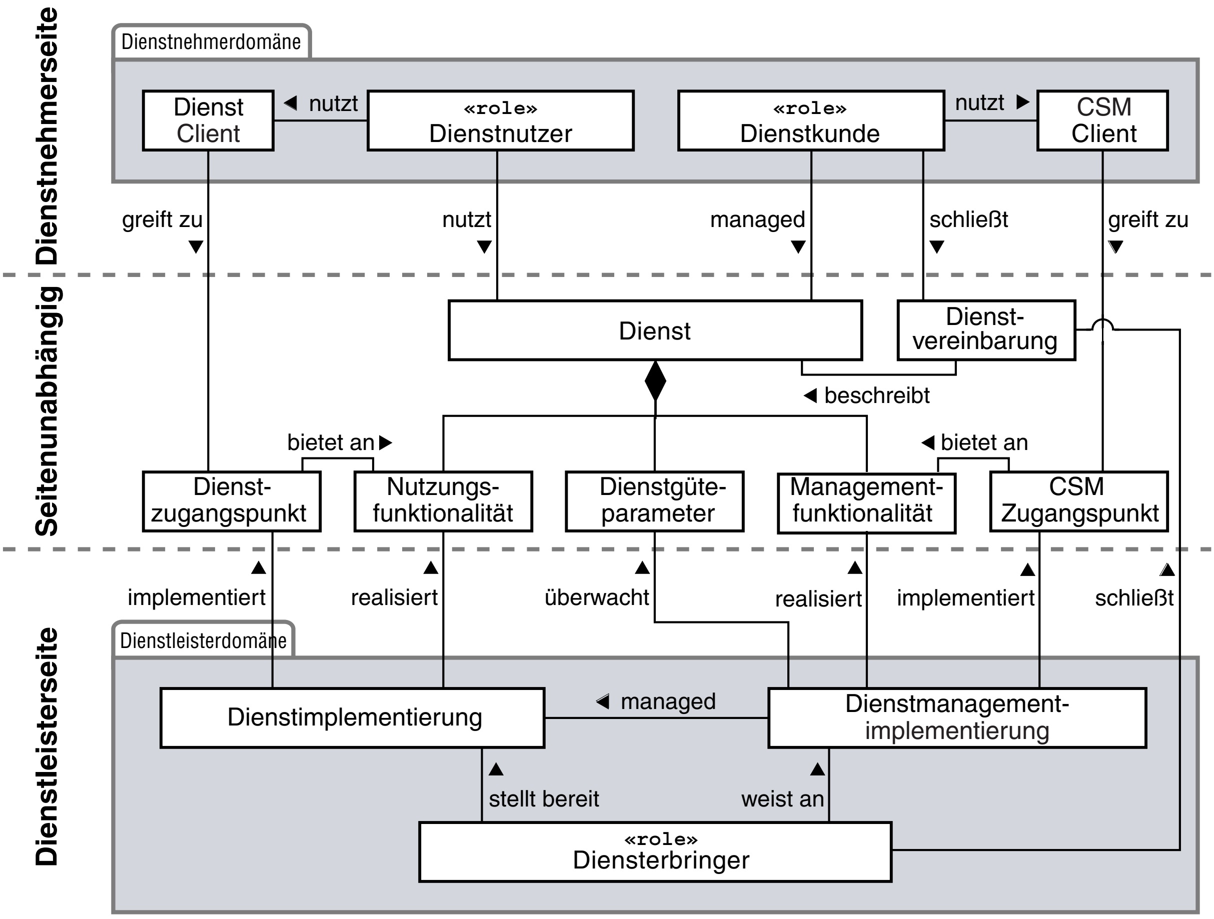

Abbildung 2.2: Dienstmodell nach [GHK+ 01]

Dienstnehmerseite, die den eigentlichen Zweck des Dienstes darstellen. Die Nutzungsfunktionalität wird vom *Dienstnutzer* in Anspruch genommen.

– *Managementfunktionalität*
Alle übrigen zwischen Dienstnehmer– und Dienstleisterseite stattfindenden Interaktionen, die notwendig sind, um den Dienst zu betreiben, aber nicht dem eigentlichen Zweck des Dienstes zugeordnet werden können, werden zur Managementfunktionalität zusammengefasst. Typischerweise werden dazu User Support, der Betrieb einer Hotline, die Rechnungsstellung und –zahlung, etc. gezählt. Hierunter fällt allerdings nicht das für die erfolgreiche Dienstbereitstellung notwendige, auf Dienstnehmerseite „intern“ betriebene Dienstmanagement, sondern nur die Managementfunktionalität, die für den Kunden über die CSM Schnittstelle nutzbar und damit sichtbar ist.

• *Dienstgüteparameter*
Die Güte, mit der die Dienstfunktionalität erbracht werden soll, wird mit Hilfe der Dienstgüteparameter beschrieben. Hierbei ist zu betonen, dass für beide Arten der Funktionalität Dienstgüteparameter definiert werden. Bei Vertragsabschluss werden in der Re-

 Als Buch erhältlich im Verlag Dr. Hut, München, www.dr.hut-verlag.de (ISBN 3-934767-97-4)

gel mit Hilfe der Dienstgüteparameter Schranken für die noch tolerierbare Dienstgüte durch Festlegung konkreter Werte spezifiziert. Erfolgt keine Wertefestlegung bezüglich der Dienstgüteparameter, so wird der Dienst nach dem Best–Effort–Prinzip betrieben.

- *Schnittstellen*
  Die *Dienstschnittstelle* umfasst alle diejenigen Entitäten, welche notwendig sind, damit von Dienstnehmerseite aus die Funktionalität des vereinbarten Dienstes genutzt werden kann. Analog zur Aufteilung der Dienstfunktionalität wird die Dienstschnittstelle in den *Dienstzugangspunkt* und in die *Customer Service Management (CSM) Schnittstelle* aufgeteilt:

  – *Dienstzugangspunkt*
    Am Dienstzugangspunkt steht dem *Dienstnutzer* (kurz: Nutzer) die vereinbarte Nutzungsfunktionalität zur Verfügung.

  – *Customer Service Management Schnittstelle*
    An der *Customer Service Management* Schnittstelle steht dem *Dienstkunden* (kurz: Kunde) die vereinbarte Managementfunktionalität zur Verfügung. Die CSM Schnittstelle kann bspw. mittels Programmierschnittstelle, Email, Telefon, etc. realisiert sein, über welche die Managementinteraktionen zwischen Kunde und *Diensterbringer* ausgetauscht werden können.

- *Dienstvereinbarung*
  Die Dienstvereinbarung enthält eine Beschreibung der Dienstfunktionalität sowie die Festlegung der dazugehörigen Dienstgüteparameter. Zusätzlich werden auch die Dienstschnittstellen spezifiziert, an denen von Dienstnehmerseite aus die beschriebene Funktionalität in Anspruch genommen werden kann. Die Dienstvereinbarung wird zwischen Diensterbringer und *Kunde* im Rahmen eines juristischen Vertrages geschlossen. In [Schm 01] wird die Strukur, der Inhalt sowie die Vorgehensweise zur Spezifikation von Dienstvereinbarungen detailliert behandelt.

- *Domänen*
  Innerhalb des Dienstmodells werden Domänen verwendet, um Zuständigkeitsbereiche festzulegen und zu beschreiben. Alle Entitäten, die sich innerhalb einer Domäne befinden, sind zwar unbedingt notwendig, um die Diensterbringung zu ermöglichen, aber nur für die Objekte innerhalb einer Domäne von Interesse.

  – *Dienstleisterdomäne*
    Die Dienstleisterdomäne fasst alle diejenigen Entitäten zusammen, die zur Erbringung des in der Dienstvereinbarung beschriebenen Dienstes notwendig sind. Insgesamt ist der *Diensterbringer* für die Bereitstellung des Dienstes verantwortlich, wofür dieser eine Dienstimplementierung und ein Dienstmanagement betreibt. Die Verantwortung des Diensterbringers für die Diensterbringung endet an den Schnittstellen des Dienstes.

- *Dienstnehmerdomäne*
  Auf Seiten des Dienstnehmers können grundsätzlich zwei unterschiedliche Rollen festgestellt werden:
  * *Dienstnutzer*
    Der Dienstnutzer kann über den Dienstzugangspunkt die Nutzungsfunktionalität des Dienstes in Anspruch nehmen.
  * *Dienstkunde*
    Der Dienstkunde abonniert den Dienst, schließt eine Dienstvereinbarung mit dem Diensterbringer und überwacht die Diensterbringung. Für die Überwachung des Dienstes nutzt er u.a. die Managementfunktionalität des Dienstes.

- *Client*
  Um die Funktionalität des Dienstes in Anspruch nehmen zu können, werden auf Dienstnehmerseite Clients sowohl für den Zugriff auf den Dienstzugangspunkt als auch auf den CSM Zugangspunkt benötigt. Die Verantwortung für den korrekten Betrieb der Clients liegt in jedem Fall im Aufgabenbereich des Dienstnehmers. Um allerdings den erfolgreichen Zugriff auf die Schnittstellen sicher zu stellen und damit eine mögliche Fehlerquelle auszuschließen, wird dennoch häufig eine genaue Beschreibung der Clients (z.B. die Browserversion) in die Dienstvereinbarung mit aufgenommen.

- *Dienstimplementierung*
  Die Dienstimplementierung, die vom Diensterbringer betrieben wird, realisiert primär die Nutzungsfunktionalität des vereinbarten Dienstes. Zusätzlich wird auch die Nutzungsschnittstelle implementiert, um den Zugriff auf die Funktionalität zu ermöglichen. Die Dienstimplementierung ist nicht nur rein technisch zu sehen: Hierbei handelt es sich um die Kombination des gesamten Wissens, des Personals sowie der Hard– und Software, die für die Dienstrealisierung erforderlich ist.

- *Dienstmanagementimplementierung*
  Die Dienstmanagementimplementierung umfasst alle erforderlichen Maßnahmen und Ressourcen, die sicherstellen, dass der Dienst in der Art und Weise geplant, installiert und betrieben wird, dass zu keinem Zeitpunkt gegen die Dienstvereinbarung verstoßen wird. Um beispielsweise die Einhaltung der vereinbarten QoS–Werte zu garantieren, müssen die Dienstgüteparameter überwacht und Schwellwerte für Managementaktionen gesetzt werden. Zusätzlich wird die CSM Schnittstelle implementiert, mit der es dem Dienstkunden ermöglicht wird, die vereinbarte Managementfunktionalität des Dienstes zu nutzen.

Zusätzlich wird in [GHH+ 01] die Beziehung zwischen Diensterbringer und *Sub–Diensterbringern* untersucht, die an der Bereitstellung des an den Dienstnehmer verkauften Dienstes beteiligt sind. In diesem Fall werden Teile der Dienstimplementierung respektive der Dienstmanagementimplementierung des Diensterbringers an weitere (Sub–)Diensterbringer ausgelagert. Grundsätzlich wird die Beziehung zwischen Diensterbringer und Sub–Diensterbringern in rekursiver Art und Weise auf die Dienstnehmer–Diensterbringer Beziehung

 Als Buch erhältlich im Verlag Dr. Hut, München, www.dr.hut-verlag.de (ISBN 3-934767-97-4)

des MNM Dienstmodells zurückgeführt. Damit ist eine Einführung neuer Rollen, um diese Beziehung auszudrücken, nicht notwendig. Auf eine tiefergehende Betrachtung dieser Assoziation wird in diesem Abschnitt verzichtet und auf [GHH+ 01] sowie [GHK+ 01] verwiesen.

In [GHH+ 02] wird eine Methodik zur Anwendung des Dienstmodells für die Modellierung konkreter Szenarios im Dienstmanagement vorgestellt, für die in [Schm 02a] eine workflow–orientierte Werkzeugunterstützung implementiert wurde.

Merkmale des Dienstmanagements: Dienst– und Kundenorientierung

Die *Dienstorientierung* und die *Kundenorientierung* sind die auschlaggebenden neuen Aspekte in der Dienstnehmer–Diensterbringer–Beziehung, die hauptsächlich die Ursache für neue Anforderungen an das technische Management bilden. Die sich daraus ergebende neue Managementdisziplin wird als Dienstmanagement bezeichnet.

Dienstorientierung bedeutet zunächst die klare Trennung der Funktionalität von der Implementierung eines Dienstes. Bei der Dienstvereinbarung zwischen Dienstnehmer und Diensterbringer wird lediglich die Dienstfunktionalität, d.h. die Nutzungs– und Managementfunktionalität, festgelegt. Die tatsächliche Implementierung der vereinbarten Funktionalität bleibt vollkommen dem Diensterbringer überlassen[1]. Der Vorteil dieser klaren Trennung besteht darin, dass der Diensterbringer die Implementierung eines Dienstes ändern kann, um z.B. Optimierungen durchzuführen, ohne dass eine Neuverhandlung der Dienstvereinbarung notwendig ist. Ein weiterer Vorteil ist, dass es für den Dienstnehmer nicht mehr notwendig ist, Implementierungsdetails in der Dienstvereinbarung zu verhandeln. Weiterhin bedingt die Dienstorientierung das Vereinbaren einer Dienstgüte, die der Diensterbringer bei der Bereitstellung des Dienstes beachten muss. Damit ist die Bereitstellung eines Dienstes nach dem *Best–Effort*–Prinzip nicht mehr ausreichend, um konkurrenzfähig zu bleiben. Der Vorteil für die Dienstnehmerseite liegt klar auf der Hand: Der Kunde kann sich mit höherer Dienstgüte zunehmend auf die bereitgestellte Dienstfunktionalität verlassen. Damit ist sogar ein Auslagern von unternehmenskritischen, IT–gestützten Geschäftsprozessen (*Business Process Outsourcing (BPO)*) denkbar, ohne dass dies das Kerngeschäft eines Unternehmens ernsthaft gefährden würde. Die Problematik, die sich beim Diensterbringer hierdurch ergibt ist, dass dies das Management, insbesondere das Dienstgütemanagement, erheblich erschwert. Es ist bisher eine immer noch ungelöste Fragestellung, wie vorausgesagt werden kann, welche Änderungen an der Dienstimplementierung welchen Einfluss auf die Güte einer erbrachten Dienstfunktionalität haben.

Durch die Kundenorientierung in der Dienstnehmer–Dienstleister–Beziehung werden zunehmend *kundenindividuelle* Dienstmerkmale bezüglich der Funktionalität, der Dienstgüte und der Dienstschnittstellen gefordert. Damit ist ein Trend weg vom *Standarddienst* hin zum *Individualdienst* zu beobachten. Infolgedessen wird das Management für den Diensterbringer in allen Belangen aufwändiger, da z.T. die dienstrealisierende Infrastruktur (z.B. Backbone–Netz) für

[1] Es existieren zweifellos auch Szenarios, in denen der Kunde gewisse Teile der Implementierung bereits im Vertrag festlegt (wie z.B. die Wahl eines bestimmten Routers zur Unternehmensvernetzung).

alle Kunden teilweise gleich ist, aber die von Kundenseite geforderten Dienstmerkmale wesentliche Unterschiede aufweisen. Die vorliegende Arbeit beschäftigt sich in diesem Umfeld mit den Auswirkungen auf das Abrechnungsmanagement.

Im Weiteren dieser Arbeit wird der Begriff „dienstorientiert“ synonym für „dienst– und kundenorientiert“ verwendet.

2.1.2 Die Teilfunktionen des Abrechnungsmanagement

Ziel dieses Abschnitts ist neben der reinen Begriffsdefinition auch das Abrechnungsmanagement zu strukturieren und damit die im Rahmen dieser Arbeit relevante Teilbereiche zu identifizieren. Grundsätzlich unterstützt das *Abrechnungsmanagement* in seiner allgemeinsten Form den Diensterbringer bei allen Aktivitäten, die notwendig sind, um die *Abrechnung* der betriebenen Dienste in der Weise zu ermöglichen, wie es in der Dienstvereinbarung ausgehandelt wurde.

Im Folgenden wird eine Aufteilung des Abrechnungsvorgangs in mehrere Teilfunktionen genannt und erklärt. Aufgabe des Abrechnungs*managements*, wie es in dieser Arbeit verstanden wird, ist es, den Diensterbringer bei der Bereitstellung der Funktionalität dieser Teilfunktionen in der Weise zu unterstützen, dass die sich aus dem Trend der Dienst– und Kundenorientierung ergebenden Anforderungen (siehe Abschnitt 2.3) erfüllt werden.

Jeder Teilfunktion wird zusätzlich die gebräuchliche englische Bezeichnung in Klammern zugewiesen. Hierbei muss nochmals ausdrücklich betont werden, dass die nachfolgend genannten Begriffe in der Literatur keineswegs einheitlich verwendet werden. Die nachfolgende Begriffsdefinition stützt sich auf die Begriffsdefinitionen des OSI Managements [ISO 10164-10], der IETF [RFC 1272, RFC 2975] und auf [Song 99].

Kostenermittlung (cost allocation) Die Kostenermittlung beinhaltet die tatsächliche Berechnung der durch die Bereitstellung und Nutzung eines Dienstes entstehenden Kosten beim Diensterbringer. Hierzu werden die fixen und variablen Kosten aller beteiligten Komponenten (z.B. Verkabelungskosten, Netzkomponenten, Verbindungsstrecken, Server, Systemdienste, etc.) [HAN 99a] miteinbezogen. Idealerweise gelingt es, eine *Kostenfunktion* festzulegen, die in Abhängigkeit zur Dienstnutzung eines Nutzers die hierbei anteilig tatsächlich entstehenden Kosten angibt.

Tarifierung (pricing, tariffing) Dieser Vorgang beinhaltet Aktivitäten zur Festsetzung des Preises für die Nutzung eines Dienstes. Hierbei kann auch der Kunde beteiligt sein, der zusammen mit dem Diensterbringer einen *Tarif* vereinbart. Oft wird der Preis auch zur Steuerung des Nutzungsverhaltens benützt, so dass ein Tarif in Abhängigkeit von anderen Faktoren wie z.B. Tageszeit oder verfügbare Bandbreite festgesetzt wird. Als Einflussfaktoren für die Tarifierung kann sowohl die bei der Kostenermittlung bestimmte Kostenfunktion als auch eine vom

 Als Buch erhältlich im Verlag Dr. Hut, München, www.dr.hut-verlag.de (ISBN 3-934767-97-4)

Diensterbringer aufgestellte *Abrechnungspolitik* sein, die Regeln enthält, wie der Tarif für eine Dienstnutzung gestaltet wird. Weiterhin kann zudem der ermittelte Tarif folgendermaßen unterschieden werden:

- *Pauschaler Tarif*
  Ist der Tarif unabhängig von dem in Anspruch genommenen Dienstnutzungsvolumen (z.B. ausgedrückt durch Verbindungsdauer, übertragene Datenmengen, CPU–Auslastung, Speicherbelegung, Anzahl der Zugriffe) spezifiziert, so spricht man von einem pauschalen Tarif. Das heißt, dass dem Dienstnehmer immer ein fester Betrag für die Dienstnutzung und –bereitstellung in Rechnung gestellt wird. Dies ist gegenwärtig die am weitesten verbreitete Form des Tarifs.

- *Nutzungsorientierer Tarif*
  Enthält der Tarif eine Komponente, die in Abhängigkeit von dem in Anspruch genommenen Dienstnutzungsvolumen spezifiziert ist, so spricht man von einem nutzungsorientierten Tarif. Diese Tarifform liegt gegenwärtig im Trend und wird zunehmend von der Dienstnehmerseite gefordert. Der hierbei erstellte Tarif enthält diskret abzählbare *Abrechnungseinheiten*, die das durch den Dienstnutzer in Anspruch genommene Dienstnutzungsvolumen widerspiegeln. Die Abrechnungseinheiten können wiederum innerhalb des Tarifs in Abhängigkeit einer *Preisfunktion* auftreten. Die Preisfunktion bildet die Anzahl der Abrechnungseinheiten in Abhängigkeit von anderen Parametern wie Dienstgüte, Tageszeit, etc. auf eine tatsächliche *Gebühr* in einer Währung ab.

Der Vorgang der Tarifierung bzw. Tarifvereinbarung kann zusätzlich noch folgendermaßen unterschieden werden:

- *Statische Tarifierung* (*static pricing*)
  Wird der Tarif für eine Dienstbereitstellung einmalig z.B. in einer Dienstvereinbarung festgesetzt und ist dieser für einen vereinbarten Zeitraum und für mehr als eine Dienstnutzung gültig, so spricht man von der statischen Tarifierung. Dies ist die weitaus häufigste Form der Tarifierung.

- *Dynamische Tarifierung* (*dynamic pricing*)
  Wird der Preis für eine Dienstnutzung immer erst kurz vor der tatsächlichen Dienstnutzung berechnet bzw. ausgehandelt und ist dieser immer nur für eine einmalige Dienstnutzung gültig, so spricht man i.d.R. in diesem Fall bereits von der dynamischen Tarifierung. Zusätzlich existieren Verfahren, die es zulassen, dass der Preis während der eigentlichen Dienstnutzung noch geändert bzw. neuverhandelt werden kann. Einflussfaktoren für den Vorgang der Verhandlung des Tarifs sind meist Parameter wie Anzahl der Anfragen, zur Verfügung stehende Güte, etc. Hierbei wurden zahlreiche *Tarifmodelle* wie dem *smart market model*, *second price auction*, etc. (siehe hierzu auch Abschnitt 3.3.1) entwickelt, die neben der in einen Tarif aufzunehmenden Abrechnungseinheiten auch den Vorgang der Tarifvereinbarung bestimmen. Diese Art der Tarifierung tritt derzeit in der Praxis aufgrund

ihrer Komplexität und hohen Ressourcenanforderungen nicht auf und wird momentan nur in der Forschung eingesetzt, um die Auswirkungen bestimmter Tarife und Tarifmodelle auf das Nutzungsverhalten theoretisch zu untersuchen.

Nutzungserfassung (usage accounting) Die Nutzungserfassung umfasst alle Aktivitäten, die notwendig sind, um das Dienstnutzungsvolumen, das von Dienstnutzern eines Dienstnehmers in Anspruch genommen wurde, zu erfassen. Dieser Vorgang wird zwar bei pauschalen Tarifen für die Durchführung der Abrechnung nicht benötigt, findet häufig dennoch in diesen Fällen statt, um dies als Grundlage für die Kostenermittlung zu verwenden. Prinzipiell wird die Nutzungserfassung in folgende zwei Teilfunktionen getrennt:

- *Messen der Dienstnutzung* (*metering*)
  Hierunter ist die Ermittlung des durch eine Dienstnutzung in Anspruch genommenen *Nutzungsvolumens* zu verstehen. Hierbei werden primär diskret abzählbare *Messeinheiten* gezählt und, je nach Dienstvereinbarung, in Abhängigkeit von weiteren Informationen wie Dienstgüte, Messzeitpunkt, Verursacher–ID, etc. gespeichert. Da u.U. große Datenmengen bei diesem Vorgang entstehen, werden in diesen Fällen zusätzlich Filter und Komprimierungstools eingesetzt, um die Datenmenge zu reduzieren.
- *Sammeln der Messdaten* (*collecting*)
  Dieser Vorgang umfasst das Sammeln und Aggregieren der Messdaten von verschiedenen Messkomponenten. Aufgabe ist es, sogenannte *Usage Records* zusammenzustellen. Usage records enthalten das von einem bestimmten Dienstnutzer in einem gegebenen Zeitraum in Anspruch genommene Dienstnutzungsvolumen.

Gebührenberechnung (charging, rating) Bei der Gebührenberechnung werden die usage records eines Dienstnutzers für einen gegebenen Zeitraum aggregiert und die, für das in Anspruch genommene Dienstnutzungsvolumen fälligen Gebühren berechnet. Hierbei werden, falls unterschiedlich, die Messeinheiten auf die Abrechnungseinheiten des vereinbarten Tarifs abgebildet. Anschließend wird der Tarif angewendet, um die Gebühr zu errechnen. In aller Regel werden bei diesem Vorgang die Gebühren aller Dienstnutzer, die einem Dienstnehmer zugeordnet sind, berechnet und zu sogenannten *customer detailed records* zusammengefasst.

Rechnungsstellung (billing) Bei der Rechnungsstellung werden die Gebühren des customer detailed records eines Dienstnehmers aufsummiert und in der vereinbarten Form als *Rechnung* (*invoice*) zusammengestellt und an den/die Adressaten verschickt. Die Rechnung enthält, im Gegensatz zu *Reports*, in aller Regel lediglich den zu zahlenden Betrag mit einer Zahlungsaufforderung.

Inkasso (payment) Dieser Vorgang enthält alle Aktivitäten, die in Zusammenhang mit dem Zahlungsverkehr mit dem Kunden stehen. Beispielsweise besteht die Möglichkeit, dass der

 Als Buch erhältlich im Verlag Dr. Hut, München, www.dr.hut-verlag.de (ISBN 3-934767-97-4)

Kunde offenstehende Rechnungen auf herkömmliche Art und Weise per Bankeinzug/Lastschrift begleicht. Denkbar wäre aber auch, dass im Voraus ein Bestand an *credit points* vom Kunden eingekauft wird, der jeweils sofort bei einer Dienstnutzung entsprechend verkleinert wird, bis dieser aufgebraucht ist. Hierbei müssen auch Eskalationsmechanismen, wie z.B. Versenden von Mahnungen, Sperren von Dienstzugangspunkten, etc. gesteuert werden.

Die bisher beschriebenen Teilfunktionen werden in der vorliegenden Arbeit als notwendige Vorgänge der Abrechnung verstanden. Das Abrechnungsmanagement umfasst hingegen alle Managementaktivitäten, die erforderlich sind, um den Abrechnungsvorgang in der Weise zu betreiben, dass die Anforderungen aus der Dienst– und Kundenorientierung (siehe hierzu Abschnitt 2.3) erfüllt sind. War es bisher üblich, dass alle Kunden gleiche Dienste mit gleichen Tarifen erhalten haben, kommen neuerdings neue Aspekte wie kundenindividuelle Tarife und Rechnungen, dienstgüteabhängige Gebühren, flexible Rechnungsstellung, etc. hinzu, die ein weitaus flexibleres, auf den Kunden und den Dienst abgestimmtes Management verlangen. Im Folgenden werden die sich neu ergebenden Anforderungen anhand eines Szenarios erklärt.

2.2 Szenarioanalyse

Die vorliegende Arbeit konzentriert sich darauf, Großkundenszenarios hinsichtlich des Abrechnungsmanagements und die sich hierbei ergebenden Problemstellungen zu analysieren. Auch wenn die noch zu entwickelnde Lösung prinzipiell allgemein (bezogen auf Szenarios unterschiedlicher Größe) anwendbar ist, so werden v.a. die in Großkundenszenarios im Bereich des Abrechnungsmanagements anzutreffenden Probleme gelöst. Im Folgenden wird zunächst ein typisches Großkundenszenario beschrieben, um anschließend die herausragenden Charakteristika auf den Punkt gebracht zusammen zu fassen.

2.2.1 Großkundenszenario: Intranet Extranet Services Anbindung

Um eine Wiederholung ähnlicher Gegebenheiten bei der Beschreibung real existierender Szenarios im Großkundenbereich zu vermeiden, wird im Folgenden ein (fiktives) Szenario durch Zusammenführung mehrerer existierender Projektszenarios erstellt, die während des Kooperationsprojekts „Abrechnungsmanagement“ in Zusammenarbeit mit der *Deutschen Telekom Systemlösungen (DeTeSystem)*[2] [Radi 02d] untersucht wurden. Die Darstellung des Szenarios dient v.a. der detaillierten Darstellung heutiger Probleme im Abrechnungsmanagement und um in Abschnitt 2.3 darauf basierend systematisch Anforderungen an das Abrechnungsmanagement abzuleiten.

[2] Mittlerweile ist die DeTeSystem nach der Unternehmensfusion mit Debis vollständig in der neugegründeten *T–Systems* aufgegangen.

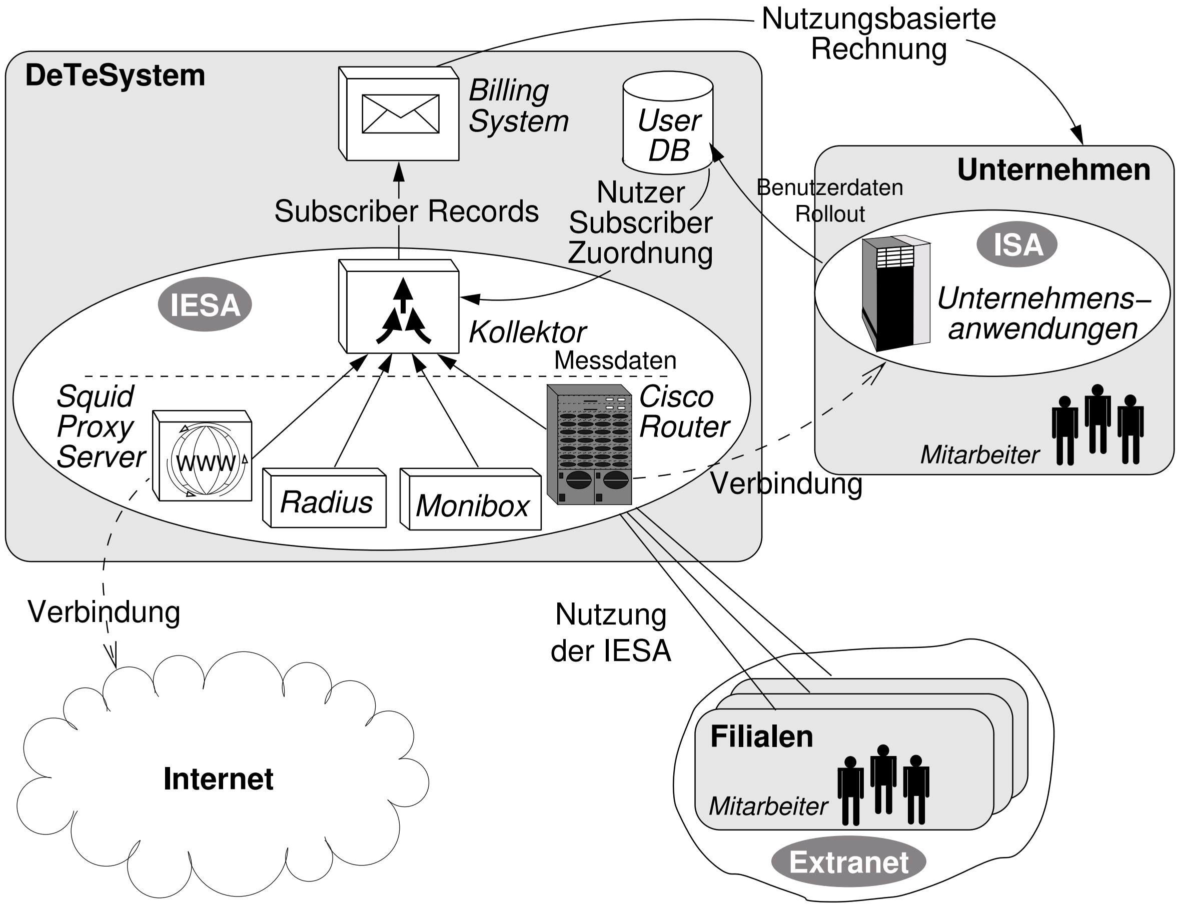

Abbildung 2.3: Ein Großkundenszenario aus Sicht des Abrechnungsmanagements

In Abbildung 2.3 ist ein typisches Großkundenszenario mit Schwerpunkt auf das Abrechnungsmanagement dargestellt. Die DeTeSystem betreibt für einen ihrer Kunden (ein international operierendes Unternehmen) eine sogenannte *Intranet Extranet Service Area (IESA)*. Als Schwerpunkt wird innerhalb der IESA für deren Nutzer ein Konnektivitätsdienst bereitgestellt, der einerseits eine Verbindung zum Intranet des Unternehmens bietet, das im Folgenden als *Intranet Service Area (ISA)* bezeichnet wird, und andererseits eine Verbindung zum Internet zur Verfügung stellt. Zusätzlich wird innerhalb der IESA für den Kunden ein Email– und Web–Hosting Dienst und ein DNS Server betrieben. Sowohl die Mitarbeiter innerhalb der *Unternehmensfilialen* als auch innerhalb der Konzernzentrale stellen die eigentlichen IESA–Dienstnutzer dar: Es werden Datenbanken und spezielle Unternehmensanwendungen, die innerhalb der ISA zusammgefasst sind, verwendet, die jeweils zugewiesenen Email Accounts genutzt und auch innerhalb der Unternehmensfilialen das Internet für den privaten Gebrauch genutzt. Auf Seiten der Filialen sind ISDN Dial–Up–Router installiert, die die Verbindung zur IESA und damit die Nutzung der zahlreichen, beschriebenen Dienste ermöglichen. Die Konzernzentrale sowie die ISA sind mit einer breitbandigen Standleitung zur IESA verbunden.

 Als Buch erhältlich im Verlag Dr. Hut, München, www.dr.hut-verlag.de (ISBN 3-934767-97-4)

Die initiale Dienstvereinbarung zwischen DeTeSystem und dem Kunden sah eine pauschale Abrechnung für die Hosting–Dienstleistung sowie für die ISA–IESA–Standleitung und eine zeitbasierte Gebührenberechnung für die aus den Filialen initiierte Internet– und ISA–Nutzung vor. Hierbei wurde für die ISA– und Internet–Nutzung die Anwendung unterschiedlicher Tarife vereinbart. Zusätzlich wurde vereinbart, dass die Nutzung des Internets den Mitarbeitern in den Unternehmensfilialen direkt in Rechnung gestellt wird. Um hierfür eine zuverlässige Verursacherzuordnung des gemessenen Dienstnutzungsvolumens (in diesem Fall die gemessene Zeit der Internetnutzung) zu ermöglichen, wurde ein regelmäßiger Daten–Rollout vereinbart, der insbesondere den Kunden dazu verpflichtete, zu den Rollout-Zeitpunkten eine aktuelle Aufstellung aller Filialenmitarbeiter und deren Abteilungs– und Standortzuordnung zu erstellen.

Um die vereinbarte Abrechnung zu bewerkstelligen, werden Einlogzeiten der Dial–In–Router mit den Messdaten des WWW Proxys und des ISA Routers kombiniert. Eine Kollektorsoftware sammelt hierfür in regelmäßigen Abständen die gemessenen Daten und aggregiert diese zu *subscriber records*. Die Aggregation sieht eine Kumulation der Messdaten über die Dienstnutzung zu einem Verursacher vor. Hierbei wird der kumulierte Wert einer *accounting user ID* zugeordnet, die den Verursacher innerhalb des Abrechnungssystems eindeutig identifiziert. Aufgrund der noch fehlenden Unterstützung von standardisierten Datenformaten und APIs, sofern Standardisierungen in diesem Bereich überhaupt existieren (siehe hierzu Kapitel 3), wird die Zusammenarbeit der an der Abrechnung beteiligten Komponenten (Messkomponenten, Kollektorsoftware und Billing System) durch Unterstützung der jeweiligen proprietären Schnittstellen und Datenformate der Komponenten erreicht. Da allerdings das Format, in dem die Einlog–Protokolle der Dial–In–Router vorlagen, von der Kollektorsoftware nicht unterstützt wird, wurde bei der DeTeSystem ein neues Modul implementiert, dass eine Transformation in ein konformes Format vorsieht. Die Implementierung eines Transformationsmoduls für das Einfügen der während des Rollouts übertragenen Benutzerdaten in die Datenbank des Abrechnungssystems war ebenfalls notwendig.

Schwierigkeiten ergaben sich nun durch die hohe Änderungsdynamik auf Seiten des Dienstnehmers. Das Kundenunternehmen fusioniert mit einem anderen Unternehmen. Hierbei änderten sich aufgrund der Neustrukturierung des Konzerns die Anforderungen an die ausgelagerten Dienste und damit auch an die bisher durchgeführte Abrechnung. Beispielsweise sollten die neu hinzugekommenen Unternehmensteile, wie z.B. die Filialen des Fusionspartners, bis zum endgültigen Vollzug der Fusion zwar an der IESA partizipieren können, aber, parallel zu den bisherigen Vereinbarungen, mit einem volumen– statt zeitbasierten Tarif abgerechnet werden. Hierfür war eine Neuprogrammierung der Kollektorsoftware notwendig, die auf Basis des Verursachers die Messdaten unterschiedlich kumuliert, da die Datenquellen und damit die gemessenen Daten dieselben geblieben sind. Eine Anpassung innerhalb des Rechnungsstellungssoftware (Billing System) war ebenfalls notwendig. Nach erfolgreicher Beendigung der Fusion und den abgeschlossenen Umstruktierungsmaßnahmen innerhalb des neu geschaffenen Unternehmens wurde auf Aufforderung der Leiter der Unternehmensfilialen, die letztendlich die ihnen ausgestellte Rechnung bezahlten, eine Kombination der bisher vereinbarten Tarife, d.h. zeit– und volumenbasiert, gefordert. Um einerseits dies in der geforderten Genauigkeit zu ermöglichen und andererseits eine unter Umständen in Zukunft geforderte dienstgüteabhängige Abrechnung zu unterstützen, war es notwendig, die Dienstnutzung auf Seiten der Nutzer, also in den Filialen, zu messen. Hierzu war eine Verteilung von Messsoftware auf die ISDN–Dial–Up Router sowie deren Konfiguration notwendig. Dies ist insbesondere bei mehr als 10.000 Filialen ein nicht zu unterschätzender Aufwand. Währenddessen wurde eine empfängerindividuelle Rechnungsstellung sowie Hot–Billing und –Reporting[3] gefordert, für welches das Billing System ausgetauscht werden musste und damit auch die vom Diensterbringer spezifisch implementierten Module. Zukünftig sollen der IESA weitere Anwendungsdienste hinzugefügt werden, für welche eigene Tarife vereinbart werden sollen. Hierfür sind dann neue Messkomponenten zu installieren bzw. unter Umständen sogar zu implementieren und in das bestehende Abrechnungssystem miteinzubinden.

Ebenfalls ergab sich auf Seiten der Unternehmensführung des Diensterbringers der Wunsch, Rabatte für gewisse Dienste und Dienstleistungen für alle Kunden gleichermaßen für einen gewissen Zeitraum zu geben, um die Kundenbindung zu erhöhen. Hierfür wäre eine Unterstützung einer auf bestimmte Dienste beschränkten, kundenübergreifenden Rabattierung durch das Billing System notwendig gewesen. Da aber mehrere Billing Systeme parallel eingesetzt wurden und diese teilweise die geforderte Funktionalität nicht unterstützten, musste dieser Schritt „manuell“ durch die verantwortlichen Administratoren durchgeführt werden.

Zusammenfassend liegt die besondere Schwierigkeit in der hohen Änderungsdynamik, die oftmals in einer umfassenden und weitreichenden Anpassung der bestehenden Abrechnungslösung durch den Diensterbringer, wie einer teilweisen Neuimplementierung, resultiert. Da bisher eine Unterstützung durch geeignete Managementwerkzeuge vollständig fehlt und damit Änderungen sowie deren Auswirkungen durch personellen Einsatz in Form der betreuenden Administratoren durchgeführt werden, ist das Abrechnungsmanagement in der bisherigen Form eine sehr fehleranfällige und zeitaufwändige Aufgabe. Damit besteht auf Seiten des Diensterbringers die Forderung nach geeigneten Managementwerkzeugen im Bereich des Abrechnungsmanagements, die insbesondere die Änderungsdynamik unterstützen.

2.2.2 Charakteristika des Szenarios

Anhand des im vorhergehenden Abschnitts detailliert beschriebenen Szenarios lassen sich folgende allgemeine Merkmale erkennen, die charakteristisch für Szenarios im Großkundenbereich sind und besondere Relevanz für die Abrechnung und das Abrechnungsmanagement haben:

1. *Individuelle Dienst– und Dienstmanagementbereitstellung*
   Die in Großkundenszenarios fast ausschließlich anzutreffende Art der Dienstbereitstellung ist die des *Individualdienstes*. Im Gegensatz zum *Standarddienst*, der eine für alle

[3] Als Hot–Billing und –Reporting bezeichnet man die (sofortige) Ausstellung von Rechnungen respektive Reports bei Anfrage des Kunden. Meist wird dies als Online–Dienst realisiert.

 Als Buch erhältlich im Verlag Dr. Hut, München, www.dr.hut-verlag.de (ISBN 3-934767-97-4)

Dienstnehmer identische Dienstnutzungs– und Dienstmanagementfunktionalität vorsieht, vereinbart der Kunde mit dem Diensterbringer eine individuelle Nutzungs– und Managementfunktionalität. Dies liegt in der Tatsache begründet, dass im Großkundenbereich die jeweiligen Anforderungen an die ausgelagerten Dienste und Dienstleistungen von Unternehmen zu Unternehmen, aufgrund von unterschiedlichen Unternehmenspolitiken und –strukturen, verschieden sind.

2. *Heterogene Zusammenstellung der Abrechnungslösung*
Die im vorhergehenden Punkt erwähnte individuelle Dienstmanagementbereitstellung betrifft insbesondere auch das Abrechnungsmanagement. Da Dienste und Tarife kundenindividuell vereinbart werden, werden ebenso in Abhängigkeit der vereinbarten Tarifen und der tatsächlichen Dienstrealisierung die Komponenten der Abrechnungslösung zusammen gestellt. Beispielsweise müssen in Abhängigkeit von der tatsächlichen Dienstimplementierung die Messkomponenten ausgewählt werden. Ebenso wird die Wahl der Rechnungsstellungssoftware von der vereinbarten Abrechnung beeinflusst. Damit ist der Betrieb einer monolithischen, für alle Kunden gleichen Abrechnungslösung weder möglich noch gewünscht.

3. *Lange Vertragslaufzeiten*
Da die Bereitstellung von Individualdiensten in der Regel mit hohem Ressourcenaufwand auf Dienstleisterseite und entsprechend hohem Kostenaufwand auf Dienstnehmerseite verbunden ist, werden entsprechend lange (mehrjährige) Vertragslaufzeiten [Schm 01] vereinbart. Nur dadurch ist auf beiden Seiten ein hoher *Return–on–Investment (ROI)* [BRS 02] möglich.

4. *Flexible Dienstvereinbarungen*
Aufgrund der langen Vertragslaufzeiten sind die Dienstvereinbarungen zwischen Kunde und Diensterbringer flexibel ausgelegt. Das heißt, die Verträge lassen Änderungen bzgl. der Dienstfunktionalität explizit zu (siehe hierzu auch [Schm 01]). Diese können unter Umständen zu einer Neuverhandlung von Teilen der Dienstvereinbarung führen.

5. *Hohe Änderungsdynamik*
Während der Vertragslaufzeit treten unausweichlich Änderungen während des laufenden Betriebs auf, die insbesondere auch direkt und indirekt Auswirkungen auf den Abrechnungsvorgang und das Abrechnungsmanagement haben. Hierbei können Änderungen sowohl auf Dienstnehmerseite, z.B. durch Mitarbeiterfluktuation, Restrukturierungsmaßnahmen der Unternehmensorganisation aufgrund von Business Process Reengineering (BPR), Fusionen, etc., als auch auf Dienstleisterseite, z.B. durch Änderungen an der Dienstimplementierung zum effizienteren Betrieb des Dienstes, festgestellt werden. Änderungen während des laufenden Betriebs sind somit in den betrachteten Szenarios nicht die Ausnahme, sondern im Gegenteil die Regel.

2.3 Entwicklung eines Anforderungskatalogs

In diesem Abschnitt werden Anforderungen an das dienst– und kundenorientierte Abrechnungsmanagement hergeleitet. Zur strukturierten Identifizierung von Anforderungen werden im folgenden Abschnitt zunächst allgemeine Anforderungen formuliert. Basierend darauf werden separat funktionale Anforderungen an den Abrechnungsvorgang, implementierungsorientierte Anforderungen an die abrechnungsrealisierenden Komponenten und schließlich Anforderungen an das Abrechnungsmanagement untersucht.

2.3.1 Allgemeine Anforderungen

Es ergeben sich die folgenden allgemeinen Anforderungen an ein dienst– und kundenorientiertes Abrechnungsmanagement:

ALL 1 *Unterstützung des vollständigen Abrechnungsvorgangs*
Es liegt auf der Hand, dass der Abrechnungsvorgang sowohl vollständig bzgl. der Funktionalität realisiert als auch diese Realisierung vom Management geeignet unterstützt werden muss.

ALL 2 *Klare Trennung des Abrechnungsvorgangs vom Abrechnungsmanagement*
Um eine klare Zuordnung von Aufgaben und damit optimierte Sichten für die Akteure, die diese Aufgaben übernehmen, zu realisieren, ist eine klare Trennung des Abrechnungsvorgangs vom Abrechnungsmanagement notwendig.

Insbesondere wegen der zuletzt genannten Anforderung sieht konsequenterweise die vorliegende Anforderungsanalyse eine Aufteilung in Anforderungen an den Abrechnungsvorgang (Abschnitt 2.3.2), an die abrechnungsrealisierenden Komponenten (Abschnitt 2.3.3) und an das Abrechnungsmanagement (Abschnitt 2.3.4) vor. Damit ist eine eindeutige Zuordnung der Anforderungen bereits in der Struktur des sich dadurch ergebenden Katalogs ersichtlich.

2.3.2 Anforderungen an den Abrechnungsvorgang

Im vorhergehenden Abschnitt 2.3.1 ist unter anderem die Unterstützung des vollständigen Abrechnungsvorgangs als Anforderung vorgesehen. Aus diesem Grund werden im Folgenden die bereits in Abschnitt 2.1.2 vorgestellten Teilfunktionen einzeln auf zu erfüllende Anforderungen hin analysiert. Wo es möglich ist, werden die Anforderungen nach Dienstnehmer– und Dienstleisterseite aufgeschlüsselt.

 Als Buch erhältlich im Verlag Dr. Hut, München, www.dr.hut-verlag.de (ISBN 3-934767-97-4)

Kostenermittlung

Anforderungen von Dienstleisterseite:

KOS 1 *Präzision*
Da die Kostenermittlung auf Dienstleisterseite eine wesentliche Rolle während der Tarifierung spielt und damit auch Voraussetzung für den unternehmerischen Erfolg des Diensterbringers ist, ist eine hohe Präzision bei der Berechnung bzw. Schätzung der Kosten für die Bereitstellung eines Dienstes gefordert.

Tarifierung

Anforderungen von Dienstnehmerseite:

TRF 1 *Nutzungsorientierung*
Tarife, die sich an der tatsächlichen Nutzung eines Dienstes orientieren, indem beispielsweise das in Anspruch genommene Nutzungsvolumen direkt in den Tarif mit einbezogen werden, werden als gerechter empfunden und damit von Dienstnehmerseite zunehmend gefordert.

TRF 2 *Dienstgüteorientierung*
Aus der Dienstorientierung heraus ergibt sich auf Dienstnehmerseite zunehmend die Forderung nach Tarifen, welche die vereinbarte und auch erfahrene Dienstgüte miteinbeziehen. Dies kann sich dann beispielsweise im automatischen Einbeziehen von Rabatten und Vertragsstrafen äußern, falls die vertraglich zugesicherte Dienstgüte vom Diensterbringer nicht eingehalten wird.

TRF 3 *Kundenindividuelle Tarifvereinbarung*
Es liegt auf der Hand, dass bei Individualdiensten auch kundenindividuelle Tarife gefordert werden, die jeweils nicht nur den Dienst–, sondern insbesondere auch den Kundencharakteristika entsprechen. Dies bedeutet auch, dass es dem Kunden ermöglicht werden sollte, den Transaktionsbegriff, der als Grundlage für bspw. die Abrechnungseinheiten und die Dienstgüte herangezogen wird, zu definieren.

TRF 4 *Dienstorientierte Abrechnungseinheiten*
Desweiteren ergibt sich aus der Dienstorientierung heraus die Forderung nach Abrechnungseinheiten, die sich an der Funktionalität und nicht an der Implementierung eines Dienstes orientieren. Dies heißt beispielsweise, dass nach Transaktionen und nicht nach übertragenem Datenvolumen in Bytes abgerechnet wird.

TRF 5 *Einfache, nachvollziehbare Tarifbestimmung*
Die zwischen Diensterbringer und Kunde vereinbarten Tarife müssen trotz der bereits erwähnten neuen Anforderungen weiterhin derart beschaffen sein, dass diese immer für die Dienstnehmerseite nachvollziehbar und damit auch nachprüfbar sind.

TRF 6 *Vorhersehbarkeit*
Die vereinbarten Tarife müssen aus Kundensicht derart gestaltet sein, dass die auf den Kunden zukommenden Gebühren vorhersehbar sind, um damit die eigenen Kosten abschätzen zu können.

Anforderungen von Dienstleisterseite:

TRF 7 *Kostendeckend*
Um die Wirtschaftlichkeit des Diensterbringerunternehmens sicher zu stellen, müssen die vereinbarten Tarife in aller Regel kostendeckend sein. Hier ist aus Dienstleistersicht erforderlich, so früh wie möglich die Kosten über den bereit zu stellenden Dienst prognostizieren zu können.

TRF 8 *Durchsetzbarkeit*
Es liegt auf der Hand, dass ein vereinbarter Tarif auch auf der gegebenen Infrastruktur durchsetzbar sein muss, da ansonsten eine Gebührenberechnung nicht erfolgen kann. Hierbei muss darauf geachtet werden, dass beispielsweise eine Messung der Dienstnutzung in der vereinbarten Art und Weise auch tatsächlich möglich ist.

Nutzungserfassung

Anforderungen von Dienstnehmerseite:

UAC 1 *Transparenz*
Die Nutzungserfassung und alle hierfür notwendigen Maßnahmen, wie bspw. die Installation von Messkomponenten, müssen vollkommen transparent für sowohl den Dienstnutzer als auch den Dienstkunden sein. Das heißt, dass insbesondere keine *zusätzlichen* Aktionen von Dienstnehmerseite notwendig sind, um den erfolgreichen Betrieb dieser Teilfunktion zu gewährleisten.

Die vom Diensterbringer zu realisierende Nutzungserfassung ist direkt von dem vereinbarten Tarif und der vereinbarten Art und Weise der Durchführung des Abrechnungsvorgangs abhängig. Damit fließen indirekt auch die bei der Tarifierung identifizierten Dienstnehmer–Anforderungen in die nachfolgenden, aus Sicht des Dienstleisters spezifizierten Anforderungen mit ein:

UAC 2 *Unterstützung einer hohen Messperformanz*
Der Performanzbegriff bezieht sich einerseits auf die Genauigkeit der Messung, d.h. dass beispielsweise Echtzeitmessungen ermöglicht werden, als auch auf einen möglichst geringen Einfluss des Messvorgangs auf die dienstrealisierende Infrastruktur, d.h. dass z.B. die Messdaten vor deren Weitergabe komprimiert werden.

UAC 3 *Flexibilität bzgl. des Messorts*
Je nach vereinbartem Tarif muss die Nutzungserfassung in unterschiedlichen Domänen stattfinden.

 Als Buch erhältlich im Verlag Dr. Hut, München, www.dr.hut-verlag.de (ISBN 3-934767-97-4)

UAC 4 *Unterstützung unterschiedlicher Messeinheiten*
Idealerweise können für unterschiedliche Dienste und vereinbarte Tarife dieselben Messeinrichtungen verwendet werden. Hierzu ist es notwendig, dass per se unterschiedliche Messeinheiten unterstützt werden.

Gebührenberechnung

Anforderung von Dienstnehmerseite:

CHA 1 *Unterstützung von On–Demand–Charging*
Es sollte eine auf Nachfrage sofort angestoßene Gebührenberechnung möglich sein. Dies ist u.a. für Hot–Billing notwendig.

Anforderung von Dienstleisterseite:

CHA 2 *Flexible, skalierbare Gebührenberechnung*
Je nach vereinbartem Tarifmodell, müssen die eingesammelten Daten über die Dienstnutzung vorverarbeitet werden. Hierbei sollte die Gebührenberechnung flexibel und erweiterbar bezüglich der Vorverarbeitungsfunktion sowie skalierbar bezüglich der Anzahl der anzuwendenden Tarife sein.

Rechnungsstellung

Anforderungen von Dienstnehmerseite:

BIL 1 *Unterstützung von Hot–Billing*
Unter Hot–Billing versteht man die sofortige, auf Kundenanforderung angestoßene Rechnungsausstellung zu prinzipiell beliebigen Zeitpunkten. Insbesondere bei nutzungsorientierten Tarifen wünscht der Kunde eine aktualisierte Aufstellung der bis zum Anforderungszeitpunkt entstandenen Kosten.

BIL 2 *Kundenorientierte Gestaltung des Rechnungsinhalts*
Je nach Verwendungszweck der Rechnung auf Dienstnehmerseite werden vom Kunden andere Anforderungen an den Inhalt der Rechnung und dessen Gestaltung gestellt. Damit muss von der Rechnungsstellung eine flexible Konfiguration der Inhaltsgestaltung in Abhängigkeit des Kunden bzw. des Empfängers ermöglicht werden.

BIL 3 *Unterstützung mehrerer Zustellungsarten*
Durch die zunehmende Verbreitung des Einsatzes von Informationstechnologien zur Erleichterung und Automatisierung von Unternehmensabläufen liegt es auf der Hand, dass ebenfalls gefordert wird, Rechnungen durch Nutzung dieser Infrastruktur an die Empfänger zuzustellen, um u.a. die Weiterverarbeitung der Rechnung in die Betriebsabläufe automatisch zu integrieren. Somit sollen Rechnungen nicht nur klassisch per Post

an die Empfänger zustellbar sein, sondern auch elektronisch, beispielsweise per Email oder Datenübertragung (z.B. EDI [EDI 96]).

Inkasso

Anforderung von Dienstnehmerseite:

INK 1 *Unterstützung unterschiedlicher Zahlungsarten*
Auch die Zahlung des in Rechnung gestellten Betrages sollte auf mehreren Wegen möglich sein, bspw. per Lastschrift, Überweisung, Kreditkarte, elektronischen Payment–Mechanismen (siehe hierzu auch [Webe 00]), usw.

Anforderung von Dienstleisterseite:

INK 2 *Unterstützung von Eskalationsmechanismen*
Bei Überschreiten von Zahlungsfristen sollte die Inkasso–Komponente automatisch Eskalationsmechanismen initiieren können (von Verschicken von Mahnungen bis Sperrung des Dienstzugangspunktes), um den Diensterbringer hiervon zu entlasten.

2.3.3 Anforderungen an die abrechnungsrealisierenden Komponenten

Nachfolgend werden im Unterschied zum vorhergehenden Abschnitt Anforderungen an die Komponenten, die den Abrechnungsvorgang realisieren, formuliert, die für alle Beteiligten gleichermaßen gelten. Die Anforderungen sind jeweils unter dem Managementgesichtspunkt abgeleitet worden, d.h. es wurde untersucht, was die an der Abrechnung beteiligten Komponenten idealerweise unterstützen sollten, damit diese besonders gut vom Management einbezogen werden können. Damit liegt der Fokus auf der Schnittstelle zwischen Abrechnungsmanagement und den Komponenten, die den Abrechnungsvorgang realisieren. Da sich die Anforderungen auf Komponenten beziehen, die sich ausschließlich in der Verantwortung des Diensterbringers befinden, sind folglich ohne Ausnahme alle identifizierten Anforderungen durch Analyse der Dienstleisterseite abgeleitet worden:

IMP 1 *Standardisierte Nutzungsschnittstelle*
Werden standardisierte Nutzungsschnittstellen (soweit vorhanden) von den Komponenten implementiert, um mit anderen Komponenten zu interagieren, so wird die Zusammenstellung einer heterogenen Abrechnungslösung erheblich erleichtert. Damit können auch Komponenten ausgetauscht und durch andere ersetzt werden, ohne den laufenden Betrieb zu stören. Der Einsatz einer standardisierten Nutzungsschnittstelle bezieht sich hierbei auf den Einsatz von standardisierten Protokollen, Datenformaten, APIs, etc. In Kapitel 3 erfolgt eine detaillierte Analyse der Standardisierungsbemühungen in diesem Bereich.

 Als Buch erhältlich im Verlag Dr. Hut, München, www.dr.hut-verlag.de (ISBN 3-934767-97-4)

IMP 2 *Standardisierte Managementschnittstelle*
Durch die Unterstützung von standardisierten Managementschnittstellen (soweit vorhanden) wird die Integration einer individuell zusammengestellten Abrechnungslösung in ein bestehendes Managementumfeld wesentlich erleichtert [HAN 99a]. Dadurch wird zudem die Realisierung eines integrierten Managements, im Gegensatz zu einem komponentenorientierten Management, unterstützt. In Kapitel 3 erfolgt ebenfalls eine detaillierte Analyse der Standardisierungsbemühungen in diesem Bereich.

IMP 3 *Domänenübergreifende Kooperation von unterschiedlichen Komponenten*
Wie bereits in Abschnitt 2.2.2 bei der Darstellung der besonderen Szenarioeigenschaften in Punkt 2 festgestellt wurde, ist jede Abrechnungslösung aus heterogenen Komponenten zusammengestellt. Damit müssen diese Komponenten in Kooperation den Abrechnungsvorgang realisieren, der unter Umständen sogar domänenübergreifende Interaktionen (zur Dienstnehmer– und/oder Sub–Dienstleisterdomäne) erfordert. Hierfür ist eine Unterstützung der Nutzungsschnittstellen derjenigen Komponenten notwendig, mit denen eine Komponente interagiert, sofern keine standardisierten Schnittstellen unterstützt werden.

IMP 4 *Sichere Interaktion zwischen beteiligten Komponenten*
Da gerade im Bereich der nutzungsorientierten Abrechnung einerseits datenschutzrelevante und damit sicherheitssensible Daten zwischen den an der Abrechnung beteiligten Komponenten ausgetauscht werden und andererseits sich ein Verfälschen der erfassten Daten direkt pekuniär auswirkt, muss die Sicherheit der Kommunikation zwischen den Komponenten gewährleistet werden. Dies kann beispielsweise durch eine verschlüsselte Kommunikation zwischen authentifizierten und für die jeweilige Aktion auch autorisierten Komponenten realisiert werden.

IMP 5 *Mandantenfähigkeit*
Um Komponenten nicht notwendigerweise dediziert einem Dienstnehmer zuordnen zu müssen, sondern im Rahmen von Optimierungsmaßnahmen Teile der individuell zusammengestellten Abrechnungslösung (wie z.B. die Rechnungsstellungskomponente) für andere Kunden wieder zu verwenden, muss die wiederverwendete Komponente mandantenfähig sein. Mandantenfähig bedeutet in diesem Zusammenhang, dass die Daten unterschiedlicher Dienstnehmer und deren Verarbeitung strikt voneinander getrennt werden. Damit soll vor allem sicher gestellt werden, dass keine sicherheitssensiblen Daten unberechtigten Dritten, wie z.B. einem Konkurrenzunternehmen, zugänglich gemacht werden.

IMP 6 *Verifizierbarkeit*
Voraussetzung für sowohl die Reporterstellung für Dienstnehmer als auch für die Ursachenlokalisierung im Fehlerfall, ist das Nachvollziehen von durchgeführten Aktionen. Hierzu sollte jede an der Abrechnung beteiligte Komponente ein Protokoll über die durchgeführten Aktionen führen.

IMP 7 *Hohe Güte der Komponenteninteraktionen*
Der Diensterbringer ist an einer in allen Belangen optimalen Ressourcennutzung bei der Bereitstellung der Abrechnungslösung interessiert. Somit sollten insbesondere die Interaktionen zwischen den Komponenten eine hohe Güte vorweisen. Diese äußert sich beispielsweise in einer geringen in Anspruch genommenen Bandbreite, einer robusten fehlertoleranten Kommunikation, einer Unterstützung unterschiedlicher Mechanismen zum Datenaustausch (Push/Pull/Event–basiert), etc.

IMP 8 *Ausfallsicherheit*
Da der unternehmerische Erfolg des Diensterbringers von der erfolgreichen und korrekten Abrechnung abhängt, muss sicher gestellt werden, dass die hierbei eingesetzten Komponenten so ausfallsicher wie möglich sind, um das Risiko zu senken.

IMP 9 *Erweiterbarkeit*
Die Anforderungen, insbesondere bezogen auf den Umfang der bereitgestellten Funktionalität der Abrechnungslösung, können sich durch Innovationen im Bereich der Dienste ändern. Um dennoch mit derartigen Änderungen Schritt halten zu können, sollte eine Komponente bezüglich ihrer Funktionalität erweiterbar sein.

2.3.4 Anforderungen an das Abrechnungsmanagement

In diesem Abschnitt werden Anforderungen an das Abrechnungsmanagement aus Sicht des Managers formuliert. Im Gegensatz zum vorhergehenden Abschnitt, in dem Anforderungen an die Schnittstelle zwischen Abrechnungsmanagement und den am Abrechnungsvorgang beteiligten Komponenten identifiziert wurden, liegt der Fokus dieses Abschnitts auf der Schnittstelle zwischen Manager und der ihn unterstützenden Managementanwendung:

MGT 1 *Integrative, einheitliche Sicht auf den Abrechnungsvorgang*
Um die Komplexität der Managementaufgabe auf ein beherrschbares Maß zu reduzieren, ist eine integrative, einheitliche Sicht auf das zu managende Ziel (in diesem Fall auf den Abrechnungsvorgang) unverzichtbare Voraussetzung.

MGT 2 *Flexibilität*
Die Spezifikation von Managementanweisungen sollte einfach und flexibel sein. Das heißt, dass es ohne weitere Schwierigkeiten möglich sein muss, bestehende Managementanweisungen zu ändern bzw. neue zu erstellen ohne die Notwendigkeit, beispielsweise den Betrieb des Dienstes, des Managementsystems, etc. anzuhalten.

MGT 3 *Automatisierung von wiederkehrenden Managementaktivitäten*
Um den Manager von immer wiederkehrenden Managementaktivitäten zu entlasten, ist eine geeignete Automatisierung dieser Aktivitäten gefordert. Zudem wird durch eine geeignete Automatisierung insbesondere auch die Fehleranfälligkeit gesenkt.

MGT 4 *Breite Unterstützung von Change–Managementaktivitäten*
Wie in Abschnitt 2.2.2 in Punkt 5 festgestellt wurde, ist insbesondere in Großkunden-

 Als Buch erhältlich im Verlag Dr. Hut, München, www.dr.hut-verlag.de (ISBN 3-934767-97-4)

szenarios aufgrund der langen Vertragslaufzeiten mit einer hohen Änderungsdynamik zu rechnen. Bisher werden die von beiden Seiten initiierten und die Abrechnung betreffenden Änderungen und deren Auswirkungen nur mangelhaft von Managementwerkzeugen unterstützt. Hierbei muss von Managementseite vor allem die Konsistenz der verteilten Datenhaltung gewahrt werden.

MGT 5 *Unterstützung heterogen zusammengestellter Abrechnungslösungen*
Bereits im vorhergehenden Punkt wird die Notwendigkeit einer vereinheitlichten Sicht auf heterogene Managementschnittstellen begründet. Um dies auch tatsächlich zu erreichen, ist die Unterstützung von heterogenen Managementschnittstellen (im Sinne von: den Zugriff auf die Schnittstelle implementieren) erforderlich.

MGT 6 *Unterstützung eines domänenübergreifenden Managements*
Neben einer vereinheitlichten Sicht auf heterogene Managementschnittstellen ist auch ein domänenübergreifendes Abrechnungsmanagement gefordert. Da insbesondere im Abrechnungsmanagement Komponenten gesteuert und überwacht werden müssen, die sich nicht in der dienstleistereigenen Domäne befinden (z.B. Messkomponenten auf Kundenseite), muss dies durch die Managementlösung geeignet unterstützt werden.

MGT 7 *Erweiterbarkeit*
Um auf sich neu ergebende Gegebenheiten reagieren zu können, sollte die Managementanwendung ohne weitere Schwierigkeiten erweiterbar sein.

MGT 8 *Skalierbarkeit*
Die Managementlösung sollte skalierbar hinsichtlich der Anzahl der zu steuernden und überwachenden Komponenten sein.

MGT 9 *Ausfallsicherheit*
Es liegt auf der Hand, dass die Managementlösung ausfallsicher gestaltet werden muss, um den unternehmerischen Erfolg des Dienstleisters sicher zu stellen.

MGT 10 *Robustheit*
Die Managementlösung sollte robust bzgl. auftretender Fehler sein.

MGT 11 *Mandantenfähigkeit*
Eine Abrechnungsmanagementlösung muss wie die abrechnungsrealisierenden Komponenten mandantenfähig sein, so dass diese ohne Einschränkung für mehrere Kunden parallel einsetzbar ist.

MGT 12 *Nachvollziehbarkeit*
Im Fehlerfall sollten die durchgeführten Managementaktionen nachvollzogen werden können, um die Ursache für das Auftreten eines Fehlers zu lokalisieren.

MGT 13 *Offenheit*
Da die Abrechnungsmanagementlösung unter Umständen mit Managementwerkzeugen aus anderen Bereichen (z.B. Leistungs–/Fehlermanagement) als auch aus anderen

Domänen (z.B. Sub–Dienstleisterdomäne) zusammenarbeiten muss, sind offene Schnittstellen eine unverzichtbare Voraussetzung hierfür.

2.3.5 Anforderungskatalog

Die in den vorhergehenden Abschnitten identifizierten und erklärten Anforderungen werden im Folgenden in Form einer übersichtlichen Tabelle zu einem Anforderungskatalog zusammengefaßt. Dieser kann dann zum Einen als Instrument zur Bewertung existierender Ansätze und Lösungen (siehe Kapitel 3) sowie zum Anderen zur Entwicklung einer neuen Lösung (siehe Kapitel 5) eingesetzt werden.

Anforderungskatalog	
Allgemeine Anforderungen	
ALL 1	Unterstützung des vollständigen Abrechnungsvorgangs
ALL 2	Klare Trennung des Abrechnungsvorgangs vom Abrechnungsmanagement
Anforderungen an den Abrechnungsvorgang	
Kostenermittlung	
KOS 1	Präzision
Tarifierung	
TRF 1	Nutzungsorientierung
TRF 2	Dienstgüteorientierung
TRF 3	Kundenindividuelle Tarifvereinbarung
TRF 4	Dienstorientierte Abrechnungseinheiten
TRF 5	Einfache, nachvollziehbare Tarifbestimmung
TRF 6	Vorhersehbarkeit
TRF 7	Kostendeckend
TRF 8	Durchsetzbarkeit
Nutzungserfassung	
UAC 1	Transparenz
UAC 2	Unterstützung einer hohen Messperformanz
UAC 3	Flexibilität bzgl. des Messorts
UAC 4	Unterstützung unterschiedlicher Messeinheiten
Gebührenberechnung	
CHA 1	Unterstützung von On–Demand–Charging
CHA 2	Flexible, skalierbare Gebührenberechnung

 Als Buch erhältlich im Verlag Dr. Hut, München, www.dr.hut-verlag.de (ISBN 3-934767-97-4)

Anforderungskatalog (*Fortsetzung*)	
Rechnungsstellung	
BIL 1	Unterstützung von Hot–Billing
BIL 2	Kundenorientierte Gestaltung des Inhalts
BIL 3	Unterstützung mehrerer Zustellungsarten
Inkasso	
INK 1	Unterstützung unterschiedlicher Zahlungsarten
INK 2	Unterstützung von Eskalationsmechanismen
Anforderungen an die abrechnungsrealisierenden Komponenten	
IMP 1	Standardisierte Nutzungsschnittstelle
IMP 2	Standardisierte Managementschnittstelle
IMP 3	Domänenübergreifende Kooperation von unterschiedlichen Komponenten
IMP 4	Sichere Interaktion zwischen beteiligten Komponenten
IMP 5	Mandantenfähigkeit
IMP 6	Verifizierbarkeit
IMP 7	Hohe Güte der Komponenteninteraktionen
IMP 8	Ausfallsicherheit
IMP 9	Erweiterbarkeit
Anforderungen an das Abrechnungsmanagement	
MGT 1	Integrative, einheitliche Sicht auf den Abrechnungsvorgang
MGT 2	Flexibilität
MGT 3	Automatisierung von wiederkehrenden Managementaktivitäten
MGT 4	Breite Unterstützung von Change–Managementaktivitäten
MGT 5	Unterstützung heterogen zusammengestellter Abrechnungslösungen
MGT 6	Unterstützung eines domänenübergreifenden Managements
MGT 7	Erweiterbarkeit
MGT 8	Skalierbarkeit
MGT 9	Ausfallsicherheit
MGT 10	Robustheit
MGT 11	Mandantenfähigkeit
MGT 12	Nachvollziehbarkeit
MGT 13	Offenheit

Tabelle 2.1: Anforderungskatalog

2.4 Zusammenfassung

In diesem Kapitel wurden zunächst die verwendete Terminologie und Grundmodelle im Bereich des Dienstmanagements und des dienst– und kundenorientierten Abrechnungsmanagements eingeführt und erläutert. Kern der Ausführungen war der Dienstlebenszyklus, das MNM Dienstmodell sowie die Teilfunktionen des Abrechnungsmanagements.

Anschließend wurde anhand einer Szenariobeschreibung im Großkundenbereich, die auf den gesammelten Erfahrungen im gemeinsam mit der DeTeSystem durchgeführten Forschungsprojekt „Abrechnungsmanagement“ basiert, die herausragenden Charakteristika analysiert und vorgestellt.

Darauf aufbauend wurde ein Anforderungskatalog entwickelt, der in den noch folgenden Kapiteln sowohl als Instrument zur Bewertung von existierenden Ansätzen und Lösungen als auch zur Entwicklung einer neuen Managementlösung dienen soll.

 Als Buch erhältlich im Verlag Dr. Hut, München, www.dr.hut-verlag.de (ISBN 3-934767-97-4)

Kapitel 3

Abrechnungsmanagement: Status Quo

Dieses Kapitel bietet einen Überblick über den derzeitigen Status Quo im Abrechnungsmanagement. Es werden Arbeiten von Standardisierungs– und Industriegremien, Forschungsansätze und existierende Produkte begutachtet. Hierbei werden die einzelnen Arbeiten anhand der im vorigen Kapitel identifizierten Teilfunktionen des Abrechnungsmanagements eingeordnet und bewertet. Zum Schluss erfolgt durch Anwendung des in Kapitel 2 entwickelten Anforderungskatalogs eine Gesamtbewertung der bisher erzielten Ergebnisse.

3.1 Einführung

Zunächst werden in Abschnitt 3.2 die Spezifikationen von Standardisierungs– als auch Industriegremien analysiert. Jedes Gremium wird kurz mit den verfolgten Zielen vorgestellt, um anschließend die im Rahmen des Gremiums für das Abrechnungsmanagement entwickelten Arbeiten zu evaluieren.

In Abschnitt 3.3 werden Forschungsarbeiten untersucht. Hierbei werden Arbeiten im Bereich der Tarifierung, welche die Mehrzahl stellen, getrennt von Arbeiten im Bereich des eigentlichen Abrechnungsmanagements präsentiert und evaluiert.

Abschnitt 3.4 beschäftigt sich schließlich mit derzeitig für das Abrechungsmanagement erhältlichen Produkten.

Zum Schluss werden die Evaluationsergebnisse durch Anwendung des in Kapitel 2 entwickelten Anforderungskatalogs übersichtlich in Form von Tabellen zusammengefasst und in Bezug auf die Verwertbarkeit des noch zu entwickelten Lösungsansatzes bewertet.

3.2 Die Arbeiten der Industrie– und Standardisierungsgremien

In den nachfolgenden Abschnitten werden jeweils separat für jedes Standardisierungs– bzw. Industriegremium die für das Abrechnungsmanagement relevanten Spezifikationen vorgestellt und bewertet. Dies umfasst die Arbeiten von OSI/TMN, TINA–C, ITIL, TM Forum, IETF, IPDR Organization und OMG.

3.2.1 OSI–Management und TMN

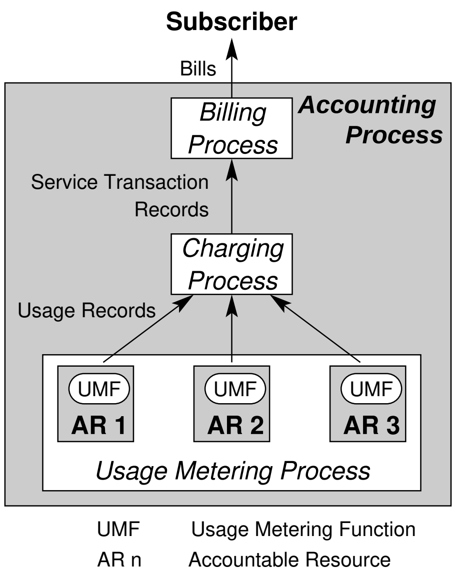

Abbildung 3.1: Die Architektur des OSI Accounting Managements

Das Ziel des von der ISO entwickelten *Open Systems Interconnection Basic Reference Models (OSI–RM)* [ISO 7498] ist eine Grundlage für die Spezifizierung von Standards für die Verbindung von offenen Systemen zu schaffen. In einer Reihe von zusätzlichen Dokumenten werden unterschiedliche Aspekte der Verbindung offener Systeme betrachtet. Einer dieser Aspekte ist das Management offener Systeme. In [ISO 7498-4] wird die Architektur des OSI–Managements gemäß der vier *Teilmodelle* Kommunikation, Funktion, Organisation und Information festgelegt. Das Funktionsmodell ist wiederum in fünf Funktionsbereiche, den sog. FCAPS (Fault, Configuration, Accounting, Performance, Security) eingeteilt. Innerhalb der einzelnen Funktionsbereiche wird die Ableitung generischer Managementfunktionen (den sog. *System Management Functions*) angestrebt, die in weiteren, selbstständigen Dokumenten [ISO 10164-x] veröffentlicht werden. Das *Telecommunications Management Network (TMN)* [ITU M.3000, Sido 98] ist ein Referenzmodell der ITU–T für das Management von Telekommunikationsnetzen und basiert in wesentlichen Teilen auf den Konzepten des OSI–Managements, so dass die relevanten TMN–Spezifikationen im Rahmen dieses Abschnitts behandelt werden.

Überblick

Im Rahmen des ISO-Standards [ISO 10164-10] wird das Abrechnungsmanagement in drei Teilprozesse unterteilt (vgl. Abbildung 3.1): *Usage Metering Process* (entspricht der Nutzungserfassung), *Charging Process* (entspricht der Gebührenberechnung) und *Billing Process* (entspricht der Rechnungsstellung). Lediglich die Nutzungserfassung ist im Rahmen von [ISO 10164-10]

 Als Buch erhältlich im Verlag Dr. Hut, München, www.dr.hut-verlag.de (ISBN 3-934767-97-4)

näher untersucht worden, die beiden übrigen Prozesse werden nur genannt und nicht weiter betrachtet.

Die in [ISO 10164-10] spezifizierten Managementobjekte und Interaktionen legen den grundlegenden Ablauf der Nutzungserfassung fest. Hierbei wurde auf eine ausreichend flexible Steuerung der Objekte und damit des Prozesses geachtet, um den Anforderungen aus unterschiedlichen Szenarios gerecht zu werden. Dies äußert sich beispielsweise darin, dass die Einsammlung der gemessenen Daten in unterschiedlicher Weise vollzogen werden kann (push, pull, eventbasiert). Folgende Managementobjekte sind innerhalb von [ISO 10164-10] spezifiziert worden:

- Ein *Accountable Object* repräsentiert eine sog. *Accountable Resource*, d.h. eine Ressource, die im Rahmen des durchzuführenden Messvorgangs überwacht werden soll.
- Das *Usage Metering Control Object* steuert den eigentlichen Messvorgang und bietet damit Methoden zum Starten, Anhalten, etc. der Datenerfassung an.
- Das *Usage Metering Data Object* kann als Oberklasse für Nutzungsdaten verstanden werden und ist einem Accountable Object zugeordnet. Da die Messdaten von der zu überwachenden Ressource abhängig sind und diese nicht allgemein angebbar sind, wurde keine detailliertere Spezifikation dieses Objekts vorgenommen.
- Ein *Accounting Event* löst den Messvorgang sowie die Aufzeichnung der Messdaten auf Basis eines Usage Metering Data Objects aus.
- Nach Beendigung des Mess– und Aufzeichnungsvorgangs werden die Messdaten in einem *Usage Metering Record* (Unterklasse eines *Log Object* [ISO 10164-6]) gespeichert.
- Die *Usage Metering Records* werden zur Weiterverarbeitung an den *Charging Process* weitergeleitet.

Damit enthält der ISO–Standard [ISO 10164-10] lediglich Managementobjekte für die Administration von gemessenen Daten und läßt insbesondere die beiden genannten Teilprozesse Charging und Billing unspezifiziert. Die weiteren, in 2.1.2 genannten Teilfunktionen des Abrechnungsmanagement bleiben gänzlich unbetrachtet.

An dieser Stelle setzt die ITU-T mit ihrem Rahmenwerk für das Telekommunikationsmanagement *Telecommunications Management Network (TMN)* [ITU M.3010] an, das, wie bereits erwähnt, auf den Konzepten und den Spezifikationen des OSI–Managements basiert. In [ITU M.3400] werden gemäß den FCAPS sog. *Function Set Groups* spezifiziert, die eine Erweiterung der OSI System Management Functions vorsehen. Innerhalb des Dokuments [ITU M.3400] wird das Abrechnungsmanagement in die folgenden vier Function Set Groups eingeteilt:

- *Usage Measurement*: Dies entspricht der Nutzungserfassung.
- *Tariffing/Pricing*: Dies entspricht der Tarifierung.

- *Collections and Finance*: Dies umfasst neben der Rechnungsstellung auch betriebswirtschaftlich administrative Aspekte, wie z.B. Personalmanagement, Rentenverwaltung, Steuererklärung, etc.
- *Enterprise Control*: Diese Menge umfasst betriebswirtschaftliche, fiskale Aspekte des Abrechungsmanagement, wie z.B. Kostenreduzierungsmethoden, Versicherungen, etc.

Jede dieser Function Set Groups enthält eine Liste von unterschiedlichen *Function Sets*, welche eine Gruppierung von Methoden zu einem gemeinsamen Oberbegriff vorsehen. Bei den meisten Function Sets bleibt es jedoch bei der bloßen Nennung und einer kurzen Erläuterung, welche Art von Methoden in die Menge mitaufgenommen werden sollten. Nur für vereinzelte Function Sets werden Methoden auch tatsächlich angegeben. Hierbei beschränkt man sich allerdings ebenfalls auf die bloße Nennung von Methodennamen mit einer meist einzeiligen Erklärung, ohne jegliche Parameter, Datentypen und –objekte festzulegen. Somit wird zwar in [ITU M.3400] das Abrechnungsmanagement umfassender als sonst üblich aufgefasst, jedoch wird weder eine detaillierte Analyse noch Spezifikation der Prozesse bzw. Funktionen durchgeführt.

Bewertung

Mit [ISO 10164-10] wird von OSI die Nutzungserfassung mittels der Definition von generischen Managementobjekten recht detailliert spezifiziert. Alle übrigen, im Abrechnungsmanagement relevanten Teilfunktionen (siehe Abschnitt 2.1.2) bleiben jedoch gänzlich unbetrachtet. Die ITU–T fasst das Abrechnungsmanagement zwar insgesamt wesentlich breiter, jedoch bleibt [ITU M.3400] sehr oberflächlich. Dennoch können die in [ITU M.3400] bzgl. des Abrechnungsmanagements betrachteten Aspekte als Anforderungsquelle verwendet werden.

3.2.2 TINA Consortium (TINA–C)

Das *Telecommunications Information Networking Architecture Consortium (TINA–C)*[1] war[2] ein Industrie– und Standardisierungsgremium im Telekommunikationsbereich, welches in zahlreichen, verabschiedeten Dokumenten Weiterentwicklungen für Telekommunikationsunternehmen im Bereich der interorganisationellen Dienstbereitstellung bot. Insbesondere sollte durch die Spezifikation einer verteilten Dienstarchitektur (siehe *Service Architecture* [TINA 97]) und Dienstmanagementarchitektur (siehe *Management Architecture* [TINA 94]) die verteilte Realisierung von neuartigen Diensten, die u.U. in Kooperation mit mehreren Providern stattfindet, erheblich erleichtert werden. Die Bestrebungen von TINA–C gründeten sich auf die Umorientierung der Telekommunikationsunternehmen von reinen Anbietern eines einfachen Telefoniedienstes auf Basis einer Telekommunikationsinfrastruktur hin zu Anbietern von teilweise

[1] `http://www.tinac.com/`
[2] Die Arbeit von TINA–C wurde Dezember 2000 offiziell für beendet erklärt.

 Als Buch erhältlich im Verlag Dr. Hut, München, www.dr.hut-verlag.de (ISBN 3-934767-97-4)

komplexen Mehrwertdiensten, die auch Dienste in Datennetzen miteinbeziehen. Da insbesondere einheitliche, standardisierte Nutzungs– und Managementschnittstellen an den Providergrenzen Voraussetzung für die kooperative, verteilte Realisierung von Mehrwertdiensten ist, hat sich TINA v.a. dieser Aufgabe gewidmet. TINA stützt sich hierbei auf den Spezifikationen von OSI–TMN auf, deren Arbeiten bereits im vorhergehendem Abschnitt 3.2.1 genannt worden sind.

Überblick

Im Bereich des Abrechnungsmanagements wurde mit [TINA 96] ein „Initial Draft“ für eine *Accounting Management Architecture* eingereicht. Hierbei wird zusätzlich zur bereits in [TINA 97] spezifizierten *Service Session*, welche die Dienstnutzung v.a. in einen zeitlichen Kontext mit einem Beginn und einem Ende der Nutzung setzt, die sog. *Service Transaction* spezifiziert, welche einen wesentlich allgemeineren, nicht nur ausschließlich zeitlich orientierten Dienstnutzungsbegriff einführt. An eine Service Transaction wird ein sog. *Accounting Management Context* gebunden, der alle abrechnungsrelevanten Informationen (die sog. 5 W–Fragen: Wer, Wann, Wo, Was, Wie) beinhaltet und aus den folgenden Bestandteilen aufgebaut ist:

- *Session Component Declaration*: Dies beinhaltet die einzelnen, möglichen Zustände und deren Übergänge, die mit einer Dienstnutzung zusammenhängen und an denen abrechnungsrelevante Ereignisse geknüpft werden können (siehe nächsten Punkt). Beispiel: Ein typischer Telefonanruf besteht aus dem Wählen, Klingeln, Reden, Besetztton, etc.

- *Event Management Declaration*: Hiermit werden Ereignisse definiert, die mit einem Zustand (Session Component) bzw. mit einem Übergang verbunden werden und welche für die Abrechnung relevant sind. Beispielsweise kann definiert werden, dass bei einem Zustandsübergang der Nutzer per Ereignisgenerierung informiert wird, da damit Gebühren einhergehen. Es müssen mindestens soviele Ereignisse deklariert werden, dass eine erfolgreiche Abrechnung der Dienstnutzung und damit Beantwortung der 5 W–Fragen möglich ist.

- *Tariff Structure Description*: Diese Datenstruktur enthält ein Tarifmodell, auf Basis dessen unterschiedliche Abrechnungsarten (pro Session, Dienstnutzung, etc.) unterstützt werden können, so dass der Nutzer u.U. eine Abrechnungsart auswählen kann.

- *Billing and Recovery Configuration*: Hiermit wird die Rechnungsstellung, bspw. bzgl. Rechnungsempfänger, Aufteilung der Rechnung auf unterschiedliche Teilnehmer, etc. als auch die sog. Recovery–Funktion konfiguriert, welche Aktivitäten vorsieht, falls eine Service Transaction nicht erfolgreich abgeschlossen werden konnte.

Für die eben vorgestellten Komponenten eines Accounting Management Contexts werden jeweils einige wenige generische Schnittstellen und Datenstrukturen spezifiziert. Hierbei stützt man sich erheblich auf die im vorhergehenden Abschnitt genannte OSI–Management Spezifikation [ISO 10164-10] ab, so dass nur wenig neue Datenstrukturen resp. Schnittstellenmethoden eingeführt werden. Zum Teil sieht das Dokument zudem eine sehr restriktive Spezifikation vor, wie beispielsweise, dass die *Billing Configuration*–Schnittstelle lediglich vier verschiedene Abrechnungsarten unterstützt.

Bewertung

Neben einigen Ungenauigkeiten bei der Begriffsdefinition, hinterläßt das Dokument [TINA 96] insgesamt einen unterspezifizierten Eindruck. Die Hauptleistung liegt in der anhand von Anwendungsfällen beispielhaft analysierten Anforderungen an eine Abrechnungsmanagementarchitektur für eine TINA–basierte Telekommunikationsinfrastruktur. Der Hauptnachteil der Spezifikation liegt in der Einführung neuer, dienstbezogener Datenstrukturen (Service Transaction, Accounting Management Context) für die Abrechnung *außerhalb* der mittels der TINA Service Architecture durchgeführten Spezifikationen. Damit wird für eine erfolgreiche Abrechnung vorausgesetzt, dass die Erzeugung von Abrechnungsdaten mittels der vorgestellten, neu definierten Datenstrukturen während der Dienstnutzung stattfindet. Dieser Sachverhalt trifft allerdings nicht auf bereits existierende, TINA–konforme Dienst– und Dienstmanagementplattformen zu. Es wird somit insbesondere nicht betrachtet, wie aus bereits anfallenden Daten abrechnungsrelevante Informationen abgeleitet werden können, um auf diese Weise einen Migrationspfad zu der neuen Abrechnungsmanagementarchitektur zu eröffnen. Desweiteren wird auch nicht die Integration von existierenden Abrechnungskomponenten in die spezifizierte Abrechnungsmanagementarchitektur adressiert, wie z.B. Billing Systeme, welche die spezifizierten Datenstrukturen und Methoden nicht unterstützen. Da das Dokument [TINA 96] aus den genannten Gründen z.T. heftigen Diskussionen ausgesetzt war, hat es den „Initial Draft"–Status nie abgelegt und wurde mittlerweile zurückgezogen.

3.2.3 IT Infrastructure Library (ITIL)

Bei der *IT Infrastructure Library (ITIL)*[3] handelte es sich ursprünglich um eine lose Sammlung von Dokumenten, die Empfehlungen des britischen *Office of Government Commerce (OGC)* für den Betrieb von IT–Diensten enthielten. Die zunächst nur für Einrichtungen der britischen Regierung gedachten Empfehlungen wurden mittlerweile zu öffentlich zugänglichen Dokumenten und entwickelten sich, nach eigenen Aussagen, zum de facto Standard im Bereich des IT Service Managements. Bis vor kurzem fehlte jedoch ein gemeinsames Modell [HAN 99], in dem sich die einzelnen Dokumente hätten einordnen lassen, so dass eine gemeinsame Sichtweise auf Zusammenhänge und Gemeinsamkeiten bisher fehlte. Seit Beginn des Jahres 2000 erfolgt jedoch bezüglich der unter dem ITIL–Label veröffentlichten Dokumente eine Restrukturierungsmaßnahme, die neben einer inhaltlichen Überarbeitung aller Dokumente auch eine Einteilung resp. Zusammenfassung der Spezifikationen zu insgesamt sechs Bändern umfasst, von denen bisher zwei erschienen sind: „Service Delivery" [ITIL 00] und „Service Support" [ITIL 01].

[3] `http://www.itil.co.uk/`

 Als Buch erhältlich im Verlag Dr. Hut, München, www.dr.hut-verlag.de (ISBN 3-934767-97-4)

Bei den Veröffentlichungen handelt es sich um sog. „Best Practice“–Dokumente, die Empfehlungen zur Planung und Realisierung eines IT Service Managements in Unternehmen auf Basis von gesammelten Erfahrungen der ITIL–Autoren enthalten. Damit handelt es sich bei ITIL im Gegensatz zu den bisher in diesem Kapitel vorgestellten Spezifikationen von OSI–TMN (siehe Abschnitt 3.2.1) und TINA–C (siehe Abschnitt 3.2.2) um einen strikten Bottom–Up–Ansatz. Die Verfasser sind Spezialisten im Bereich des IT Service Managements, die in unterschiedlichen Unternehmen beratend oder aktiv am Betrieb eines Dienstmanagements partizipieren. Anspruch der ITIL ist, dass die Empfehlungen auf alle möglichen IT–Dienste und Unternehmensstrukturen (von mittelständisch bis Großunternehmen) anwendbar sind. Bei der Beschreibung der Empfehlungen wird eine *prozessorientierte Sicht* angewendet, d.h. es werden Aktivitäten beschrieben, welche von Organisationseinheiten, Rollen, etc. mittels des Einsatzes von Ressourcen ausgeführt werden. Es sind allerdings, abhängig vom Autor eines Abschnitts, starke Unterschiede im Abstraktions– und Detailniveau der Beschreibungen der einzelnen Prozesse feststellbar: meist werden die typisch notwendigen Prozessaktivitäten anhand von Beispielen in Prosa erläutert; nur in seltenen Fällen werden formalere, abstraktere Beschreibungsmöglichkeiten, wie die Anwendung von Prozessgraphen, eingesetzt. Implementierungsaspekte und Realisierungsdetails werden grundsätzlich nicht betrachtet, da dies als szenarioabhängig gewertet wird und damit die Allgemeinheit der Empfehlungen verloren ginge.

Überblick

Mit dem Thema Abrechnungsmanagement befasst sich innerhalb von [ITIL 01] der Managementprozess „Financial Management for IT services“, der in die folgenden drei Teilprozesse aufgeteilt wird:

- *Budgeting*: Dieser Prozess schätzt und überwacht die Ausgaben für die Bereitstellung von IT–Diensten innerhalb eines Unternehmens.
- *IT Accounting*: Dieser Prozess beschäftigt sich mit der Analyse von Kosten, die durch den Betrieb von Diensten entstehen.
- *Charging*: Dieser Prozess beinhaltet alle Aktivitäten, die üblicherweise zum „klassischen“ technischen Abrechnungsmanagement gezählt werden, demnach insbesondere die Tarifierung, Gebührenberechnung und Rechnungsstellung.

Der Schwerpunkt der Betrachtung liegt auf den betriebswirtschaftlichen Aspekten des Abrechnungsmanagements, die von der Rolle des *IT Finance Managers* ausgeführt werden, so dass insbesondere die Teilprozesse „Budgeting“, „IT Accounting“ sowie die Tarifierung innerhalb des „Charging“-Teilprozesses beschrieben werden. Recht detailliert wird die Kostenanalyse erläutert, welche die Anwendung eines *Kostenmodells* vorsieht, das alle für das Finance Management relevanten, kostenverursachenden Elemente als Posten enthält. Damit einhergehend werden Entscheidungskriterien beschrieben, welche Posten in eine Kostenaufstellung auf welche Weise mitaufzunehmen sind.

Auch für die Tarifierung werden Entscheidungskriterien angegeben, wie ein geeigneter Tarif, beispielsweise in Abhängigkeit vom Kunden (intern/extern) und den Auswirkungen auf das Nutzungsverhalten, für eine Dienstleistung zusammenzustellen ist. In Zusammenhang damit werden auch die dafür notwendigen Bestandteile eines SLAs beschrieben. Die technische Realisierung eines Abrechnungsmanagements durch Einsatz von Werkzeugen wird allerdings dem ITIL-Credo folgend vollständig nicht betrachtet.

Bewertung

Wie bereits erwähnt, liegt der Schwerpunkt bzgl. des Abrechnungsmanagements auf den betriebswirtschaftlichen Aspekten. Hierbei werden Prozessschritte allerdings recht ausführlich, wenn auch nicht in formaler Art und Weise, an Beispielen erläutert. Damit bietet ITIL eine neue Sichtweise auf das Abrechnungsmanagement, die zumindest als Anforderungsquelle für die technische Realisierung dienen kann, auch wenn Implementierungsaspekte und damit die Umsetzung und Realisierung eines Abrechnungsmanagements per se nicht untersucht wird. Insbesondere die Beschreibung der Kostenanalyse als auch der Tarifierung und damit zusammenhängend die Erläuterung der abrechnungsrelevanten Bestandteile eines SLAs bieten wertvolle Hinweise in diesem Bereich. Nichtsdestotrotz, aufgrund der fehlenden einheitlichen und v.a. formalisierten Prozessbetrachtung fällt die Anwendung resp. Umsetzung der Empfehlungen schwer, sofern nicht bereits detaillierte Kenntnisse der Materie vorhanden sind (siehe hierzu auch die Anmerkungen in [BRS 02, GHK+ 01]).

3.2.4 TeleManagement Forum (TM Forum)

Das *TeleManagement Forum (TM Forum)*[4], ehemals *Network Management Forum (NMF)*, ist ein non–profit Industriekonsortium, das sich vorrangig mit dem Betrieb und Management von Diensten im Telekommunikationsbereich beschäftigt. Ursprüngliches Ziel war eine industrienahe Verabschiedung von Spezifikationen, die es ermöglichen, sog. *Operations Support Systems (OSS)* über Telekommunikationsanbieter–Grenzen hinweg miteinander zu koppeln. Bei OSS handelte es sich meist um proprietäre, mainframe–basierte, stand–alone Systeme der Telko–Unternehmen, die zum Management der Kommunikationsinfrastruktur, zur Inventarverwaltung, Rechnungsstellung, etc. eingesetzt wurden. Durch zahlreiche Unternehmensfusionen und –kooperationen wurde die Kopplung von historisch gewachsenen OSS zu einem ernstzunehmenden Problem. Durch die bereits erwähnte Wandlung der Telko–Unternehmen von reinen Kommunikationsdienste–Anbietern zu (IT–)Diensteanbietern werden mittlerweile sog. *Next Generation OSS (NGOSS)* propagiert, die eine wesentlich dynamischere Einführung von Diensten und des dafür notwendigen Managements unterstützen sollen. An diesem Punkt setzt nun das TM Forum an, das die Konzeption, Entwicklung und Einführung von NGOSS durch entsprechende Spezifikationen unterstützen will.

[4] `http://www.tmforum.org/`

 Als Buch erhältlich im Verlag Dr. Hut, München, www.dr.hut-verlag.de (ISBN 3-934767-97-4)

Ähnlich zum Ansatz von ITIL (siehe Abschnitt 3.2.3) für den Betrieb und das Management von IT–Diensten, wird in der *Guidebook*–Reihe des TM Forums eine *prozess–* und *kundenorientierte Sicht* auf diese Aufgabe angewandt. Zentrale Rolle spielen hierbei die *Telecom Operations Map (TOM)* [GB 910] und die *enhanced Telecom Operations Map (eTOM)* [GB 921], welche die Beschreibung aller für einen Service Provider notwendigen Prozesse enthalten. Im Gegensatz zu ITIL wird allerdings ein strukturierter *Top–Down–Ansatz* zur Prozessbeschreibung gewählt. Das heißt, dass ein „globales" Prozessmodell existiert, in dem die identifizierten Teilprozesse in Beziehung zueinander gesetzt werden. Zudem werden innerhalb der TOM in einem Verfeinerungsschritt Prozessgraphen zur Darstellung von Teilprozessen verwendet. Allerdings erreicht TOM/eTOM bei weitem nicht die Detailtiefe von ITIL.

Überblick

In Abbildung 3.2 ist das TOM–*Business Process Framework* dargestellt, welches die anbieter–internen Managementprozesse veranschaulicht und in welchem die abrechnungsrelevanten Prozesse grau hinterlegt sind. Wie der Abbildung entnommen werden kann, sind die Prozesse in die folgenden vier Schichten eingeteilt:

- *Customer Interface Management Processes*
  Diese Schicht enthält diejenigen Prozesse, welche die direkte Schnittstelle zwischen Customer und Provider realisieren. Diese Prozesse sind bisher vom TM Forum nicht weiter spezifiziert worden.
- *Customer Care Processes*
  Diejenigen Prozesse, die jeweils *kundenabhängig* sind, sind in dieser Schicht wiederzufinden. Das heißt, dass die von den Prozessen angestoßenen Aktionen jeweils immer genau einen Kunden betreffen und zum Teil auch kundenindividuell realisiert werden müssen.
- *Service Development and Operations Processes*
  Dieser Schicht werden die *dienstabhängigen* Prozesse zugeordnet. Managementaktionen, die von diesen Prozessen angestoßen werden, betreffen i.d.R. eine Menge von Kunden.
- *Network and Systems Management Processes*
  Diese Schicht definiert Prozesse des Netz-und Systemmanagements. Das TM Forum setzt hierbei auf den Spezifikationen von OSI–TMN auf (siehe Abschnitt 3.2.1).

Neben dieser Schichtung wird vom TM Forum ebenfalls eine vertikale Einteilung der Prozesse vorgenommen, die ebenfalls in Abbildung 3.2 dargestellt ist. Die auf diese Weise vorgenommene Dreiteilung des Modells soll nach Ansicht des TM Forums die drei Hauptaufgaben des Providers aus Sicht des Kunden widerspiegeln: Vertragserfüllung (Service Fulfillment), Sicherstellung der Diensterbringung (Service Assurance), Rechnungsstellung (Billing). Diese vertikale Einteilung spiegelt laut TOM auch den Dienstlebenszyklus wider.

Die im Business Process Framework aufgeführten Prozesse werden innerhalb der TOM jeweils in eigenen Prozessgraphen dargestellt und beschrieben. Bei den Prozessgraphen handelt es sich aber nur um sog. Informationsflussdiagramme, die einen Prozess als Black–Box betrachten und

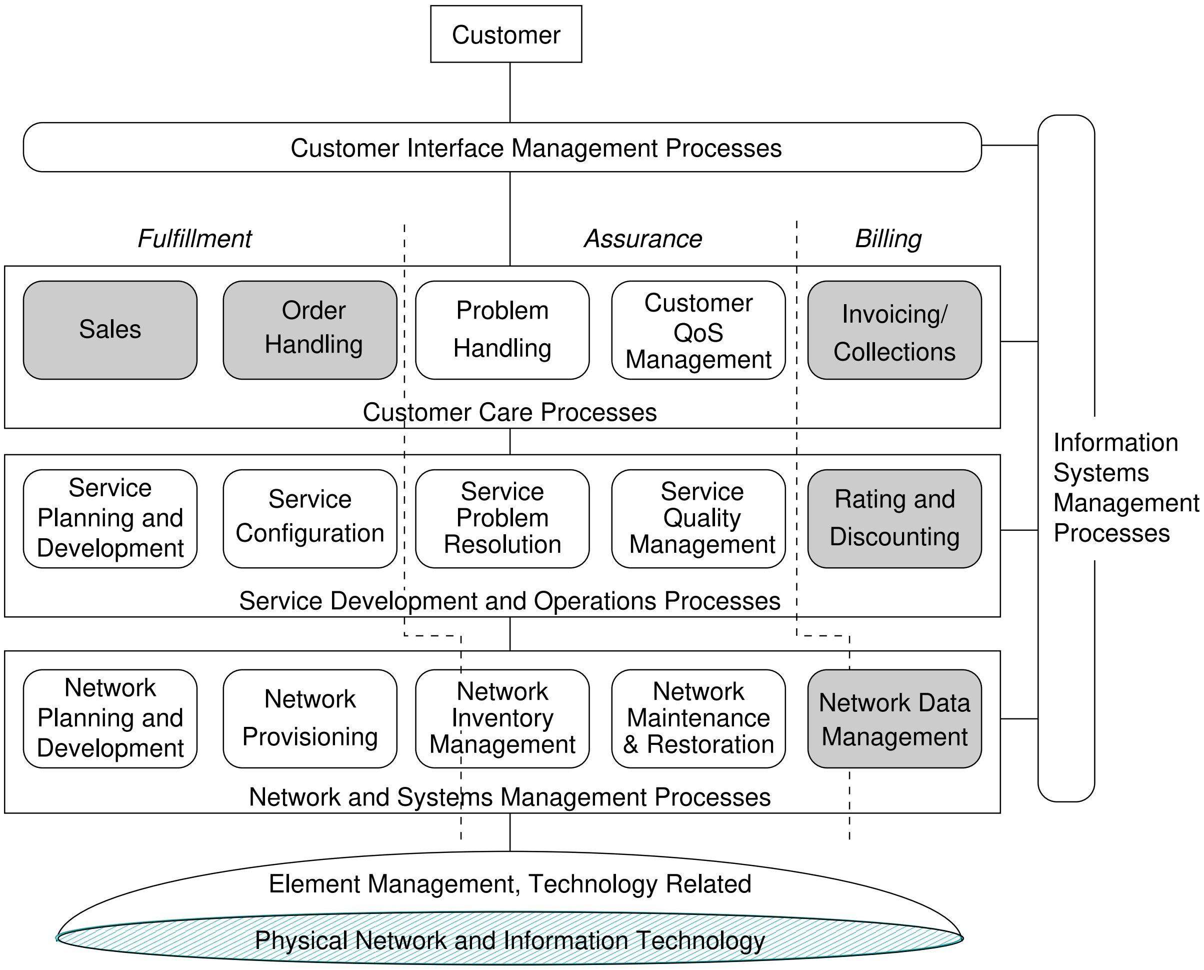

Abbildung 3.2: Das TOM–*Business Process Framework* aus [GB 910]

jeweils nur die Ein– und Ausgabedaten visualisieren. Somit bietet die TOM keine genaue Beschreibung des Ablaufs eines Prozesses hinsichtlich der dafür notwendigen Aktivitäten. Es wird lediglich der Informationsfluss zwischen den Prozessen durch Pfeile angedeutet, ohne jedoch die ausgetauschten Daten genauer zu spezifizieren. Die internen Bearbeitungsschritte eines Prozesses werden stichpunktartig und in informeller Art und Weise beschrieben und sollen lediglich als Anhaltspunkte für die tatsächliche Implementierung dienen. Um sich einen Eindruck über den Detailgrad der Prozessbeschreibung verschaffen zu können, ist in Abbildung 3.3 der abrechnungsrelevante *Invoice and Collection Process* dargestellt.

Die zu Beginn erwähnte eTOM [GB 921] erweitert die TOM, indem neue Schichten und Teilprozesse hinzugefügt werden. Die Erweiterung beinhalten einen „Strategy, Infrastructure and Product“–Block von Prozessen, der v.a. die Planung der als „Operations“ bezeichneten Prozesse

 Als Buch erhältlich im Verlag Dr. Hut, München, www.dr.hut-verlag.de (ISBN 3-934767-97-4)

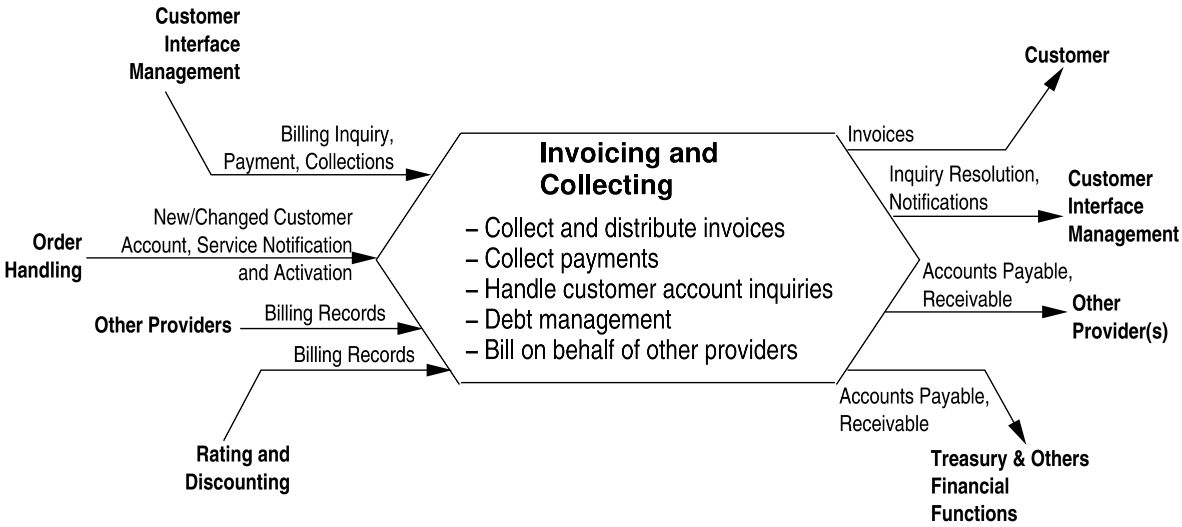

Abbildung 3.3: Der *Invoice and Collection Process* der TOM aus [GB 910]

der TOM vorsieht. Desweiteren wurde explizit eine „Supplier"–Schicht eingefügt, welche die Verbindung zu Sub–Dienstleistern ermöglicht. Zudem wurde ein „Enterprise Management"–Block von Prozessen hinzugefügt, welcher nicht–technische Managementprozesse beinhaltet. Im Gegensatz zur TOM führt die recht aktuelle Weiterentwicklung eTOM noch *keine* näheren Erläuterungen zu den neu hinzugefügten Prozessen in Form von Informationsflussgraphen auf. Diese werden lediglich mit einer kurzen Beschreibung genannt. Bezüglich des Abrechnungsmanagements sind somit kaum neue Erkenntnisse zu gewinnen.

Bewertung

Die Telecom Operations Map [GB 910] bietet ein wertvolles Prozessmodell im Bereich der Dienstbereitstellung. Die enthaltenen Informationsflussdiagramme bieten konkrete Anhaltspunkte für eine Implementierung und sind somit als Hilfestellung für Service Provider zu verstehen. Zusätzlich lassen sich aus den Diagrammen ebenfalls Anforderungen an die Prozesse ableiten.

Aufgrund des überblickartigen Charakters von [GB 910] wird ein Prozess als Black Box mit Ein- und Ausgabedaten betrachtet. Es werden keine Akteure, keine Ressourcen, keine Schnittstellen (im Sinne von Objekt*interfaces*) und keine Protokolle spezifiziert und adressiert. Desweiteren werden die Prozesse jeweils einzeln beschrieben, so dass ein Ende–zu–Ende–Prozessfluss hinsichtlich einer Aufgabe (wie z.B. dem Abrechnungsmanagement) selbst erarbeitet werden muss, um sich einen Überblick zu verschaffen. Zudem hat TOM/eTOM nicht die Detailtiefe wie ITIL. Dafür werden Prozesse geordnet und strukturiert dargestellt, so dass sich beide Ansätze gut ergänzen.

Insgesamt bietet TMForums TOM/eTOM gute Hinweise auf die notwendigen (Management–)

Aufgaben zur *Realisierung* eines kundenorientierten Abrechnungsvorgangs, auch wenn keine Implementierungsaspekte betrachtet werden. Allerdings werden keine Mechanismen zur *Steuerung* des Abrechnungsvorgangs behandelt, so dass eine wesentliche Aufgabe des Abrechnungsmanagements außer Acht gelassen wird.

3.2.5 Internet Engineering Task Force (IETF)

Die Aufgabe der Weiterentwicklung von Technologiespezifikationen für das Internet wird von der *Internet Engineering Task Force (IETF)*[5] übernommen. Hierbei wird das Aufgabenfeld in mehrere Bereiche (*areas*) unterteilt. Die unterschiedlichen Aspekte jedes Bereichs werden wiederum von einzelnen *Working Groups* bearbeitet, die sich mit einer speziellen Thematik und vorher definierten Zielen (z.B. Spezifikation eines Protokolls) innerhalb eines Bereichs beschäftigen. Ist das Ziel erreicht, so wird i.d.R. die Working Group aufgelöst. Die Arbeit jeder Working Group wird durch die Spezifizierung und Verabschiedung von Internet Standards dokumentiert. Die Working Groups veröffentlichen diesbezüglich in unterschiedlichen Zeitabständen *Internet Drafts*, welche Standardisierungsvorschläge zur Diskussion stellen und immer nur eine begrenzte Zeit gültig sind, und *Request for Comments (RFC)*, welche auf Basis von Internet Drafts und dem damit verbundenen Review–Prozess entwickelt worden sind und meist nach einer gewissen Zeit zu einem *Internet Standard* umgewandelt werden. Die speziell für das Abrechnungsmanagement relevanten Working Groups sind *Authentication, Authorization and Accounting (AAA)* und die mittlerweile aufgelöste *Realtime Traffic Flow Measurement Group (RTFM)*, deren wichtigste Arbeiten im Nachfolgenden beschrieben werden.

Überblick

Der Schwerpunkt der Untersuchungen der (mittlerweile aufgelösten) *Realtime Traffic Flow Measurement Group (RTFM)* lag in der Spezifikation einer *Architektur* sowie von *Protokollen für die Nutzungserfassung* im Internet, so dass insbesondere die erfassten Daten für die Netzplanung (Trendanalyse) und die Abrechnung verwendet werden können. Die *Authentication, Authorization and Accounting (AAA)* Working Group hingegen beschäftigt sich v.a. mit der Spezifikation von *Netzzugangsprotokollen*, wie z.B. Einwahlprotokollen, die u.a. den Anforderungen der Abrechnung genügen. Nachfolgend werden die Arbeiten der beiden Gruppen getrennt voneinander beschrieben.

Arbeiten im Kontext der Realtime Traffic Flow Measurement WG (RTFM)
In [RFC 1272] wird das Internet Accounting erstmals definiert und die Basisarchitektur festgelegt. Hierbei wird explizit sowohl die Terminologie als auch die Architektur des OSI Accounting Managements nach [ISO 10164-10] (siehe auch Abschnitt 3.2.1) übernommen.

[5] `http://www.ietf.org/`

 Als Buch erhältlich im Verlag Dr. Hut, München, www.dr.hut-verlag.de (ISBN 3-934767-97-4)

Wie bereits erwähnt, liegt der Fokus der Themen, die im Kontext von RTFM bearbeitet wurden, auf der Nutzungserfassung.

In [RFC 2722] wird nun auf Basis von [RFC 1272] eine Verfeinerung der Architektur hinsichtlich der Nutzungserfassung entworfen (siehe Abbildung 3.4). Der Fokus der Betrachtung liegt hierbei auf dem Messvorgang, d.h. es wird geklärt, was auf welcher Weise (im Internet) gemessen werden kann. Zur Steuerung des Messvorgangs sieht die Architektur nun explizit einen *Manager* vor, welcher, neben dem *Meter*, der die eigentliche Messung durchführt, auch den *Meter Reader*, der die Messdaten einsammelt und damit dem Kollektor entspricht, konfiguriert. In [RFC 2720] wird diesbezüglich auch eine *Meter MIB* spezifiziert, mit welcher der Manager den Meter über SNMP konfigurieren kann.

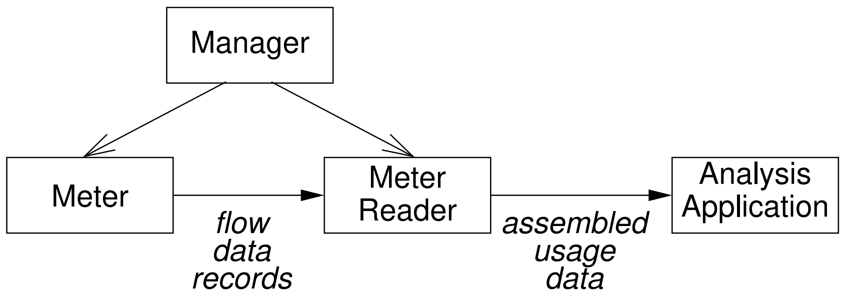

Abbildung 3.4: Die Traffic Flow Measurement Architektur nach [RFC 2722]

Die Metrik, auf Basis derer die Messung durchgeführt wird, ist durch ein sog. *Traffic Flow* definiert. Ein Traffic Flow ist in diesem Fall ein Hilfskonstrukt für das verbindungslose Internet Protocol, das mit einem Aufruf oder einer Kommunikationsverbindung zu vergleichen wäre. Ein Flow ist stattfindender, messbarer Verkehr zwischen zwei Endpunkten, der durch einen Start- und Endzeitpunkt beschränkt wird. Während der Dauer eines Flows werden gewisse, vorher festgelegte Einheiten, wie z.B. IP–Pakete in einer bestimmten Richtung, gemessen, die später für die Abrechnung, Netzplanung, etc. verwendet werden können. Ein Meter verwendet für die Identifizierung von Flows und damit für die Zuordnung von Datenverkehr zu Flows, sog. *Rule Sets*, die durch einen Manager per Konfiguration festgelegt werden. Eine Sprache zur Beschreibung dieser Regeln wird mit der sog. „Simple Rule Set Language (SRL)“ in [RFC 2723] spezifiziert. Desweiteren wird in [RFC 2724] eine Erweiterung der Flow–Definition durch Hinzufügen neuer Attribute diskutiert, um die RTFM–Architektur u.a. auch für die Dienstgütemessung einzusetzen. Die Working Group *IP Performance Metrics (IPPM)* beschäftigt sich dediziert mit der Definition von neuen Metriken [RFC 2330], mit denen die Leistung und Qualität eines IP–basierten Netzes bewertet werden kann. Mit *NeTraMet*[6] wird zur Evaluierung der RTFM–Architektur eine frei beziehbare Referenzimplementierung angeboten.

Lag mit den bisher vorgestellten Spezifikationen der Fokus innerhalb der IETF bzgl. der Nutzungserfassung v.a. auf der Messung, wird in [RFC 2975] nun die Architektur explizit auf Anforderungen hinsichtlich der *Übertragung* von Messdaten im Allgemeinen und Abrechnungsdaten im Besonderen hin untersucht. Hierzu werden die Schnittstellen zwischen den beteiligten

[6] `http://www2.auckland.ac.nz/net//Accounting/ntm.Release.note.html`

Komponenten anhand der in Abbildung 3.5 dargestellten Architektur analysiert.

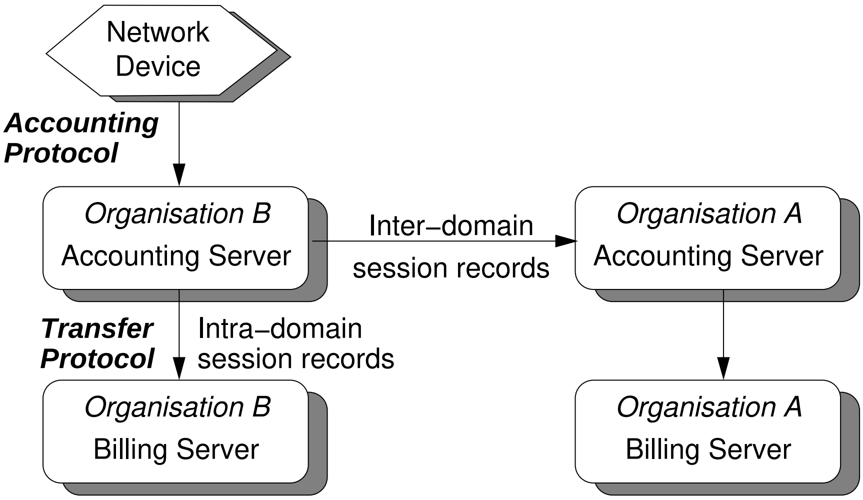

Abbildung 3.5: Die Internet Accounting Architektur nach [RFC 2975]

Insbesondere wird nun explizit zwischen einem *Accounting Protocol* und einem *Transfer Protocol* unterschieden. Das Accounting Protocol wird zwischen einer Netzkomponente, an der (Mess–)Daten anfallen, und einem Accounting Server, der die Daten einsammelt und weiterverarbeitet, eingesetzt. Die mit Hilfe des Accounting Protocols ausgetauschten Daten werden allgemeiner aufgefasst (z.B. Rechnerkennungen, etc.), so dass es sich nicht unbedingt um RTFM–Messdaten auf Basis von Flows handeln muss. Das Transfer Protocol wird hingegen zwischen einem Accounting Server und Billing Server eingesetzt. Bei den hierbei ausgetauschten Daten handelt es sich explizit um für Abrechnungszwecke aggregierte Daten (*Session Records*). Desweiteren wird in diesem Zusammenhang zwischen intra– und interorganisationellen Session Records unterschieden. In [RFC 2924] werden zudem bisher, durch unterschiedliche Gremien wie der ITU-T, ETSI, etc. standardisierte Abrechnungsdatenformate auf die dort spezifizierten Abrechnungsattribute untersucht, um dies als Basis für die Spezifikation eines neuen IETF–Abrechnungsdatenformats und Accounting–Protokolls zu verwenden. Allerdings ist es bisher nur gelungen, den sog. *Accounting Data Interchange Format (ADIF)* zu definieren, bei dem es sich um einen im Januar 2001 abgelaufenen Internet Draft handelt. Der derzeitige Status wird als „Work in Progress“ bezeichnet, wobei seit längerem keine Aktivitäten mehr in der Working Group bzgl. ADIF zu beobachten sind. In gleicher Weise sind derzeit keinerlei Bemühungen hinsichtlich der Spezifikation eines Transferprotokolls zu erkennen. Dies liegt wahrscheinlich u.a. darin begründet, dass diese Arbeiten innerhalb der IETF durch die erfolgreiche IPDR–Initiative (siehe Abschnitt 3.2.6) als obsolet betrachtet werden.

Mit der Spezifikation eines Accounting Protocols im Kontext von Netzzugangsprotokollen, wie z.B. Einwahlprotokollen, beschäftigt sich die Working Group AAA, deren Arbeiten im Nachfolgenden vorgestellt werden.

 Als Buch erhältlich im Verlag Dr. Hut, München, www.dr.hut-verlag.de (ISBN 3-934767-97-4)

Arbeiten im Kontext der Authentication, Authorization and Accounting (AAA) WG Ziel der AAA WG ist die Spezifikation eines neuen Dienstzugangsprotokolls (DIAMETER), das den Anforderungen heutiger Anwendungsszenarios bzgl. der Authentifizierung, Autorisierung und Abrechnung genügt. Hierfür werden sowohl bestehende AAA–Protokolle wie „Remote Authentication Dial In User Service (RADIUS)" [RFC 2865], als auch die Anforderungen, die aus der Unterstützung von Mobilität resultieren, z.B. durch Einsatz von MobileIP [RFC 2002], analysiert.

Bei RADIUS handelt es sich in erster Linie um einen Authentifizierungs– und Autorisierungsdienst für verschiedenartig realisierte Einwahlzugänge (z.B. per PPP [RFC 1662]). Der Einwahlserver agiert als RADIUS–Client gegenüber einem RADIUS–Server, der die notwendige Authentifizierung und Autorisierung des (eingewählten) Dienstnutzers auf Basis der vom RADIUS–Client übermittelten Daten übernimmt. In [RFC 2866] ist eine Erweiterung des RADIUS–Protokolls hinsichtlich der Abrechung spezifiziert, so dass ein Radius Server als vollständiger AAA–Server bzgl. Einwahldiensten agiert. Die Erweiterung sieht v.a. die Definition von neuen Protokollattributen vor, auf Basis derer Daten zu Abrechnungszwecken übertragen werden können. Um sowohl RADIUS–Clients als auch –Server in ein gegebenes Managementumfeld miteinzubeziehen, wurden in [RFC 2620] und in [RFC 2621] entsprechende MIBs definiert. Allerdings sehen diese MIBs (bis auf eine Ausnahme[7]) lediglich Managementvariablen für Statistiken vor, mit denen beispielsweise der gegenwärtige Status eines RADIUS–Clients resp. eines Servers bzgl. Accounting–Anfragen, zugeordneten Clients/Servern, etc. abgefragt werden können.

In [RFC 2977] werden AAA–Anforderungen spezifiziert, die insbesondere durch die Nutzung von MobileIP entstehen. Die darin identifizierten Anforderungen sollen insbesondere in die Spezifikation von DIAMETER fließen, das in Zukunft RADIUS ersetzen soll. Die Spezifikation von DIAMETER existiert bisher allerdings lediglich als Internet Draft.

ETSI–Arbeiten im Kontext der Internet–Abrechnung Der Vollständigkeit halber seien an dieser Stelle auch die Bemühungen der ETSI genannt. Ziel der *European Telecommunications Standards Institute (ETSI)*[8] ist die Spezifikation von internationalen Telekommunikations–Standards für den europäischen Kontinent. Beispielsweise liegt der gegenwärtige Fokus auf der Spezifikation von UMTS–Standards.

In [TR 101 734] werden Abrechnungsattribute spezifiziert, die eine einheitliche Abrechnung von Internet–basierten Diensten innerhalb von Europa ermöglichen sollen. Grundsätzlich werden hierbei die im Internet eingesetzten Protokollarchitekturen auf abrechenbare Leistungen untersucht. Neben den bekannten Abrechnungseinheiten (z.B. übertragenes Volumen auf Basis der IP–Pakete, etc.) wurden auch DiffServ/IntServ in die Untersuchungen aufgenommen. Er-

[7]Die RADIUS Accounting Server MIB [RFC 2621] sieht eine Managementvariable zum Neustart des Servers vor.

[8]`http://www.etsi.org/`

gebnis ist ein Katalog von abrechenbaren Leistungen, wobei die überwiegende Mehrheit nur durch Einsatz von DiffServ oder IntServ herangezogen werden können.

Bewertung

Der Schwerpunkt der Betrachtung der IETF liegt auf der Nutzungserfassung im Allgemeinen und der Messung von Datenverkehr im Besonderen. Hierzu wurde die von OSI spezifizierte Accounting–Architektur (siehe Abschnitt 3.2.1) übernommen und auf die Besonderheiten der Internet Architektur adaptiert, indem z.B. passende Metriken definiert wurden. Da jede nutzungsorientierte Abrechnung auf der Messung der eigentlichen Dienstnutzung beruht und damit letztendlich auch von Messungen des Datenverkehrs abhängig sein kann, müssen die Spezifikation der IETF in diesem Bereich beachtet werden. Diese sind durch die weite Verbreitung des Internets von hohem Wert.

Der vollständige Abrechnungsvorgang ist von der IETF allerdings lediglich als Anforderungsquelle und zwar jeweils nur in einigen Spezialfällen (Einwahlzugänge, MobileIP, etc.) miteinbezogen worden. Das Abrechnungsmanagement im Sinne dieser Arbeit, demnach also zur Steuerung des Abrechnungsvorgangs, wurde sogar lediglich bzgl. des Messvorgangs in Form der spezifizierten Meter MIB berücksichtigt.

3.2.6 IPDR Organization

Die *IPDR Organization*[9] ist ein Industriegremium, das sich mit der Spezifikation eines Datenformats zum Austausch von abrechnungsrelevanten Daten, dem sog. *IP Detailed Record (IPDR)*, sowie einem Protokoll für deren Übertragung beschäftigt. Da, wie in Abschnitt 2.2 am Szenario erläutert wurde, bisher einheitliche, standardisierte Schnittstellen zwischen den an der Abrechnung beteiligten Komponenten fehlten, wurde deren Kooperation durch die Unterstützung von proprietären Schnittstellen der jeweiligen Komponenten oder durch Implementierung von Gateways zwischen den Komponenten erreicht. Da sich insbesondere Telekommunikationsunternehmen zunehmend durch Fusion oder Kooperation zusammenschließen, um wettbewerbsfähiger zu werden, ist insbesondere eine einfache, flexible und schnelle Zusammenführung der an der Rechnungsstellung beteiligten Komponenten unabdingbare Voraussetzung für den Erfolg der Kooperation. Folglich erhöhten insbesondere Telko–Unternehmen den Druck auf Hersteller von Billing–Systemen und Kollektorsoftware sich in diesem Bereich auf einheitliche Schnittstellen zu einigen. Ergebnis war die Gründung der IPDR Organization, der sich alle namhaften Hersteller von Billing Systemen, wie z.B. Telcordia und Lucent, von Kollektorsoftware, wie z.B. XACCT und NARUS, sowie auch Telekommunikationsunternehmen, wie z.B. AT&T, angeschlossen haben.

[9] `http://www.ipdr.org/`

 Als Buch erhältlich im Verlag Dr. Hut, München, www.dr.hut-verlag.de (ISBN 3-934767-97-4)

Überblick

Primäres Ziel der IPDR Organisation ist es, das sog. *IP Detailed Record* Datenformat zu spezifizieren, das dem aus der Telefonie bekanntem Call Detailed Record, allerdings für Dienste in Datennetzen, entspricht. Um Schnittstellen zu identifizieren, auf Basis welcher dann anschließend die zu spezifizierenden Protokolle und Datenformate eingeordnet werden, wird zunächst in der derzeit aktuellen Version der Spezifikation [IPDR 311] das in Abbildung 3.6 dargestellte IPDR–Referenzmodell festgelegt. Hierbei basiert das Referenzmodell auf der Telecom Operations Map (TOM) des TM Forums (siehe Abschnitt 3.2.4) und ist dem „Network Data Management"–Prozess der untersten Schicht zuzuordnen. Das in Abbildung 3.6 dargestellte Referenzmodell sieht insgesamt sechs verschiedene Entitäten vor: den Service Consumer, das Service Element, den IPDR Recorder (IR), den IPDR Store (IS), den IPDR Transmitter (IT) und das Business Support System (BSS). Die sich zwischen den einzelnen Entitäten befindlichen Schnittstellen werden von A bis E durchnummeriert. Die Spezifikation [IPDR 311] konzentriert sich auf die Spezifikation des D–Interfaces, dass sich zwischen dem IT und BSS befindet.

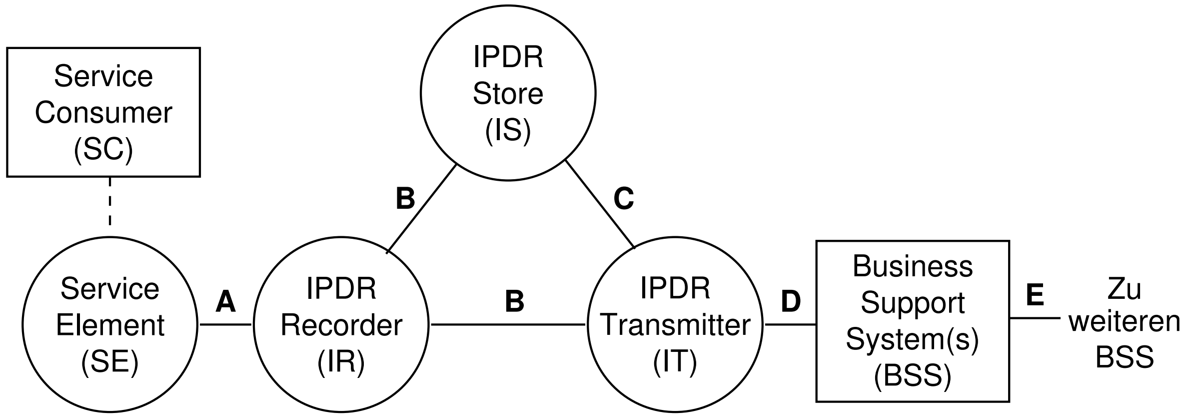

Abbildung 3.6: Das IPDR Referenzmodell nach [IPDR 311]

Die Struktur eines IPDR–Dokuments und damit das Datenformat wird durch Verwendung von *XML–Schema* festgelegt. Damit werden Nutzungs– resp. Abrechnungsdaten, die als IPDR–Dokumente vorliegen, als XML–Dokumente übertragen und können folglich mit üblichen XML–Parsern dekodiert werden. Grundsätzlich sieht das Datenformat eine Unterscheidung zwischen einem *dienstunabhängigen* und einem *dienstspezifischen Teil* eines IPDR–Dokuments vor. Die Struktur des dienstunabhängigen Teil wird mit der sog. *IPDR Master Schema* festgelegt, das mit [IPDR 311] derzeit in der Version 3.1.1 vorliegt. Die dienstspezifischen Teile werden in separaten Dokumenten veröffentlicht.

Das IPDR Master Dokument sieht einen Header und Footer vor, die allgemeine Daten, wie z.B. Versionsnummer und Erstellungsdatum, enthalten. In einem IPDR Master Dokument können mehrere dienstspezifische IPDR Dokumente eingebettet sein, welche die eigentlichen Usage Data Records mit Daten über eine spezifische Dienstnutzung enthalten. Da IPDR richtig erkannt hat, dass es unmöglich ist, allgemeingültige, für alle denkbaren Dienste anwendbare Abrechnungsattribute zu standardisieren, wird einerseits eine Methodik vorgegeben, wie dienstspezifische Abrechnungsattribute spezifiziert werden können, und andererseits werden für unterschiedliche Dienste, wie VoIP, ASP, etc. dienstspezifische Abrechnungsattribute von der IPDR standardisiert. Die dienstspezifischen Abrechnungsattribute ermöglichen die Beantwortung der 5–W–Fragen[10]. Das Datenformat ist derart ausgelegt, dass Erweiterungen ohne weiteres möglich sind.

Wie bereits erwähnt, wird neben der Spezifikation eines Datenformats auch ein Protokoll zur Übertragung von Abrechnungsdaten festgelegt. Das Protokoll ist flexibel, da es sowohl ein Push–, Pull– als auch hybrides Modell zur Initiierung der Datenübertragung vorsieht. Desweiteren wurde beim Protokolldesign auf eine sichere und robuste Datenübertragung geachtet.

Um die Übertragung von IPDR Dokumenten performanter zu gestalten, wurde mit der aktuellen Version auch die Speicherung von IPDR Nutzungsdaten im sog. *XDR Format* spezifiziert. Bei XDR handelt es sich um eine komprimierte, nicht menschenlesbare Form eines IPDR XML Dokuments. In [IPDR 311] wird beschrieben, wie die Abbildung in beide Richtungen realisiert wird.

Bewertung

Die IPDR Organization weist u.a. durch die Festlegung eines Referenzmodells zur eindeutigen Identifizierung der Schnittstellen ein sehr strukturiertes Vorgehen zur Spezifikation sowohl des Datenformats als auch des dazugehörigen Übertragungsprotokolls auf. Das Ergebnis kann als sehr gelungen bewertet werden. Da sich zudem der IPDR Organization alle relevanten Hersteller von Billing–Systemen und Kollektorsoftware angeschlossen haben, ist die Durchsetzung und Verbreitung der Spezifikation wesentlich wahrscheinlicher als bei den bisher als gescheitert anzusehenden Standardisierungsbemühungen der IETF in diesem Bereich.

Dennoch muss an dieser Stelle betont werden, dass hierbei nur eine Standardisierung der *Nutzungsschnittstellen* der an der Abrechnung beteiligten Komponenten vorgenommen wird. Die jeweiligen *Managementschnittstellen* werden nicht betrachtet. Damit wird zwar in Zukunft die Kooperation von IPDR–konformen Komponenten erleichtert, welches das erklärte Ziel der IPDR–Initiative ist, jedoch bleibt das Management dieser Komponenten, z.B. zur Steuerung der kooperierenden Komponenten, davon unangetastet. Damit bleibt das Management der an der Abrechnung beteiligten Komponenten auch mit Unterstützung der IPDR–Spezifikation weiterhin uneinheitlich, isoliert und folglich weiterhin schwierig.

[10] Wer, Wann, Was, Wie, Wo.

 Als Buch erhältlich im Verlag Dr. Hut, München, www.dr.hut-verlag.de (ISBN 3-934767-97-4)

3.2.7 Object Management Group (OMG)

Die *Object Management Group (OMG)*[11] ist ein Ende der 80er Jahre gegründetes Industriekonsortium, das sich mit der Spezifikation von Technologien zur Unterstützung der verteilten Kooperation von beliebigen Software–Bausteinen[12] beschäftigt. Die bekanntesten, der Obhut der OMG obliegenden Technologien sind die *Common Object Request Broker Architecture (CORBA)* und die *Unified Modeling Language (UML)*. Kern von CORBA ist der sog. *Object Request Broker (ORB)*, der für verteilte Objekte den Aufwand zur Kommunikation über Systemgrenzen hinweg senkt, indem die Systemgrenzen transparent erscheinen. In diesem Zusammenhang wurden auch zahlreiche, für eine Middleware typischen Dienste (die sog. *CORBAservices*), wie z.B. Lokalisierungs–, Namens–, Ereignisdienst, etc., als integraler Bestandteil von CORBA spezifiziert. In zahlreichen Working Groups der OMG werden nun zusätzlich *bereichsspezifische* Spezifikationen durchgeführt. Mit dem Management von Infrastrukturen beschäftigt sich die *Telecommunications Domain Task Force*, die kürzlich in [OMG 02-09-01] die für die Abrechung relevante „Federated Charging and Rating Facility (FCR)“ definiert hat, die im Folgenden erläutert wird.

Überblick

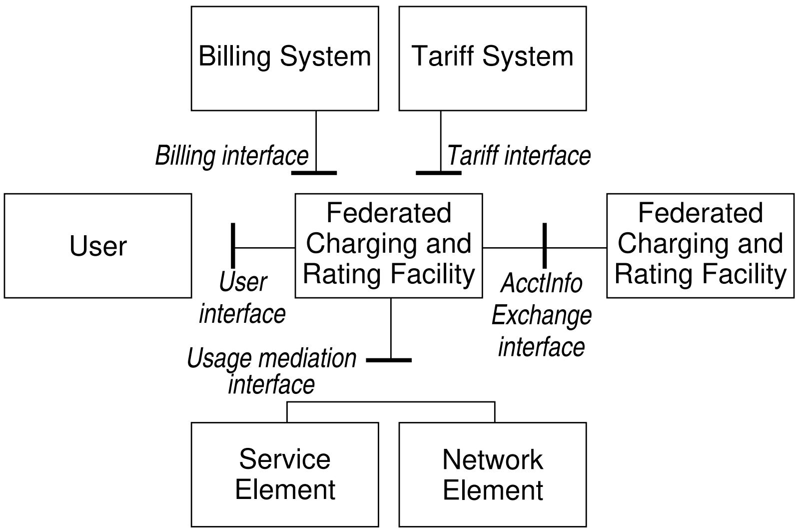

Abbildung 3.7: Die FCR–Architektur aus [OMG 02-09-01]

Die in [OMG 02-09-01] spezifizierte „Federated Charging and Rating Facility (FCR)“ sieht eine Komponente vor, welche Rohdaten von unterschiedlichen Messpunkten einsammelt, diese geeignet aggregiert und auf die so gewonnenen Daten einen passenden Tarif anwendet. Das mit FCR verfolgte Ziel ist die in einer Multiprovider–Umgebung eingebettete Kooperation von verteilten, an der Abrechnung beteiligten Komponenten durch Spezifikation von geeigneten Schnittstellen zu unterstützen. Hierbei werden ausgehend von der in Abbildung 3.7 dargestellten Referenzarchitektur insgesamt fünf Schnittstellen identifiziert. Auf Basis von Use Cases wird strukturiert die notwendige Funktionalität der identifizierten Schnittstellen untersucht. Die daraufhin festgelegten Schnittstellen sehen jeweils ausschließlich Methoden zum Versenden und Empfangen von

[11] `http://www.omg.org/`

[12] Der Begriff *Software–Baustein* umfasst sowohl einzelne Objekte als auch vollständige Software–Produkte.

Daten in unterschiedlichen Formaten, wie z.B. als IPDR–Dokument oder CORBA Property List, vor. Zum Austausch der Daten wird demnach explizit das von der IPDR Organization standardisierte Datenformat für Abrechnungsdaten (siehe Abschnitt 3.2.6) unterstützt.

Bewertung

Der Fokus der in [OMG 02-09-01] spezifizierten FCR liegt auf der Unterstützung der Kopplung von Abrechnungssystemen in Multiprovider–Umgebungen. Sofern die an der Abrechnung beteiligten Komponenten die spezifizierten Schnittstellen *verwenden*, z.B. im Fall der Service resp. Network Elements, bzw. die spezifizierten Schnittstellen *implementieren*, z.B. im Fall der Billing resp. Tariffing Systeme, kann die Kooperation von heterogen zusammengestellten Abrechnungskomponenten tatsächlich einfacher realisiert werden. Allerdings kann die Durchsetzung der Spezifikation und damit des Ansatzes zum gegenwärtigen Zeitpunkt nicht beurteilt werden.

Allerdings muss auch in diesem Fall festgestellt werden, dass nur die Nutzungsschnittstellen der an der Abrechnung beteiligten Komponenten adressiert werden, während die Managementschnittstellen, mit denen beispielsweise die Kooperation der Komponenten gesteuert und überwacht werden kann, außer Acht gelassen werden. Somit bleibt mit dem vorgestellten Ansatz das Abrechnungsmanagement unangetastet und folglich weiterhin isoliert und uneinheitlich.

3.3 Forschungsarbeiten

In diesem Abschnitt werden Forschungsarbeiten auf dem Gebiet der Abrechnung und des Abrechnungsmanagements vorgestellt. Es sei bereits an dieser Stelle angemerkt, dass sich die überwiegende Mehrzahl der Forschungsarbeiten mit der Tarifierung auseinandersetzen. Ein Hauptteil der Untersuchungen besteht darin, ein ideales *Tarifmodell* für netzbasierte Dienste zu definieren und dieses geeignet durchzusetzen. Desweiteren existiert eine geringe Anzahl von Arbeiten, die sich mit dem Abrechnungsmanagement, wie es in dieser Arbeit verstanden wird, beschäftigen.

Nachfolgend werden die Arbeiten aus beiden Bereichen vorgestellt, wobei Forschungsarbeiten aus dem Bereich der Tarifierung nur überblicksartig präsentiert werden, da diese nicht zum Kern der Arbeit gehören.

3.3.1 Forschungsarbeiten im Bereich der Tarifierung

Wie bereits erwähnt, existieren zahlreiche Forschungsarbeiten zum Thema der Tarifierung. Eine Schwierigkeit liegt hierbei in der Entwicklung von *ökonomischen Modellen*, auf Basis derer

 Als Buch erhältlich im Verlag Dr. Hut, München, www.dr.hut-verlag.de (ISBN 3-934767-97-4)

Tarifmodelle entwickelt werden. Tarifmodelle dienen als Grundlage für die Wahl resp. Vereinbarung eines Tarifs zwischen Kunde und Dienstleister für einen gegebenen Dienst. Hierfür enthält ein Tarifmodell bereits eine Menge von Abrechnungseinheiten, so dass nur noch der *Preis* für die Abrechnungseinheit zwischen Kunde und Anbieter festgelegt werden muss. Sind die Preise festgelegt, so ergibt dies den anzuwendenden Tarif. Hierbei hat der Dienstleister das Interesse, mindestens seine Kosten bei der Dienstbereitstellung zu erwirtschaften. Die Schwierigkeit liegt nun darin, dass ein entwickeltes Tarifmodell, im Sinne einer *Rückkopplung*, sowohl die Wahl eines konkreten Tarifs (Was kann sich der Kunde leisten?) als auch das Nutzungsverhalten (Wie kann der Nutzer den Dienst maximal ausreizen?) beeinflusst. Das Nutzungsverhalten wiederum beeinflusst die entstehenden Kosten beim Dienstleister, da in Abhängigkeit davon die Dienstimplementierung geplant und realisiert wird. Sind die Schätzungen bzgl. des Nutzungsverhaltens im Verhältnis zum tatsächlich auftretenden zu gering, so ist der *erfolgreiche* Betrieb des Dienstes gefährdet (Überlast), welches u.U. Vertragsstrafen bei Nichteinhalten von QoS–Vereinbarungen nach sich ziehen kann. Sind die Schätzungen zu hoch, so wird vom Dienstleister mehr in die Infrastruktur investiert als tatsächlich notwendig, so dass die Kosten und damit der Tarif entsprechend höher als notwendig sind. Dies wiederum kann bedeuten, dass der Tarif aufgrund der Höhe von den Kunden nicht angenommen wird, so dass im Endeffekt die Kosten nicht gedeckt werden.

Damit muss das ökonomische Modell sowohl die Wahl des Tarifs *als auch* das damit zusammenhängende Nutzungsverhalten als *Schätzung* enthalten, damit das Ziel, mindestens die entstehenden Kosten zu decken, erreicht wird. Diese Schätzungen werden meist durch statistische Analysen gewonnen (z.B. durch Anwendung der Methoden der *Ökonometrie*[13] [Asse 02]) oder beruhen auf einfache Annahmen.

Die bisher in diesem Bereich veröffentlichten Forschungsarbeiten konzentrieren sich auf die Entwicklung von *Tarifierungsmechanismen*, die bereits auf ein vorher bestimmtes Tarifmodell aufsetzen. Somit stehen die grundsätzlichen Abrechnungseinheiten fest und es geht darum, den Preis für eine Abrechnungseinheit zu konkretisieren. Ein Tarifierungsmechanismus ist nun die prozedurale Beschreibung der Bestimmung resp. der Wahl eines konkreten Tarifs und damit der Festlegung der jeweiligen Preise der Abrechnungseinheiten. Bei den Tarifierungsmechanismen unterscheidet man zwischen *statischen* und *dynamischen* Mechanismen.

Ein statischer Tarifierungsmechanismus setzt, z.B. basierend auf Schätzungen bzw. Annahmen über das zukünftige Nutzungs– und Kundenverhalten, einen Tarif und damit die Preise für die Abrechnungseinheiten für einen bestimmten Zeitraum (z.B. Dauer einer Dienstvereinbarung) fest. Es liegt auf der Hand, dass, falls die Schätzungen sich erheblich von der Wirklichkeit unterscheiden, Fehlplanungen auftreten und damit u.U. keine Kostendeckung zu erreichen ist. Mit der statischen Tarifierung ist allerdings kein zusätzlicher Aufwand bei der Dienstrealisierung zu betreiben.

[13] Die Ökonometrie basiert auf der Regressionsanalyse, in der die zu schätzenden Variablen (endogen) von gegebenen Variablen (exogen) abhängig sind. Desweiteren kann sich diese Abhängigkeit „zeitlich verzögern“, so dass Rückkopplungen ausdrückbar sind.

In einem dynamischen Tarifierungsmechanismus ist das aktuelle Nutzungs– und Kundenverhalten unmittelbar als Rückkopplung in der Tariffestlegung enthalten. Meist wird auf Basis von Messungen der Ressourcenbelegung, die aus dem gegenwärtigen Dienstnutzungsverhalten resultiert, der Preis für die Abrechnungseinheiten festgelegt. Der Preis ist dann nur für einen sehr beschränkten, kurzen Zeitraum gültig (bspw. für die Übertragung genau eines IP–Pakets). Es liegt auf der Hand, dass mit der dynamischen Tarifierung erheblich zusätzlicher Aufwand während der Dienstrealisierung und –bereitstellung verbunden ist, da der Tarifierungsmechanismus direkt mit der tatsächlichen Dienstnutzung verbunden ist. Somit müssen immer Daten über den aktuellen Stand der Dienstnutzung verfügbar sein. Dies hat meist erheblichen Einfluss auf die tatsächliche Dienstrealisierung, falls die Verfügbarkeit der (Mess–)Daten nicht vorgesehen ist.

Im Folgenden werden einige statische und dynamische Tarifierungsmechanismen vorgestellt.

Statische Tarifierungsmechanismen

Paris Metro Pricing (PMP) Dieses Verfahren, das in [Odly 99] vorgestellt wird, basiert auf der einfachen Idee, dass ein Netz in verschiedene, logische Kanäle mit fester Kapazität eingeteilt wird, welche unterschiedlich hohe, nutzungsorientierte Preise haben. Auch wenn im einfachsten Fall mit den verschiedenen Kanälen keinerlei Dienstgütegarantien verbunden sind, wird dem Nutzer resp. dem Kunden mit einem höheren Preis eine höhere Dienstqualität suggeriert. Es wird nämlich davon ausgegangen, dass sich die unterschiedlich hohen Preise regulierend auf das Nutzungsverhalten auswirken. Wird demnach eine höhere Qualität gewünscht, so wird davon ausgegangen, dass der Kunde/Nutzer einen teureren Kanal wählt. Da der Kanal die (teureren) Gebühren nutzungsorientiert berechnet, wird davon ausgegangen, dass nur soviele Ressourcen verwendet werden, wie tatsächlich nötig sind. Zudem besteht die Annahme, dass, wenn die Dienstgüte in einem niedrigeren Kanal zu schlecht wird, die Nutzer in einen höheren Kanal wechseln, in dem diese aber dann ihr Nutzungsverhalten ändern resp. anpassen. Auf diese Weise erfolgt eine Selbstregulierung alleinig durch den Einsatz unterschiedlicher Tarife. Abwandlungen dieses Verfahrens sehen auch beispielsweise den Einsatz von Prioritäten vor. PMP wird als die einfachste Umsetzung eines DiffServ–Verfahrens angesehen.

Cumulus Pricing Scheme (CPS) Der u.a. in [SGFR 01, SRGF 01] veröffentlichte Tarifierungsmechanismus, der innerhalb des EU–geförderten Projekts *Market Managed Multiservice Internet (M3I)*[14] entwickelt wurde, basiert auf einem Vertrag (dem sog. *Cumulus Pricing Contract*) zwischen Kunde und Anbieter, in dem der Kunde den erwarteten Bedarf des Nutzers, beispielsweise hinsichtlich der benötigten Bandbreite, festlegt. Unterscheiden sich allerdings die Angaben vom Kunden von dem tatsächlichen Nutzungsverhalten, werden sog. *Cumulus Points (CP)* vom Anbieter vergeben. Überschreitet das Nutzungsverhalten die vertraglichen Angaben des Kunden, so werden rote CPs vergeben. Hingegen, wenn das Nutzungsverhalten den CPC un-

[14] `http://www.m3i.org/`

terschreitet, werden grüne CPs vergeben. Die CPs werden jeweils miteinander verrechnet. Wird nun in einem im CPC festgelegten Zeitraum ein vertraglich festgelegter Schwellwert an erhaltenen CPs überschritten, so wird ein Eskalationsmechanismus ausgelöst. Dies kann bedeuten, dass bei roten CPs die Dienstnutzung teilweise oder vollständig vom Anbieter eingeschränkt wird. Bei grünen CPs könnte eine Neuverhandlung des Vertrags initiiert werden. Sowohl die Anzahl der Eskalationsstufen als auch die dann in Kraft tretenden Reaktionen werden im CPC vereinbart. CPS ist ein sehr guter Kompromiss zwischen wirtschaftlicher Realisierbarkeit und technischer Durchführbarkeit. Das Tarifmodell ist für den Kunden vorhersehbar. Desweiteren ist der Vertrag sowohl kunden– als auch anbieterorientiert ausgelegt, da durch die vereinbarten Reaktionsschwellen die Neuverhandlung von Verträgen bei geändertem Nutzungsverhalten miteinbezogen wird. Ein ähnlicher Tarifierungsmechanismus wird vom Verein Deutsches Forschungsnetz (DFN) betrieben.

Dynamische Tarifierungsmechanismen

Smart Market Die grundsätzliche Idee des in [MaVa 93] erstmals veröffentlichten Verfahrens ist, dass der Preis jeder Nachricht (bspw. eines IP–Pakets) vom Grad der momentan im Netz herrschenden Überlast durch eine Art Auktion zwischen den sendewilligen Teilnehmern bestimmt wird. Hierbei setzt sich der Preis zusammen aus einem pauschalen Betrag, der immer anfällt, und einem nutzungsorientierten Betrag, der von der Stausituation im Netz abhängig ist. Da der ermittelte Preis von der gegenwärtigen Überlastsituation abhängig ist, gilt dieser prinzipiell immer nur für eine zu übertragene Nachricht und muss damit für jede Nachricht neu festgestellt werden. Wie bereits erwähnt, wird der Preis für eine Nachricht tatsächlich durch ein *Auktionsverfahren* festgesetzt. Hierfür enthält jede Nachricht in ihrem Header ein Gebot für die Weiterleitung. Die Gateways (bspw. Router) sortieren die Nachrichten nach den Geboten. Sofern eine Überlast besteht, landen die Nachrichten mit niedrigerem Gebot im Puffer und werden ggf. verworfen. Somit werden nur diejenigen Nachrichten weitergeleitet, deren Gebot ausreichend hoch ist. Eine besonderes Merkmal des Verfahrens ist es, dass die weitergeleiteten Nachrichten nicht das eigene Gebot als Gebühr bezahlen, sondern das jeweilig höchste Gebot der Menge der verworfenen Nachrichten. Damit zahlen die erfolgreichen Sender nur die „Überlast"-Kosten, die bei denjenigen Sendern entstanden sind, die aufgrund der Priorisierung der Nachricht mit höherem Gebot *nicht* übertragen wurden. Dieses Verfahren wird deswegen auch als *second price auction* bezeichnet. In [RFS 99] werden verbesserte Auktionsverfahren (Delta Auction, ChiPS) vorgestellt, die z.B. die Verzögerung bei Neugeboten vermindern. Implementiert werden die Verfahren, indem QoS–Mechanismen wie DiffServ oder IntServ bzgl. der Auktionen erweitert werden.

Ex–Post Pricing/ABC Schema Dieses Verfahren sieht eine Gebühr vor, die aus einem pauschalen, vor der Dienstnutzung bekannten Betrag besteht (ex–ante) und einem Betrag, der sich sowohl an der tatsächlichen Dienstnutzung als auch an der im Netz vorherrschenden Situation orientiert. Der Preis für den nutzungsorientierten Teil wird allerdings im Gegensatz zum Smart

Market Verfahren erst nach der Dienstnutzung bestimmt (ex–post), wenn die Situation im Netz aufgrund von gemessenen Daten ausgewertet werden kann. In der Regel wird für die Bestimmung eines Preises für die Dienstnutzung eine monoton steigende Preisfunktion angewendet, die sich beispielsweise an dem Gesamtverkehrsaufkommen, der Stausituation, etc. orientiert. Damit ist im Gegensatz zum Smart Market die zu zahlende Gebühr bekannt und ist nicht durch die Gebote anderer Kommunikationsteilnehmer beeinflußbar. Variationen dieses Verfahrens bestehen nun in der Art und Weise, wie die Preisfunktion festgelegt wird, d.h. wie der Preis im Verhältnis zu einer oder mehreren Verkehrscharakteristika gesetzt wird. Hierzu wurde innerhalb des *CA$HMAN–Projekts* („Charging and Accounting in Schemes in Multiservice ATM Networks"), welches in [Song 99] ausführlich dokumentiert ist, das sog. *ABC–Schema* entwickelt. Das ABC–Schema ermittelt in Abhängigkeit von der *Dauer T*, dem *Volumen V* und einer pauschalen Gebühr *c* die Gesamtgebühr *G* in folgender Weise:

$$G = aT + bV + c,$$

wobei *a* die Gebühr für die Dauer einer Verbindung ist (z.B. Euro/Sekunde) und *b* die Gebühr für das Volumen einer Verbindung ist (z.B. Euro/Mbit). Um nun die Gebühren a und b zu bestimmen, wird die sog. *Effektive Bandbreite* verwendet. Bei der Effektiven Bandbreite handelt es sich um eine *Schätzung* über die erwartete Bandbreitennutzung einer Nutzer–Verbindung in Abhängigkeit von dessen Dienstgüteanforderungen. Die Effektive Bandbreite wird durch statistische Analysemethoden gewonnen, die abhängig von den Ressourcen im Netz (allgemein: *space* s) und einem Zeitintervall ist (allgemein: *time* t). Das Tarifierungsverfahren sieht nun vor, dass der Nutzer vor der tatsächlichen Dienstnutzung seine durchschnittliche Bandbreite angibt, die dieser für die Verbindung in Anspruch nehmen wird. In Abhängigkeit davon und mit der ermittelten Effektiven Bandbreite werden die Preise a und b bestimmt. Je näher der vom Dienstnutzer angegebene Durchschnittsverbrauch an dem tatsächlich in Anspruch genommenen ist, desto günstiger fällt die Preisbestimmung für ihn aus. Das ABC–Schema wurde im Kontext von ATM–basierten Netzinfrastrukturen entwickelt und ist nur in Ausnahmefällen auf verbindungslose Netzinfrastrukturen anwendbar. Desweiteren ist die Bestimmung der Effektiven Bandbreite sehr schwierig und nur solange gültig, wie keine Änderungen an der Netzkonfiguration durchgeführt werden, auf Basis derer die Schätzung ermittelt wurde.

Fazit und Bewertung

Im Rahmen dieses Abschnitts konnte nur ein selektiver Überblick über einige statische und dynamische Tarifierungsmechanismen gegeben werden. Dennoch kann generell gesagt werden, dass sich *dynamische* Tarifierungsmechanismen nicht für den Einsatz in Outsourcing–Szenarios eignen, da diese einerseits hohe Anforderungen an die dienstrealisierende Ressourcen stellen (Gebote bewerten und ggf. neuen Vorschlag an Sender schicken) und andererseits die Infrastruktur direkt erweitert werden muss (z.B. gegebene QoS–Mechanismen anpassen), um das Verfahren zu implementieren. Zudem sehen die bisherigen Veröffentlichungen meist eine Vefahrensrealisierung auf Transitebene vor, so dass anwendungsorientierte Dienste, bis auf einige

 Als Buch erhältlich im Verlag Dr. Hut, München, www.dr.hut-verlag.de (ISBN 3-934767-97-4)

Spezialszenarios wie Video–on–Demand, keine Rolle spielen. Insgesamt ist der Aufwand für dynamische Verfahren bisher sowohl in der Implementierung als auch später im Betrieb zu hoch. Verfahrensrealisierungen existieren deswegen bisher nur in laborartigen Testumgebungen.

Statische Tarifierungsmechanismen sind hingegen wesentlich einfacher umzusetzen, da sie meist keine (direkte) Anpassung der Infrastruktur erfordern, um das Verfahren selber umzusetzen[15]. Zusätzlich erfüllt es per se die an die Tarifierung gestellten Anforderungen wie Vorhersehbarkeit. Das CPS–Verfahren ähnelt zudem Tarifierungsmechanismen, wie sie in heutigen Outsourcing–Szenarios eingesetzt werden.

Für eine Beschreibung weiterer Tarifierungsmechanismen wird der interessierte Leser auf [SRL 01] verwiesen.

3.3.2 Forschungsarbeiten im Bereich des Abrechnungsmanagements

Wie bereits erwähnt, existieren im Gegensatz zu den Arbeiten der Standardisierungsgremien bisher kaum Forschungsarbeiten auf dem Gebiet des Abrechnungsmanagements außerhalb der Tarifierung. Alle bekannten Arbeiten konzentrieren sich meist auf eine konkrete Problematik eines Teilvorgangs innerhalb der Abrechnung und vermeiden einen ganzheitlichen Blick auf die Abrechnung und das Abrechnungsmanagement.

Beispielsweise beschäftigt sich die in [SAWW 01] vorgestellte Arbeit mit einer prototypischen Implementierung der TINA Accounting Management Architecture (siehe Abschnitt 3.2.2), wobei insbesondere Sicherheitsaspekte adressiert werden.

In [BTLD 01] werden, basierend auf TMForums TOM (siehe Abschnitt 3.2.4) und den Arbeiten der IPDR–Initiative (siehe Abschnitt 3.2.6), Anforderungen und Schnittstellen bzgl. der Gebührenberechnung in einer Multiprovider–Umgebung identifiziert. Die hierbei erzielten Ergebnisse sind direkt in das OMG–Dokument [OMG 02-09-01] geflossen, das bereits in Abschnitt 3.2.7 besprochen wurde.

In [Boet 90] wird ein Abrechnungsmanagement für Dienste in Datennetzen entwickelt. Auch wenn alle 4 OSI Teilmodelle (Informations–, Funktions–, Kommunikations–, Organisationsmodell) hinsichtlich des Abrechnungsmanagements untersucht werden, liegt doch der Schwerpunkt der Analyse auf der Festlegung eines Informationsmodells, indem generische Managementobjekte spezifiziert werden, und eines Funktionsmodells, indem generische Abrechnungsmanagementdienste spezifiziert werden. Damit werden explizit generische Managementschnittstellen für die zu steuernden und zu überwachenden Komponenten entwickelt. Hier-

[15] Es liegt auf der Hand, dass, wenn ein nutzungsorientierter Tarif vereinbart wurde, Messungen der Dienstnutzung erfolgen müssen und deswegen u.U. Eingriffe in die Infrastruktur notwendig sind. Allerdings erfordert die Implementierung des *Tarifierungsverfahrens* keinen zusätzlichen Aufwand (wie z.B. das Versenden von zusätzlichen Nachrichten, etc.).

zu werden, auf Basis einer umfassenden Anforderungsanalyse, die Teilvorgänge der Abrechnung hinsichtlich der für das Abrechnungsmanagement relevanten Aspekte untersucht. Für ausgewählte OSI–Anwendungsdienste (Directory, FTAM und MHS) wird gezeigt, wie diese in das Abrechnungsmanagement durch Anwendung der entwickelten Modelle integriert werden können. Die in [Boet 90] entwickelten Modelle sind trotz der starken OSI–Orientierung zweifellos auch zum gegenwärtigen Zeitpunkt von hohem Wert, da die entwickelten abrechnungsspezifischen Management„schnittstellen" auch für nicht–OSI–Dienste und in einer nicht–OSI–Managementumgebung anwendbar sind[16]. Allerdings wird das Abrechnungsmanagement nur in der Betriebsphase betrachtet, so dass die Teilfunktionen früher Lebenszyklusphasen außer Acht gelassen werden. Gleichfalls wird die, speziell in Outsourcing–Szenarios auftretende Dynamik im Change–Management nicht berücksichtigt.

Da eine Schwierigkeit im Abrechnungsmanagement darin liegt, zu identifizieren, was an abrechnungsrelevanten Informationen tatsächlich für einen spezifischen Dienst zur Verfügung gestellt und damit gemessen werden muss, wird in [Schw 97] eine Methodik zur allgemeinen Analyse und Spezifikation von abrechnungsrelevanten Managementinformationen entwickelt. Hierbei wird ein kombiniertes Top–Down– und Bottom–Up–Vorgehen vorgeschlagen. Ausgehend von Abrechnungspolitiken, die beispielsweise aus Unternehmensrichtlinien abgeleitet sind, werden im Top–Down–Vorgehen abrechungsrelevante Templates abgeleitet. Im Bottom–Up–Vorgehen werden auf Basis des OSI–Schichtenmodells die an der Diensterbringung beteiligten Entitäten an den SAPs erkannt und die im vorhergehenden Schritt entwickelten Templates spezialisiert. Damit ist die Identifikation der relevanten Messpunkte möglich. Zudem kann anschließend eine dienstspezifische Management Information Base (MIB) abgeleitet werden. Mit dem vorgestellten Ansatz wird zweifellos ein relevantes Problem im Abrechnungsmanagement gelöst. Allerdings wird der Abrechnungsvorgang und das Abrechnungsmanagement nicht entlang des vollständigen Lebenszyklus betrachtet und somit auch in diesem Fall die zu unterstützende Dynamik, die aus den notwendigen Managementaktivitäten in der Change–Phase resultiert, nicht adressiert.

Um eine Flexibilisierung im Abrechnungsmanagement zu erreichen, wird in [HaCa 99] eine policy–basierte Architektur für die Konfiguration von Messkomponenten, Gebührenberechnungskomponenten sowie Rechnungsstellungssoftware vorgeschlagen. Bei Policies handelt es sich um regelähnliche Ausdrücke in einer einheitlichen Sprache, die mit einer auf das Wesentliche reduzierten Skriptsprache zu vergleichen wäre. Hierbei soll die Konfiguration jedes Komponententyps durch typspezifische Policies erreicht werden. Die in [HaCa 99] vorgestellte Arbeit fokussiert auf Charging–Policies zur Umsetzung von dienstgüteorientierten Tarifen, die auf Basis einer vorausgesetzten Internet–Dienstgüte–Architektur (DiffServ/IntServ) durchgesetzt werden sollen. Es wird allerdings explizit keine Policy–Sprache angegeben und statt dessen auf die Internet–Policy–Architektur verwiesen. Im Rahmen dieser Architektur wurde jedoch bisher ebenso keine Policy–Sprache festgelegt, so dass die Art und Weise der Umwandlung

[16]Die OSI–konformen Spezifikationen der Teilmodelle, wie z.B. das Informationsmodell, müssten in die entsprechende Managementarchitektur, wie z.B. das Internet Management, „transformiert" werden.

 Als Buch erhältlich im Verlag Dr. Hut, München, www.dr.hut-verlag.de (ISBN 3-934767-97-4)

von Policies in konkrete Managementanweisungen unklar bleibt. Damit zusammenhängend fehlt eine Beschreibung der genauen Prozedur der Herleitung von Policies und damit, wie diese für eine Komponente spezifiziert werden können. In einer Nachfolgearbeit [CZS 01] der gleichen Forschergruppe wird dies allerdings für die Konfiguration von Messkomponenten der Internet–RTFM–Messarchitektur (siehe Abschnitt 3.2.5) auf Basis von NeTraMet untersucht. Hierbei wird ausgehend von einem SLA, der aus den drei Bestandteilen *Meter Characteristics*, *User Profile* und *Tariff* besteht, ein *Policy Translator* verwendet, um eine *Meter Policy* durch „Kopieren" der relevanten SLA–Daten zu erzeugen. Bei der Meter Policy handelt es sich um eine auf das Wesentliche reduzierte *Rule Set* der NeTraMet Simple Rule Set Language (SRL) [RFC 2723]. Damit müssen laut [CZS 01] in einem SLA bereits detailliert technische Charakteristika für die Meter–Konfiguration spezifiziert werden, was aber den in dieser Arbeit aufgestellten Anforderungen der Kunden– und Dienstorientierung widerspricht. Die Arbeit der Forschergruppe wurde in der IRTF–Gruppe *Authorization, Authentication and Accounting Architecture (AAAArch)* fortgesetzt, die kürzlich mit [RFC 3334] hierzu einen *Request for Comment (RFC)* veröffentlicht hat. Im Wesentlichen beinhaltet der RFC die Ergebnisse der bereits genannten Arbeiten sowie zusätzlich eine Diskussion von Alternativen der Aufteilung der Architektur. Allerdings fehlt auch in diesem Fall die Spezifikation einer Policy–Sprache sowie die detaillierte Betrachtung der übrigen, an der Abrechnung beteiligten Komponenten. Das Management von Komponenten außerhalb der Betriebsphase wird gleichfalls nicht betrachtet.

Fazit und Bewertung

Im Grunde ist der zuletzt vorgestellte Ansatz, welcher Policies im Abrechnungsmanagement einsetzt, als durchaus vielversprechend in Hinblick auf die Erfüllung der Anforderungen, die aus dem in dieser Arbeit fokussierten Szenario der Individualdienste resultieren, zu bewerten. Policies eignen sich insbesondere für eine integrative Managementsicht sowie zur Abstraktion und damit Vereinfachung des Managements. Zudem kann man mit Anwendung policy–basierter Ansätze ein dynamisches Management erreichen, das sich auf den gegenwärtigen Status einer zu managenden Entität einstellt. Die in [HaCa 99, CZS 01, RFC 3334] veröffentlichten Arbeiten bieten v.a. eine detaillierte und fundierte Beschreibung der Konfiguration von RTFM–konformen Messkomponenten mittels Policies, mit denen die Überwachung und Messung von IP–basiertem Netzverkehr möglich ist. Allerdings stellt diese Art der Messkomponente, insbesondere in Outsourcing–Szenarios, nur eine von vielen dar. Somit fehlt in diesem Kontext bisher, neben der Betrachtung des vollständigen Dienstlebenszyklus und der übrigen an der Abrechnung beteiligten Komponenten, ein generalisierender, umfassender Ansatz, um damit beispielsweise nicht mehr nur auf eine Art von (Mess–)Komponente beschränkt zu sein. In diesem Zusammenhang muss auch eine einheitliche, den Anforderungen gerecht werdende Policy–Sprache definiert werden. Die vorliegende Arbeit wird sich dieser Thematik in den noch kommenden Kapiteln widmen.

3.4 Beispiele für kommerzielle Produkte

In den folgenden Abschnitten werden typische Managementplattformen, Messwerkzeuge, Kollektorsoftware und Billing Systeme hinsichtlich des Abrechnungsmanagements untersucht. Im Vordergrund stehen vor allem Managementgesichtspunkte und weniger die Funktionalität einer gegebenen Komponente. Zum Schluss erfolgt eine abschließende Bewertung.

3.4.1 Managementplattformen

Es sind keine Managementplattformen, die üblicherweise in Datennetzen für das Netz–, System– und Enterprise–Management eingesetzt werden, bekannt, welche das Abrechnungsmanagement als einen Teilfunktionsbereich miteinbeziehen. Prominente Managementplattformen sowie deren nachzukaufenden Managementmodule zur Erweiterung des Funktionsbereichs, wie z.B. *HP OpenView*[17], *IBM Tivoli*[18], *CA Unicenter*[19], etc., betrachten lediglich (in Teilbereichen) das Leistungs–, Konfigurations– und Sicherheitsmanagement. Ein Grund hierfür liegt darin, dass das Abrechnungsmanagement ursprünglich ausschließlich im Telekommunikationsbereich eine Rolle gespielt hat, da Datennetze überwiegend nur zur Rechnervernetzung in organisationsinternen Bereichen eingesetzt wurden und somit die kunden– und dienstorientierten Aspekte, welche sich beispielsweise in einer „verkauften“ Leistung zwischen einem Anbieter und einem Kunden äußern, fehlten.

Alleinig Mess– und Überwachungswerkzeuge, welche grundsätzlich anwendungsunabhängig Messdaten liefern und bisher insbesondere im Leistungs– und Fehlermanagement eingesetzt werden, können gut durch Managementplattformen integriert werden. Da diese Werkzeuge auch für die nutzungsorientierte Abrechnung benötigt werden, kann die Teilfunktion der Nutzungserfassung unter die Kontrolle einer integrierenden Managementplattform gebracht werden. Allerdings liefert eine Managementplattform lediglich eine einheitliche Entwicklungs– und Laufzeitumgebung für Management*anwendungen*. Diese werden entweder durch separate Module des Plattformherstellers nachgekauft oder selber implementiert. Da, wie vorhin schon erwähnt, keinerlei Module für das Abrechnungsmanagement verfügbar sind, müsste eine derartige Anwendung erst implementiert werden.

Nicht selten wird zur Gebührenberechnung und Rechnungsstellung auch betriebswirtschaftliche Software wie *SAP/R3*[20] verwendet. Für diese Art der Anwendungen gibt es mittlerweile separate Erweiterungsmodule, mit denen ein Management dieser Anwendungen über eine Enterprise–Managementplattform, wie z.B. IBM Tivoli oder CA Unicenter, möglich ist. Allerdings liegt der Schwerpunkt dieser Erweiterungen auf der Überwachung des Anwendungsstatus; ein aktives,

[17] `http://www.openview.hp.com/`
[18] `http://www.tivoli.com/`
[19] `http://www.ca.com/`
[20] `http://www.sap.com/`

 Als Buch erhältlich im Verlag Dr. Hut, München, www.dr.hut-verlag.de (ISBN 3-934767-97-4)

steuerndes Eingreifen ist nur sehr begrenzt möglich (z.B. Neustart der Anwendung). Damit werden insbesondere für das Abrechnungsmanagement relevante Aspekte nicht betrachtet.

3.4.2 Messwerkzeuge

Wie bereits erwähnt, existieren zahlreiche Mess– und Überwachungswerkzeuge, die grundsätzlich für die Nutzungserfassung eingesetzt werden können, allerdings ursprünglich für das Leistungs– und Fehlermanagement entwickelt worden sind. Damit ist nicht unbedingt gesichert, dass die Messdaten in denjenigen Messeinheiten vorliegen, wie sie für die Abrechnung, in Form von Abrechnungseinheiten, benötigt werden.

Im Rahmen eines Kooperationsprojekts mit der *DeTeSystem* (mittlerweile: T–Systems), welches in [Fisc 01] teilweise dokumentiert ist, wurden zahlreiche Mess– und Überwachungswerkzeuge untersucht. Es existieren in Abhängigkeit von den Messpunkten folgende Arten von Werkzeugen:

- *Netzanalysetools:*
  Hierbei wird der Netzverkehr überwacht und teilweise/ganz aufgezeichnet. Ein Teil der Werkzeuge unterstützt auch eine „semantische“ Auswertung des überwachten Verkehrs, d.h. es wird eine Protokollanalyse durch Kenntnis des eingesetzten Anwendungsprotokolls vollzogen. Die Überwachung des Netzverkehrs kann sowohl durch separate Analysekomponenten, wie z.B. Probes, als auch direkt in den Koppelelement geschehen (z.B. *Cisco NetFlow*). Produktbeispiele wären: *CompuWare's EcoSCOPE, Visual Networks Uptime* und *Apptitude MeterFlow*.

- *Systemanalysetools:*
  In diesem Fall werden Systemparameter, wie z.B. CPU–Auslastung und Speicherbelegung, einzelnen Threads/Prozessen und damit letztendlich Anwendungen zugeordnet. Momentan existieren für alle Managementplattformen Agenten, die eine derartige Überwachung durchführen, wie z.B. *HP PerfView*.

- *Anwendungsüberwachungstools:*
  In diesem Fall sind sog. ARM[21]–Agenten gemeint, welche Messdaten von ARM–instrumentierten Anwendungen aufzeichnen. Da diese sehr einfach zu realisieren sind, existieren für viele Managementplattformen derartige Agenten (z.B. HP). Allerdings sind bisher kaum ARM–instrumentierte Anwendungen zu finden.

Alle eben genannten Werkzeuge lassen sich meist problemlos in einem integrierten Management einsetzen, da standardisierte Managementschnittstellen, z.B. in Form von Internet–MIBs, implementiert werden.

[21] ARM: Application Response Measurement API [C014, C807].

3.4.3 Kollektorsoftware

Die Funktionsweise von Kollektorsoftware wird anhand von *XACCTusage*[22] erklärt, dem derzeitigen Marktführer in diesem Bereich. Die Software wurde ebenfalls im Rahmen eines Kooperationsprojekts mit der DeTeSystem untersucht.

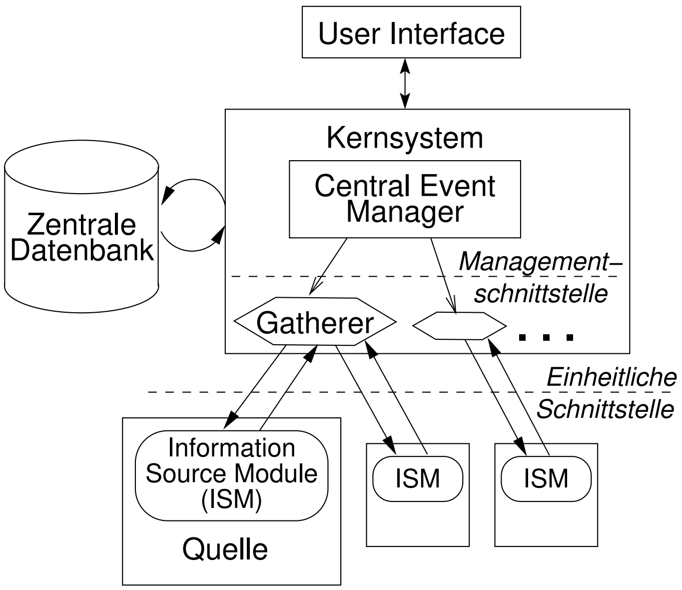

Abbildung 3.8: Die Architektur von XACCTusage

Die grundsätzliche Architektur der Kollektorsoftware XACCTusage ist in Abbildung 3.8 zu ersehen. XACCTusage besteht aus einer *zentralen Datenbank*, in der die Abrechnungsdatensätze gespeichert werden und einem *Kernsystem*, bestehend aus dem sog. *Central Event Manager* und einem oder mehreren sog. *Gatherern*. Ein Gatherer ist der tatsächliche Kollektor, der aus unterschiedlichen Datenquellen die notwendigen Abrechnungsinformationen einsammelt. Der Central Event Manager wird zur einheitlichen Konfiguration der sich auf dem System befindenden Gatherer verwendet. Damit ein Gatherer aus unterschiedlichen Datenquellen die erforderlichen Daten einsammeln kann, werden sog. *Information Source Modules (ISM)* installiert, die einen datenquellenspezifischen Zugriff ermöglichen. So existieren z.B. ISMs für RADIUS, Cisco Netflow, Apache Log Files, etc. Der Gatherer kann hierbei sowohl bzgl. der einzusammelnden Daten als auch bzgl. des eingesetzten Datensammlungsmodells (Event–basiert/Polling) konfiguriert werden. Über ein User Interface ist es möglich, die aggregierten, sich in der Datenbank befindenden Abrechnungsdatensätze zu betrachten. Desweiteren ist es ohne weiteres möglich, automatisch Abrechnungsdatensätze an ein Prebilling– bzw. Billing–System weiterzureichen. XACCT bietet hierbei flexible Konfigurationsmöglichkeiten sowohl bzgl. des Datenformats der Usage Records (das in Abschnitt 3.2.6 vorgestellte, von der IPDR verabschiedete Datenformat wird unterstützt) als auch der Schnittstelle zum Versenden der Datensätze.

XACCTusage bietet keine standardisierte Managementschnittstelle an, mit der es möglich wäre, die Software in eine bestehende Managementumgebung zu integrieren. Damit ist eine Konfiguration und Steuerung des Tools nur über das werkzeugeigene Benutzerinterface möglich. Damit wird ein isoliertes, im Gegensatz zu einem integrierten Management gefördert.

[22] `http://www.xacct.com/`

 Als Buch erhältlich im Verlag Dr. Hut, München, www.dr.hut-verlag.de (ISBN 3-934767-97-4)

3.4.4 Billing Systeme

Wie eingangs bereits erläutert wurde, war das Abrechnungsmanagement für Dienste in Datennetzen bis vor kurzem unüblich, so dass statt dessen in derartigen Fällen bestehende Abrechnungssysteme für den Telekommunikationsbereich hierfür adaptiert wurden. In Telekommunikationsunternehmen wurde das Abrechnungsmanagement allerdings von historisch gewachsenen, höchst proprietären OSS durchgeführt (vgl. die Aussagen in den Abschnitten 3.2.4 und 3.2.6). Die Funktionalität war insbesondere auf die Rechnungsstellung spezialisiert, da die Tarife meist einfach gestaltet waren, während die Anzahl der Nutzer hingegen groß war. Da die OSS keinerlei standardisierten Nutzungs– und Managementschnittstellen für das Abrechnungsmanagement vorsehen, ist deren Integration in eine bestehendes Management, das sich an einer standardisierten Managementarchitektur orientiert, nicht durchführbar. Die heute erhältlichen, sog. *IP Billing Systeme*, die für die Rechnungsstellung für datennetzbasierte Dienste gedacht sind, wurden auf Basis dieser ursprünglich für den Telekommunikationsmarkt gedachten Billing Systeme entwickelt und zeigen vergleichbare, proprietäre Charakteristika, wohingegen eine gewisse Flexibilisierung hinsichtlich der Tarifgestaltung feststellbar ist. In der Regel kann aber ein Billing–Produkt nicht ohne weiteres in Off–the–Shelf–Manier installiert und betrieben werden. Insbesondere in umfangreichen Einsatzszenarios wird ein Billing System von Entwicklern des Herstellers den Kundenwünschen entsprechend angepasst. Bekannte Hersteller von Billing Systemen in diesem Bereich sind *CSG Kenan — Billing Plattform (BP)*[23], *Billing Concepts*[24] und *Convergys*[25].

Auf eine tiefergehende Betrachtung von Billing Systemen wird an dieser Stelle verzichtet und auf [HuTh 02] verwiesen, das detailliert die Realisierung von heutigen Billing Systemen beschreibt.

3.4.5 Fazit und Bewertung

Aufgrund des Ursprungs von Abrechnungssoftware im Telekommunikationsmarkt weisen, abgesehen von Werkzeugen zum Messen der Dienstnutzung, die eingesetzten Softwareprodukte (Kollektorsoftware, Billing Systeme) in aller Regel keinerlei standardisierte, noch offengelegte(!) Managementschnittstelle auf. Damit gestaltet sich das Management dieser Komponenten immer uneinheitlich sowie isoliert und wird dadurch unnötig zusätzlich erschwert. Ein integriertes Management von Softwareprodukten im Abrechnungsbereich, wie es in [HAN 99] dargestellt wird, wurde von den Herstellern bisher in keiner Weise angemessen betrachtet.

Da allerdings bereits hohe Investitionen in diesem Bereich getätigt worden sind, muss die in dieser Arbeit neu zu entwickelnde Lösung bestehende Werkzeuge und Softwarekomponenten in das Management integrieren. Die Durchsetzbarkeit einer Lösung, die auf ein vollständiges

[23] `http://www.csgsystems.com/`
[24] `http://www.billingconcepts.com/`
[25] `http://www.convergys.com/`

Redesign und Ersetzen von bestehenden Komponenten setzt, wird aus Gründen des Investitionsschutzes stark angezweifelt.

3.5 Zusammenfassung: Gesamtbewertung existierender Ansätze

In diesem Kapitel wurde eine Vielzahl von Arbeiten sowohl von Standardisierungsgremien als auch Forschungsgruppen vorgestellt und analysiert, die sich mit der Abrechung resp. dem Abrechungsmanagement beschäftigen. Zudem wurden gegenwärtig für das Abrechnungsmanagement verfügbare Produkte auf ihre Fähigkeiten hin evaluiert. Ergebnis der Untersuchungen ist, dass bisher zwar wertvolle Arbeit auf diesem Gebiet geleistet wurde, allerdings noch kein Ansatz entwickelt wurde, welcher die aus der zunehmenden Dienst– und Kundenorientierung resultierenden Anforderungen erfüllt.

Um sich einen besseren Überblick über die Evaluationsergebnisse zu verschaffen, sind in Tabelle 3.1 die Arbeiten, die sich mit dem *Abrechnungsmanagement* beschäftigen, den hierfür relevanten Anforderungen des in Abschnitt 2.3.5 erstellten Anforderungskatalogs gegenübergestellt. In gleicher Art und Weise werden in Tabelle 3.2 die Arbeiten und Produkte, die sich mit der *Realisierung des Abrechnungsvorgangs* in der Betriebsphase beschäftigen, an den hierfür anwendbaren Anforderungen des Anforderungskatalogs gemessen. Aus Platzgründen wird auf die Aufnahme der Tarifierungsverfahren und der hierfür anwendbaren Anforderungen in der Tabelle verzichtet. In den Tabellen wird jeweils mit einem Haken (✓) angezeigt, ob eine Anforderung grundsätzlich erfüllt wird. Eingeklammerte Haken bedeuten, dass die jeweiligen Dokumente sich nicht direkt zu der Anforderung äußern, aber aus anderen Quellen darauf geschlossen werden kann.

Wie insbesondere anhand der Tabelle 3.1 geschlossen werden kann, existiert noch kein Ansatz, der alle an das Abrechnungsmanagement gestellten Anforderungen erfüllt. Insbesondere fehlt bisher ein Managementansatz, der eine ganzheitliche, am Dienstlebenszyklus orientierte Betrachtung des Abrechnungsvorgangs vornimmt. Während ITIL und TMForums TOM/eTOM zwar durch die Prozessorientierung prinzipiell eine integrative, durchaus am Lebenzyklus orientierte Sicht auf den Abrechnungsvorgang erreichen, werden von diesen beiden Gremien nur eine *Analyse* des Abrechnungsvorgangs vorgenommen, jedoch keine Managementlösung entwickelt. Damit fehlt auch jegliche Betrachtung einer Automatisierung von Managementaktivitäten. ITIL und TMForum setzen bei ihren Betrachtung unterschiedliche Schwerpunkte, so dass sich beide Arbeiten gut ergänzen.

Die Standardisierungsgremien OSI/TMN, TINA–C und IETF hingegen spezifizieren für den Teilbereich der Nutzungserfassung *standardisierte Managementschnittstellen*, während die IPDR Organization und die OMG für die Nutzungserfassung *standardisierte Nutzungsschnitt-*

 Als Buch erhältlich im Verlag Dr. Hut, München, www.dr.hut-verlag.de (ISBN 3-934767-97-4)

stellen spezifizieren. Damit werden für die übrigen, an der Abrechnung beteiligten Komponenten keine Spezifikationen durchgeführt. Zudem muss betont werden, dass Schnittstellenstandards zwar für ein integriertes Management sowie für eine effiziente Komponentenkooperation Voraussetzung sind, diese aber keinen Lösungsansatz bieten können, um die steigende Komplexität des Abrechnungsmanagements handhabbar zu machen. Dies ist und bleibt einer hierfür entwickelten *Managementanwendung* vorbehalten. Die Entwicklung einer derartigen Managementanwendung wird allerdings bei vorhandenen, standardisierten Managementschnittstellen erheblich erleichtert.

Die Forschungsarbeiten im Bereich des Abrechnungsmanagement beschäftigen sich u.a. mit der Spezifikation von noch fehlenden Schnittstellenstandards. Lediglich die in Rahmen von [HaCa 99, CZS 01, RFC 3334] vorgestellte Arbeit bietet einen Ansatz, um mittels Policies die Konfiguration von Abrechnungskomponenten flexibler zu gestalten und damit die Komplexität handhabbarer zu machen. Allerdings wird nur die Nutzungserfassung detailliert betrachtet. Die Problematik der Integration von nicht–standard–konformen HW/SW–Komponenten, wie sie momentan mehrheitlich im Abrechnungsbereich anzutreffen sind, wird ebenso wenig adressiert wie die Entwicklung einer einheitlichen, abstrakten Policy–Sprache. Damit weisen auch die in den Arbeiten genannten Policy–Beispiele ein sehr technisches, auf einen Komponententyp zugeschnittenes Niveau auf. Dies wiederum verhindert eine breite, u.U. automatisierte Unterstützung von Change–Managementaktivitäten durch Policies, wie z.B. den Austausch einer Komponente. Somit kann der policy–basierte Ansatz an sich als durchaus erfolgversprechend beurteilt werden, allerdings fehlt bisher die Entwicklung einer auf diesem Ansatz beruhenden Managementlösung, welche die in dieser Arbeit identifizierten Anforderungen erfüllt.

	OSI/TMN	TINAC	ITIL	TMF	IETF	IPDR	OMG	[Boet 90]	[Schw 97]	[RFC 3334]
ALL 1			✓	✓						
ALL 2	✓			✓	✓			✓	✓	
MGT 1			✓	✓				✓		(✓)
MGT 2	✓							✓		✓
MGT 3										✓
MGT 4										
MGT 5										
MGT 6	✓	✓		✓	✓			✓	✓	✓
MGT 7	✓							✓		(✓)
MGT 8					✓			✓		✓
MGT 9	✓				✓			(✓)		
MGT 10	✓	✓			✓			✓		
MGT 11										
MGT 12										
MGT 13								✓		

Tabelle 3.1: Bewertung bisheriger Arbeiten im Abrechnungsmanagement

	OSI/TMN	TINAC	ITIL	TMF	IETF	IPDR	OMG	Mess-komp.	Kollektor-software	Billing Systeme
UAC 1	✓	✓			✓	✓	✓	✓	✓	
UAC 2	✓	✓			✓	✓	✓	✓	✓	
UAC 3	✓	✓			✓	✓	✓	✓	✓	
UAC 4	✓	✓			✓	✓	✓	✓	✓	
CHA 1						✓	✓			
CHA 2						✓	✓			✓
BIL 1		✓								✓
BIL 2		✓								✓
BIL 3		✓								
INK 1	✓	✓				✓				
INK 2										
IMP 1	✓	✓			✓	✓	✓	✓	(✓)	
IMP 2	✓	✓			✓			✓		
IMP 3	✓	✓			✓	✓	✓			
IMP 4	✓				✓			✓	(✓)	
IMP 5										(✓)
IMP 6										✓
IMP 7	✓	✓			✓	✓	✓	✓	✓	
IMP 8	✓				✓			✓	(✓)	(✓)
IMP 9	✓				✓	✓				

Tabelle 3.2: Bewertung bisheriger Arbeiten zur Realisierung der Abrechnung in der Betriebsphase

 Als Buch erhältlich im Verlag Dr. Hut, München, www.dr.hut-verlag.de (ISBN 3-934767-97-4)

Kapitel 4

Entwicklung eines Prozessmodells für das Abrechnungsmanagement

Da bisherigen Ansätzen vor allem eine flexible Unterstützung der Dynamik im Abrechnungsmanagement fehlt, die beispielsweise aus den auftretenden Änderungsaktivitäten resultiert, und dies die geforderte dienst– und kundenorientierte Abrechnung erschwert bzw. unmöglich macht, werden in diesem Kapitel die dynamischen Aspekte des Abrechnungsvorgangs untersucht. Hierzu wird zunächst ein Prozessmodell entworfen, das sowohl Teilprozesse zur Realisierung einer dienstorientierten Abrechnung enthält als auch die Übergänge zwischen diesen darstellt. Desweiteren werden die in den einzelnen Teilprozessen auftretenden Aktivitäten detailliert beschrieben. Im Anschluß daran werden Entitäten, die zwischen Aktivitäten und Teilprozessen ausgetauscht werden, zu einem Abrechnungsdienstmodell, das eine Verfeinerung und Erweiterung des MNM Dienstmodells ist, zusammengefasst. Diese Analyse dient schließlich als Basis für die zu entwickelnde Lösung in den noch folgenden Kapiteln dieser Arbeit.

4.1 Einführung und Vorgehensmodell

Der *Abrechnungsprozess* erbringt in seiner Gesamtheit die vollständige Funktionalität, die für die Bereitstellung einer dienstorientierten Abrechnung notwendig ist. In diesem Kapitel wird der Abrechnungsprozess in *Teilprozesse* gegliedert, die jeweils eine hierfür erforderliche Teilfunktionalität erbringen. In dieser Arbeit werden auch Vorgänge dem Abrechnungsprozess zugeordnet und konsequenterweise in die Prozessbeschreibung mit aufgenommen, die üblicherweise zum technischen Management gezählt werden. Als *Abrechnungsmanagement* wird im Weiteren dieser Arbeit das Management und damit die Unterstützung und Steuerung des im Folgenden beschriebenen Abrechnungsprozesses verstanden.

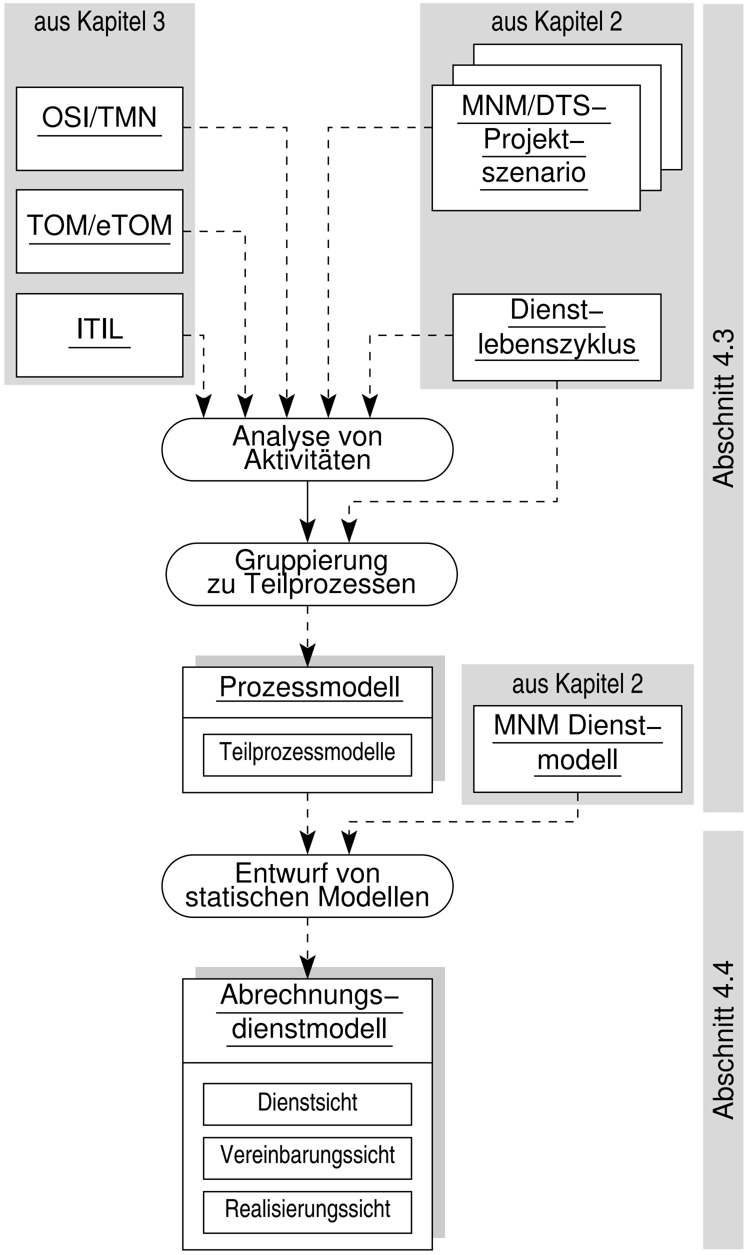

Abbildung 4.1: Vorgehensmodell und Ergebnisse des Kapitels 4

In Abbildung 4.1 ist das generelle Vorgehen, das in diesem Kapitel zur Entwicklung des Prozessmodells angewandt wird, visualisiert. Um die Teilprozesse zu identifizieren und zu strukturieren, wird als Basis der Dienstlebenszyklus aus Abschnitt 2.1.1 verwendet. Hierbei wird jede Lebenszyklusphase auf *Aktivitäten* untersucht, die zur Realisierung einer dienstorientierten Abrechnung notwendig sind. Unter einer Aktivität wird hierbei die innerhalb einer Abstraktionsstufe kleinste ausführbare Einheit verstanden, die von einer Organisationseinheit, einer einzelnen Person oder maschinell durchgeführt werden kann.

Die Untersuchung resp. die Identifizierung von Aktivitäten wurde auf Basis unterschiedlicher Quellen durchgeführt. Sowohl die im Rahmen des MNM-Team–DeTeSystem–Kooperationsprojekts „Abrechnungsmanagement" analysierten, realen Szenarios (siehe hierzu Abschnitt 2.2) als auch die innerhalb der TMN System Management Function Sets (siehe hierzu Abschnitt 3.2.1) identifizierten, generischen Managementfunktionen zur Steuerung des Abrechnungsvorgangs werden mit einbezogen. Desweiteren dienen ebenfalls die durch die ITIL spezifizierten Managementaktivitäten des Abrechnungsmanagements (siehe hierzu Abschnitt 3.2.3) als auch die durch TM Forums TOM identifizierten Informationsflüsse (siehe hierzu Abschnitt 3.2.4) als Grundlage für die in diesem Kapitel durchgeführte Analyse. Trotz der eben explizit erwähnten, verwendeten Quellen und der vielen, in Kapitel 3 vorgestellten Arbeiten, fehlt bisher eine umfassende, prozessorientierte, sich nicht nur auf einen Aspekt konzentrierende Betrachtung der (dienstorientierten) Abrechnung. Zudem werden in den genannten Arbeiten

 Als Buch erhältlich im Verlag Dr. Hut, München, www.dr.hut-verlag.de (ISBN 3-934767-97-4)

die Einzelvorgänge der Abrechnung meist in Prosa formuliert beschrieben. Somit fehlt bisher eine formale Darstellung, die einen ganzheitlichen Überblick über den Abrechnungsvorgang und die zu steuernden Entitäten verschafft.

Die ermittelten Aktivitäten werden anhand einfacher Entscheidungskriterien zu Teilprozessen gruppiert. Die wesentlichen Kriterien gründen sich hierbei auf die zeitliche und räumliche Ausführung von Aktivitäten. Teilprozesse über mehrere Lebenszyklusphasen werden als nicht sinnvoll erachtet, so dass die Phasengrenzen ein natürliches, zeitliches Einteilungskriterium darstellen. Desweiteren wird ebenfalls das Ausführen einer Reihe von Aktivitäten über Organisationsgrenzen hinweg, die keine bzw. kaum Interaktionen zwischen den Beteiligten erfordern, als naheliegendes räumliches Einteilungskriterium angesehen. Zusätzlich wird in der detaillierteren Betrachtung der Teilprozesse jeder Aktivität eine, eventuell leere Menge von *Eingabeentitäten* und *Ausgabeentitäten* zugewiesen. Eine Entität bildet eine Informationseinheit, die zwischen den Teilprozessen respektive den Aktivitäten eines Prozesses ausgetauscht wird. Dies können Ereignisse ebenso wie komplexere Datenstrukturen sein. Da diese schlussendlich ebenso Objekte des Managements darstellen, müssen diese identifiziert werden. Ergebnis dieses in Abschnitt 4.3 durchgeführten Schritts ist ein *Prozessmodell* des Abrechnungsvorgangs, das in zahlreiche Teilprozessmodelle untergliedert ist.

Das Prozessmodell wird daraufhin in Abschnitt 4.4 dazu verwendet, ein statisches *Abrechnungsdienstmodell* zu entwerfen, das relevante Abrechnungsaspekte bei ausgeblendeter Zeitachse visualisiert. Das Abrechnungsdienstmodell ist als eine Erweiterung des MNM Dienstmodells aus Abschnitt 2.1.1 in eine Dienstsicht, Vereinbarungssicht und Realisierungssicht aufgeteilt.

4.2 Überblick über den Abrechnungsprozess

In Abbildung 4.2 ist ein Überblick über die identifizierten, notwendigen Teilprozesse sowie die dazugehörigen Prozessübergänge zur Verwirklichung einer dienst– und kundenorientierten Abrechnung dargestellt. Es ist zu erkennen, dass ein Teil der Teilprozesse die in Abschnitt 2.1.2 genannten Teilfunktionen der Abrechnung sind. In Abschnitt 4.3 dieses Kapitels erfolgt eine detaillierte Beschreibung des visualisierten Abrechnungsprozesses, indem jeder Prozess auf dessen Aktivitäten, Übergänge und Ein–/Ausgabeentitäten untersucht wird.

Die Aufgabe des Abrechnungsmanagements ist es nun, den in Abbildung 4.2 dargestellten Abrechnungsprozess durch Ressourcenbereitstellung etc. zu ermöglichen *und* diesen in der Art und Weise zu steuern, dass die aus der Dienstvereinbarung abgeleiteten Anforderungen erfüllt werden.

Wie bereits in Abschnitt 2.2 festgestellt wurde, verhindert insbesondere die von bisherigen Ansätzen nur unzureichend beachtete *Dynamik* des Abrechnungsprozesses einen effizienten Be-

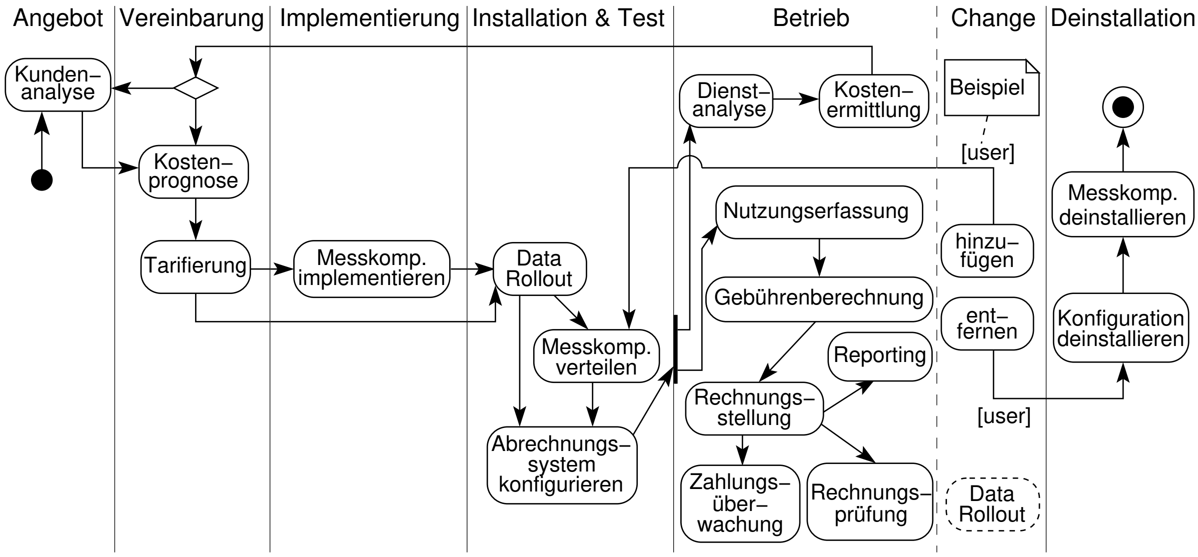

Abbildung 4.2: Der Abrechnungsprozess entlang des Dienstlebenszyklus

trieb des Abrechnungsmanagements. Die Dynamik resultiert teilweise daraus, dass die Reihenfolge des Ablaufs von Aktivitäten innerhalb eines Prozesses in Abhängigkeit von der Dienstvereinbarung unterschiedlich sein kann. Folglich ist die *Implementierung* des Abrechnungsprozesses meist maßgeschneidert und variiert damit von Kunde zu Kunde. Dennoch muss von Seiten des Abrechnungsmanagements ermöglicht werden, dass der Abrechnungsprozess so unabhängig wie möglich von dessen tatsächlicher Implementierung gesteuert werden kann. Insgesamt soll auch weiterhin zwar ein maßgeschneidertes Abrechnungsmanagement verwirklicht werden können, jedoch *ohne* die Notwendigkeit nach einer maßgeschneiderten Implementierung des Abrechnungsmanagements, wie es bisher der Fall ist. Desweiteren erfährt der Abrechnungsprozess zusätzlich insbesondere durch die in der Change–Lebenszyklusphase auftretenden *Änderungsaktivitäten* eine enorme Dynamik: Jede Änderung parallel zum laufenden Betrieb hat z.T. enormen Einfluss auf den Abrechnungsvorgang. Je nach Änderungsaktivität muss eine Kette von weiteren Aktivitäten ausgeführt werden, um einen aus Abrechnungssicht wieder konsistenten Zustand zu erlangen. Wie bereits in Abbildung 4.2 angedeutet, können die Folgeaktivitäten bereits einmal, eventuell in einer früheren Phase, ausgeführt worden sein. Gelingt es, das Anstoßen dieser Aktivitäten aus Managementsicht formal zu beschreiben, so ist eine flexible Unterstützung der auftretenden Dynamik möglich. Dies wird in Kapitel 5 näher untersucht. Voraussetzung hierfür ist allerdings, die Aktivitäten der Change–Phase und deren Auswirkungen zu untersuchen, um anschließend diese Analyse für eine geeignete Unterstützung durch ein Abrechnungsmanagement zu nützen.

Im Folgenden werden die Teilprozesse auf deren Aktivitäten und Ein–/Ausgabeentitäten untersucht. Zum Schluss werden die dabei identifizierten Entitäten und Akteure in einem Abrechnungsdienstmodell zusammengefasst.

 Als Buch erhältlich im Verlag Dr. Hut, München, www.dr.hut-verlag.de (ISBN 3-934767-97-4)

4.3 Analyse der Teilprozesse und Entitäten der Abrechnung

In diesem Abschnitt werden die in Abbildung 4.2 überblicksweise dargestellten Teilprozesse näher untersucht. Ziel ist es, eine möglichst feingranulare, aber dennoch generische Beschreibung der an der Abrechnung beteiligten Aktivitäten und Entitäten zu erstellen. Letztendlich stellen diese Entitäten die Objekte des Managements dar.

Unmittelbares Ziel dieser Analyse ist es die auf diese Weise gewonnene formale, detaillierte, aber dennoch generische Beschreibung der Teilprozesse dazu zu verwenden, um den generellen Ablauf und Umfang der erforderlichen Aktivitäten für eine dienstorientierte Abrechnung nachvollziehen zu können. Da z.T. eine klare Trennung nicht möglich und v.a. nicht sinnvoll ist, werden neben den Abrechnungsaktivitäten auch notwendige Aktivitäten beschrieben, die üblicherweise zum „klassischen“ Management gezählt werden. Dennoch werden in dieser Arbeit, wie einleitend in Abschnitt 4.2 bereits definiert wurde, nur Aktivitäten als zum Abrechnungsmanagement gehörend betrachtet, die die in den nachfolgenden Abschnitten dargestellten Teilprozesse resp. Aktivitäten überwachen und steuern (also demnach z.B. die Ausführung eines Teilprozesses resp. einer Aktivität anstoßen).

Mittelbares Ziel dieses Abschnitts ist es jedoch, die nachfolgende Analyse für die Spezifikation von notwendigen, die Abrechnung unterstützenden Managementanweisungen zu verwenden. Ohne den noch folgenden Kapiteln dieser Arbeit vorgreifen zu wollen, werden in diesem Abschnitt neben (Management–)Objekten auch Ereignisse identifiziert, deren Kenntnis Voraussetzung für eine erfolgreiche Definition von Managementanweisungen ist.

Es muss explizit darauf hingewiesen werden, dass die identifizierten Teilprozesse und Aktivitäten nicht exakt in der nachfolgend beschriebenen Art und Weise in allen Szenarios vorkommen. Es treten Variationen auf, die sich in dem Fehlen eines oder mehrerer Teilprozesse und in der reihenfolgeveränderten Ausführung einzelner, innerhalb eines Teilprozesses auftretender Aktivitäten äußern. Diese Variationen resultieren aus den unterschiedlichen Diensten und Verträgen als auch aus unterschiedlichen Geschäfts– und Unternehmenspolitiken. Folglich dient die Analyse in den folgenden Abschnitten auch dazu, das Spektrum der durch ein Abrechnungsmanagement zu unterstützenden Flexibilität zu untersuchen.

In den nun folgenden Abschnitten wird zunächst eine Notation zur Visualisierung von Prozessen (Abschnitt 4.3.1) erklärt, um daran anschließend diese Notation für die Beschreibung der in Abbildung 4.2 dargestellten Teilprozesse zu verwenden (Abschnitt 4.3.2– 4.3.18).

4.3.1 Verwendete Notation zur Prozessbeschreibung

Die Teilprozesse werden sowohl in Prosa erläutert als auch durch die standardisierte Notation der *Unified Modeling Language (UML)* [UML 1.4, RJB 98] visualisiert. Zur Beschreibung von

Prozessen sind von der UML sog. *Aktivitätsdiagramme* vorgesehen, deren Elemente nachfolgend kurz erläutert werden und in Abbildung 4.3 schematisch dargestellt sind.

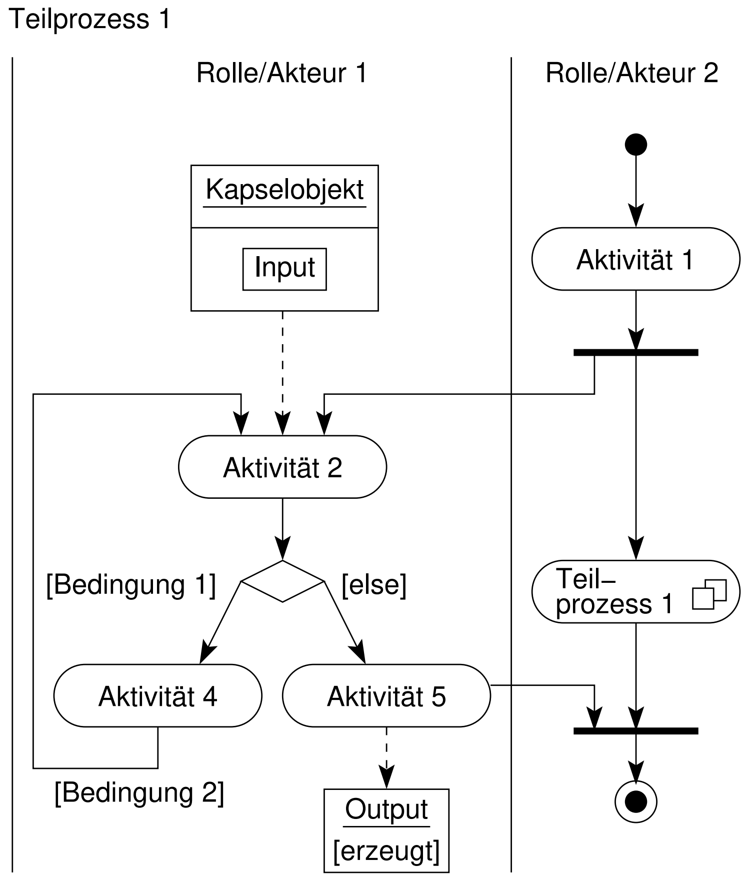

Abbildung 4.3: Beschreibungselemente von Aktivitätsdiagrammen

Ein *Prozess* wird als Ablauf von *Aktivitäten* angesehen, die als kleinste ablaufende Teilschritte auf einer Abstraktionsebene verstanden werden. Eine Aktivität wird innerhalb eines UML Aktivitätsdiagrammes als Kasten mit abgerundeten Ecken visualisiert. Ein *Teilprozess* wird als Aktivität mit zwei sich überdeckenden Quadraten angezeigt. Übergänge zwischen Aktivitäten werden durch gerichtete *Pfeile* zwischen den jeweilig beteiligten Aktivitäten dargestellt, wobei die Gründe für einen Übergang neben den Pfeilen notiert werden können. *Bedingungen* für den Eintritt eines Übergangs werden hingegen zusätzlich in eckigen Klammern geschrieben. Eine Verzweigung innerhalb eines Ablaufs, die eine Entscheidung voraussetzt, wird durch einen *Diamanten* angezeigt, aus dem mehrere Pfade in Kombination mit einer Bedingung führen. Durch den Einsatz von Synchronisationsbalken können einerseits Ablaufstränge parallelisiert als auch wieder zusammengeführt werden. Weiterhin besteht die Möglichkeit, durch Einziehen von senkrechten Linien, Aktivitäten in Zuständigkeitsbereiche (engl.: *swim lanes*), die diese ausführen, einzuteilen. Die einzelnen Bereiche werden Akteuren bzw. Rollen zugeordnet, die jeweils am oberen Rand mittig zwischen zwei Linien notiert werden. Werden Objekte durch Aktivitäten und umgekehrt beeinflusst, z.B. in der Weise, dass Objekte als Eingabe– bzw. Ausgabeentitäten dienen, so wird dies durch gerichtete, *gestrichelte Pfeile* angezeigt, die das jeweilige Objekt mit der dazugehörigen Aktivität verbinden. Das Objekt wird hierbei in der für UML üblichen Notation als Kasten gezeichnet, wobei zusätzlich innerhalb des Kastens die beim Objekt ausgelöste Zustandsänderung in eckigen Klammern notiert werden kann.

In Abweichung von der Standardnotation für UML Aktivitätsdiagramme werden in dieser Arbeit neben Kompositionen zusätzlich auch *Aggregationen* in der für UML üblichen Kompositionsdarstellung als Kasten im Kasten als Ein– und Ausgabeentitäten für Aktivitäten verwendet.

 Als Buch erhältlich im Verlag Dr. Hut, München, www.dr.hut-verlag.de (ISBN 3-934767-97-4)

Diese Notation wird immer dann herangezogen, wenn für die Erklärung von Aktivitäten der Kontext der „umgebenden“ Entität relevant ist, auch wenn nur das dargestellte *innere* Objekt tatsächlich als Ein– bzw. Ausgabeentität fungiert.

Im Weiteren erfolgt eine detaillierte Beschreibung der in Abbildung 4.2 auf Seite 76 identifizierten Teilprozesse in der in diesem Abschnitt vorgestellten Notation.

4.3.2 Kundenanalyse

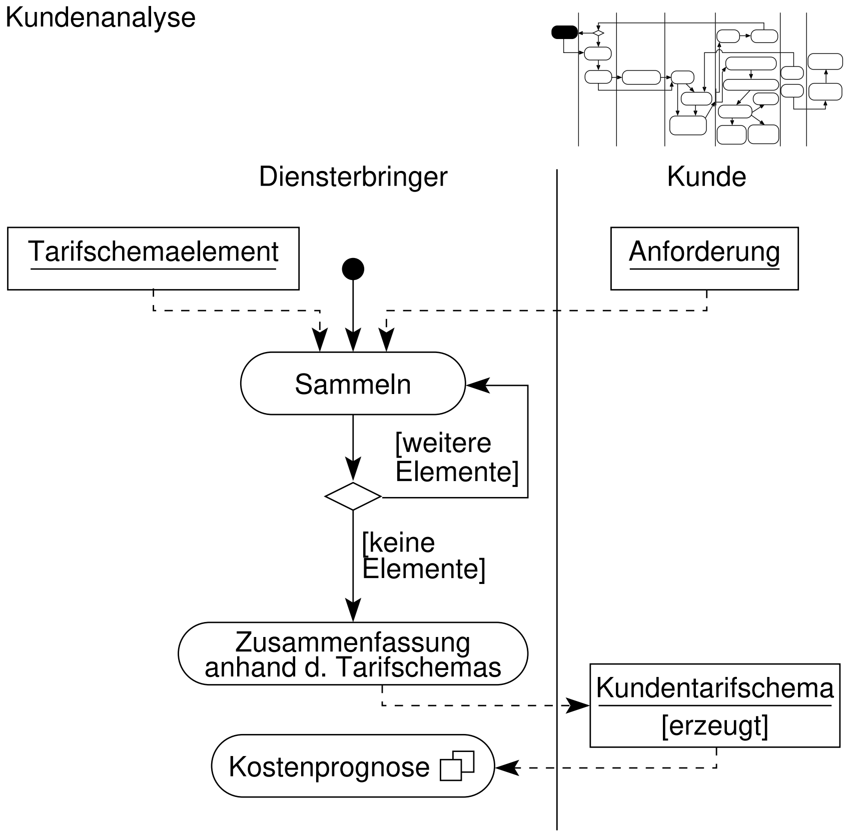

Abbildung 4.4: Kundenanalyse

Der in Abbildung 4.4 dargestellte Teilprozess der *Kundenanalyse*, der in der Angebotsphase des Dienstlebenszyklus stattfindet, beinhaltet im Wesentlichen das Sammeln von Kundenanforderungen, die sich auf tarifrelevante Teile der zu verhandelnden Dienstvereinbarung beziehen. Hierbei werden die einzelnen Elemente eines *Tarifschemas*, wie Abrechnungseinheiten und Preisfunktionen, auf Anforderungen des Kunden hin untersucht. Ein durch diesen Teilprozess erzeugtes *Kundentarifschema* legt hierbei den aus Kundensicht grundlegenden Aufbau eines *Tarifmodells* fest und bestimmt damit primär, welche Abrechnungseinheiten prinzipiell in den Tarif mitaufgenommen werden sollen. Desweiteren kann bestimmt werden, wie die einzelnen Abrechnungseinheiten jeweils im Vergleich zueinander innerhalb eines Tarifs zu gewichten sind und wie die dazugehörigen *Preisfunktionen* zusammengestellt sind, die die zu zahlende Gebühr genau einer Abrechnungseinheit in Abhängigkeit von (Kontext–)Parametern wie erfahrene Dienstgüte, Anzahl gleichzeitig zugreifender Nutzer, etc. festlegt. Eine genaue Betrachtung des Aufbaus eines Tarifmodells und damit eines Tarifschemas erfolgt in Abschnitt 4.3.4, so dass an dieser Stelle darauf verzichtet wird.

Die Granularität der vom Kunden in Form eines Kundentarifschemas spezifizierten Anforderungen kann hierbei stark variieren. Das Spektrum reicht in diesem Fall von der Angabe von abzurechnenden Interaktionen, die als Basis für die Spezifikation von Abrechnungseinheiten dienen sollen, bis zur exakten Angabe eines anzuwendenden Tarifs. Folglich variiert auch die Form des erzeugten Kundentarifschemas, das sowohl in weniger formalisierter Art und Weise als in Prosa formulierte Anforderungsliste vorliegen kann als auch als anwendbares Tarifmodell.

Die Kundenanalyse kann auch mit Fehlen eines explizit existenten Kunden durchgeführt werden. Hierfür werden dann eminente Kundencharakteristika, wie z.B. die erwartete, genutzte Bandbreite, etc. zur Analyse der Kundenanforderungen an den Tarif herangezogen werden.

Das Kundentarifschema dient dem Teilprozess der Kostenprognose in der anschließenden Vereinbarungsphase des Dienstlebenszyklus als Eingabeentität.

 Als Buch erhältlich im Verlag Dr. Hut, München, www.dr.hut-verlag.de (ISBN 3-934767-97-4)

4.3.3 Kostenprognose

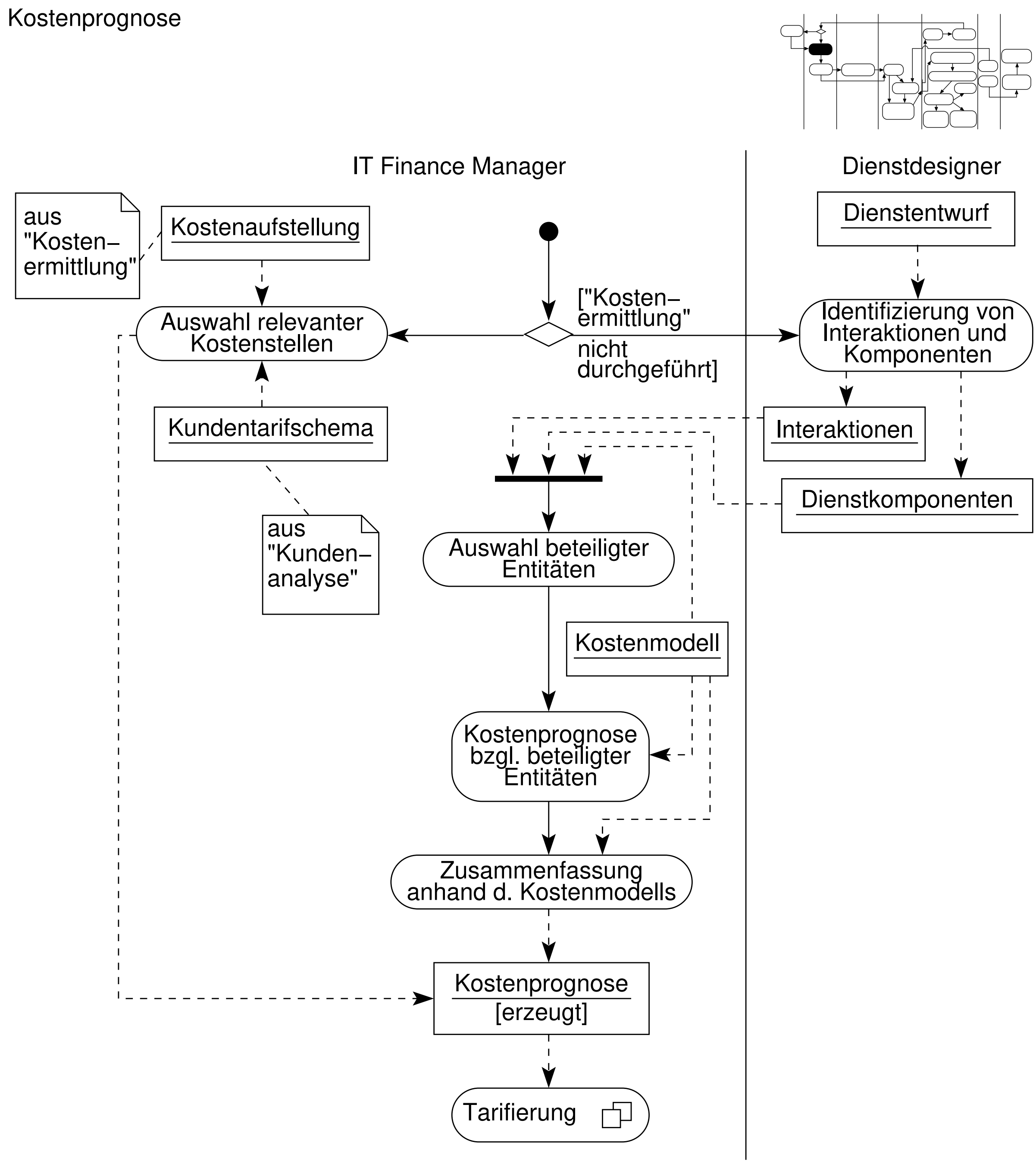

Abbildung 4.5: Kostenprognose

Der in Abbildung 4.5 dargestellte Teilprozess der *Kostenprognose* findet in der Vereinbarungsphase des Lebenszyklus statt. Ziel ist die möglichst exakte Voraussage der zu erwartenden Kosten für die Erstellung, Installation und den Betrieb des von einem Kunden gewünschten Dienstes. Insbesondere wird die Kostenprognose für die Aushandlung eines Tarifs zwischen Diensterbringer und Kunde verwendet, so dass eine möglichst genaue Abschätzung Voraussetzung für den geschäftlichen Erfolg des Dienstleisters ist.

Prinzipiell gibt es zwei Pfade in diesem Teilprozess:

Mit Kostenaufstellung In diesem Fall wird die Kostenprognose auf Basis eines bereits existierenden Dienstes durchgeführt. Bei diesem Dienst kann es sich um einen dem noch zu vereinbarenden Dienst ähnlichen „Fremddienst“ handeln, oder um einen Dienst, der zwischen Dienstnehmer und Dienstleister vereinbart wurde, bereits betrieben wird und für welchen Neuverhandlungen durchgeführt werden sollen. In beiden Fällen kann auf eine vorhandene *Kostenaufstellung* zurückgegriffen werden, welche Ergebnis des Teilprozesses „Kostenermittlung“ ist und in der Betriebsphase des Lebenszyklus stattfindet (siehe Abschnitt 4.3.10). Die Kostenaufstellung enthält die vom *IT Finance Manager* errechneten Kosten eines Dienstes, in dem die tatsächlichen Anschaffungs–, Installations– und Betriebskosten geeignet gewichtet addiert werden. Im Falle des Fremddienstes werden nur die Kostenstellen für die *Kostenprognose* ausgewählt, die den Kundenanforderungen bzgl. des Dienstes und des Tarifs entsprechen. Im anderen Fall wird die Kostenaufstellung ohne weitere Veränderung als Kostenprognose übernommen.

Ohne Kostenaufstellung Fehlt ein existierender Dienst für die Abschätzung der Kosten, führt der *Dienstdesigner* einen groben Entwurf des gewünschten Dienstes durch. Dieser Entwurf wird zur Analyse von Interaktionen und benötigten Komponenten zur Dienstrealisierung verwendet. Der IT Finance Manager verwendet dann den auf diese Weise gewonnenen und verfeinerten Dienstentwurf, um mit Hilfe eines *Kostenmodells* die relevanten Entitäten der Dienstrealisierung für die Kostenprognose auszuwählen. Ein Kostenmodell nach [ITIL 01] dient hierbei als Hilfestellung, da es u.a. Richtlinien enthält, welche Entitäten grundsätzlich für die Kostenprognose miteinbezogen werden sollen. Das Kostenmodell wiederum ist durch den IT Finance Manager durch Einbeziehen von Unternehmensrichtlinien erstellt worden und wird auf Basis von betriebswirtschaftlichen Analysen gewonnen. Da dies außerhalb des Fokus dieser Arbeit liegt, wird auf eine tiefergehende Betrachtung der Erstellung eines Kostenmodells verzichtet und auf [ITIL 01] verwiesen. Im nächsten Schritt wird die tatsächliche Kostenprognose bzgl. der ausgewählten Entitäten durchgeführt. Auch in diesem Schritt bietet das Kostenmodell Hinweise, auf welche Weise Klassen von Entitäten zu bewerten sind[1]. In Anschluss daran werden die errechneten Kosten anhand der Vorschläge des Kostenmodells geeignet in einer Aufstellung zu einer Kostenprognose zusammengefasst. Die Kostenprognose kann hierbei unterschiedlich aufgeschlüsselt sein, z.B. nach Anzahl der Nutzer, pro Monat, etc.

Die Kostenprognose gestaltet sich bisher ohne einen existierenden (Vergleichs–)Dienst sehr schwierig. Wenn keine Erfahrungswerte über die Dienstrealisierung und deren Betrieb vor-

[1]Beispielsweise kann das Kostenmodell vorschlagen, dass für Güter wie Router, Server, etc. nur der Anschaffungspreis in Betracht gezogen wird und nicht die Administrationskosten.

 Als Buch erhältlich im Verlag Dr. Hut, München, www.dr.hut-verlag.de (ISBN 3-934767-97-4)

liegen, wie es für Individualdienste üblicherweise der Fall ist, ist oft nur eine sehr grobe Abschätzung möglich. Da die Kostenprognose auf Seiten des Dienstleisters bei der Tarifierung (siehe den folgenden Abschnitt 4.3.4) eine sehr große Rolle spielt, wird konsequenterweise auch die Tarifierung für Individualdienste als sehr schwierig angesehen. Im folgenden Abschnitt wird die Tarifierung näher beleuchtet.

4.3.4 Tarifierung

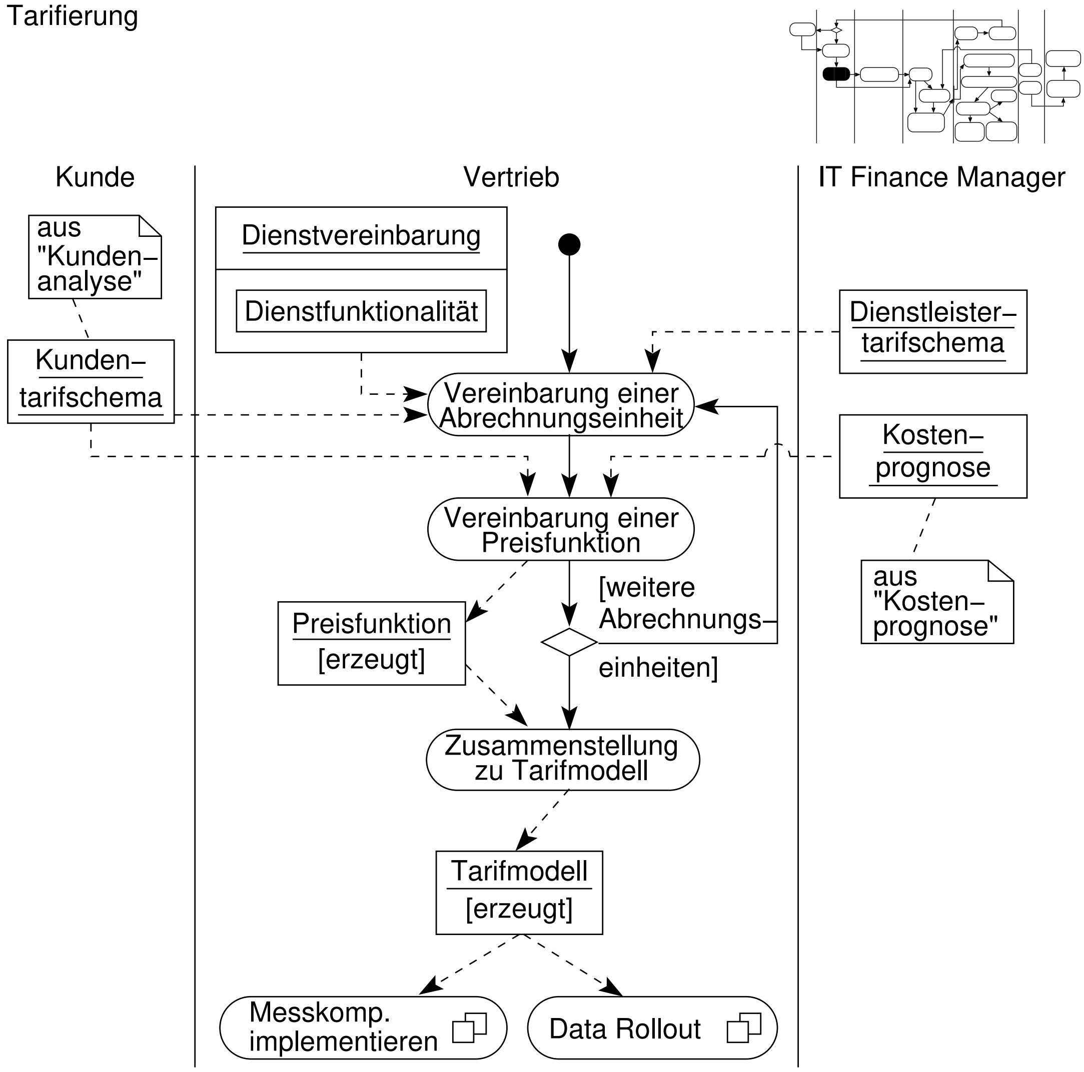

Abbildung 4.6: Tarifierung

Ziel des in Abbildung 4.6 dargestellten Tarifierungsteilprozesses, der in der Vereinbarungsphase des Dienstlebenszykluses stattfindet, ist die Vereinbarung eines *Tarifmodells*[2] zwischen Kunde

[2]Synonym wird auch von einem *Abrechnungsmodell* gesprochen.

und dem *Vertrieb* des Dienstleisters. Da das Tarifmodell für die Gebührenberechnung einer vom Dienstleister regelmäßig erbrachten Leistung, wie Bereitstellung und Betrieb des vereinbarten Dienstes, verwendet werden soll, haben beide Seiten z.T. konkurrierende Ziele bzgl. der Gestaltung des Tarifmodells: Der Dienstleister will zu jedem Zeitpunkt mindestens die entstehenden Kosten decken, während der Kunde daran interessiert ist, nur für die Leistung des Dienstleisters zu bezahlen, die tatsächlich zu seinem Nutzen erbracht wird (also beispielsweise in Abhängigkeit von der tatsächlich erfahrenen Dienstgüte im Gegensatz u.U. zur vereinbarten Dienstgüte).

Nach einer Analyse der bis zum gegenwärtigen Zeitpunkt entwickelten und eingesetzten Tarifmodelle [Radi 02d, Kali 02] kann eine generische Struktur eines Tarifmodells abgeleitet werden, die im Folgenden vorgestellt wird. Diese Tarifmodellstruktur ist im Wesentlichen eine Verallgemeinerung des innerhalb des CA$HMAN–Projekts entwickelten *ABC*–Tarifmodells (siehe hierzu Abschnitt 3.3.1). Die generische Tarifmodellstruktur bestimmt die in diesem Teilprozess zu vereinbarenden Elemente:

$$\begin{aligned} G(x) &= p_1(x)A_1 + p_2(x)A_2 + \ldots + p_k(x)A_k + p_0(x) \\ &= \sum_{i=1}^{k} p_i(x)A_i + p_0(x) \end{aligned}$$

A_i	Anzahl von *Abrechnungseinheiten* des gleichen Typs i, welche in geeigneter Art und Weise das in einem *Abrechnungszeitintervall* (tatsächlich) in Anspruch genommene Dienstnutzungsvolumen bzgl. einer Funktionalitätseigenschaft widerspiegelt (wie z.B. Byte, Paket, Zeiteinheit in ms, belegte Ressourcen, reservierte Ressourcen, etc.)
$p_i(x)$	*Preisfunktion*, welche die *Gebühr* genau einer Abrechnungseinheit des Typs A_i festlegt
$p_0(x)$	pauschale Gebühr für die Dienstnutzung (z.B. konstante Dienstzugangsgebühr), die durchaus aus mehreren Pauschalgebühren zusammengesetzt sein kann
$G(x)$	die vom Kunden im vereinbarten Abrechnungsintervall an den Dienstleister zu entrichtende Gebühr für die Dienstbereitstellung

Ein anzuwendender *Tarif* ergibt sich nun durch das Festsetzen des Parameters x: Damit ergibt sich die Gebühr ausschließlich in Abhängigkeit des Dienstnutzungsvolumens, das durch die Abrechnungseinheiten repräsentiert wird.

Wie in der obigen Aufstellung beschrieben, ermittelt die Preisfunktion $p_i(x)$ die Gebühr für *eine* Abrechnungseinheit A_i in Abhängigkeit des Parameters x. Damit bestimmt der Parameter x die tatsächlich in Rechnung zu stellende Gebühr für eine Abrechnungseinheit. Hierbei ist der Parameter x *nicht* auf einen eindimensionalen Wertebereich festgelegt, sondern kann als Vektor

 Als Buch erhältlich im Verlag Dr. Hut, München, www.dr.hut-verlag.de (ISBN 3-934767-97-4)

$x = (x_1, x_2, \ldots, x_n)$ mehrere, im vorliegenden Fall n Wertebereiche repräsentieren. Das Spektrum eines durch den Parameter x_i abdeckbaren Wertebereichs ist sehr breit: Beispielsweise können mit einem x_i unterschiedliche Dienstgüteklassen, Tageszeitklassen, Nutzergruppenklassen, etc. ausgedrückt werden. Der Wertebereich hängt u.a. von der grundsätzlichen Unternehmenspolitik des Dienstleisters (ausgedrückt durch das *Dienstleistertarifschema*) als auch von den Wünschen des Kunden ab. Damit bildet eine Preisfunktion $p_i(x)$ einen u.U. mehrdimensionalen Parameter x auf einen (eindimensionalen) Preis in einer Währung ab. Die Gestaltung der Preisfunktion $p_i(x)$ ist von der zugeordneten Abrechnungseinheit A_i abhängig. Ausdrücklich muss an dieser Stelle betont werden, dass in aller Regel für eine einzelne Preisfunktion nicht alle Wertebereiche x_i relevant sind. Um *Vertragsstrafen* und *Rabatte* bezüglich einer Abrechnungseinheit A_i auszudrücken, kann der Wertebereich der Preisfunktion $p_i(x)$ auch ausdrücklich negativ gewählt werden. Damit wird insbesondere der auf Kundenseite entstehende wirtschaftliche Verlust durch eine nicht ausreichend zur Verfügung gestellte Funktionalitätseigenschaft explizit gemacht. Somit gilt es in diesem Teilprozess folgende Elemente zu vereinbaren:

a) Abrechnungseinheiten $A_1, \ldots, A_k$

b) Parameter $x = (x_1, x_2, \ldots, x_n)$ und damit die Wertebereiche der x_i (mit $i \in [1, n]$)

c) Preisfunktionen $p_0, ..., p_k$

d) das Abrechnungs(zeit)intervall, das festlegt, in welchem Zeitrhythmus die Abrechnung und damit die Rechnungsstellung durchzuführen ist

Da mit Werteänderungen von x insbesondere auch unterschiedliche Gebühren einhergehen, muss zusätzlich exakt festgelegt werden, welche Bedingungen erfüllt sein müssen, um derartige Werteänderungen zu ermöglichen. Dies ist insbesondere bei dynamischen, erst in der Betriebsphase feststellbaren Wertebelegungen relevant, wie z.B. eine bestimmte Dienstgüteklasse, Tageszeittarifklasse, etc., um spätere Missverständnisse und –interpretationen zu vermeiden.

Beispiel: Zur Erläuterung der Bestandteile eines Tarifmodells wird ein bewußt einfaches Beispiel herangezogen, welches sich an dem in Abschnitt 2.2 beschriebenen Szenario orientiert. Zwischen Dienstleister und Kunde wird zunächst unabhängig von der Dienstnutzung eine pauschale Gebühr von *xy* Euro pro Monat (entspricht dem Abrechnungsintervall) für den Betrieb und das Management des Dienstes vereinbart. Zudem wird eine nutzungsorientierte Tarifierung in Abhängigkeit von der Zugehörigkeit des Dienstnutzers zu einer *Nutzergruppe* vereinbart. Sofern der Dienstnutzer ein Mitarbeiter in der Konzernzentrale des Kunden ist, so soll nach entstandenem Volumen abgerechnet werden (*xx* Euro/MByte). Handelt es sich bei dem Dienstnutzer um einen durch das Extranet angeschlossenen Agenturmitarbeiter, so soll nach Dauer der Dienstnutzung abgerechnet werden (*yy* Euro/Stunde). Desweiteren gewährt der

Dienstleister zur Nutzungssteuerung einen 10% Rabatt, falls die Dienstnutzung außerhalb der üblichen Arbeitszeiten stattfindet. Um die Dienstgüte im Tarifmodell zu berücksichtigen, wird pauschal *zz* Euro Rabatt resp. Vertragsstrafe pro ausgefallener Stunde angerechnet, sofern der Ausfall des Dienstes länger als zwei Stunden dauert. Die nachfolgende Tabelle fasst die Tarifvereinbarungen nochmal übersichtlich zusammen:

Parameter x_i	**Wertebereich**	**Vereinbarung**
x_1: Nutzergruppe	Konzernzentrale Agentur	Volumen: *xx* Euro/MByte Dauer: *yy* Euro/Stunde
x_2: Zeitraum	übliche Arbeitszeit sonst	Standardtarif 10% Rabatt
x_3: Ausfall	Dienstausfall bis 2 Stunden Dienstausfall ab 2 Stunden	Kein Rabatt pauschal *zz* Euro/Ausfallstunde

Wie an der Tabelle ersehen werden kann, werden die Parameter, welche die Tarifwahl beeinflussen, als Parameter x_i zusammen mit den dazugehörigen Wertebereichen notiert. Ebenfalls ist ersichtlich, dass noch einige Ungenauigkeiten bestehen, die in der Dienstvereinbarung beseitigt werden müssen, um spätere Missverständnisse zwischen dem Kunden und Dienstleister zu vermeiden. Beispielsweise muss genau festgelegt werden, was „übliche Arbeitszeit“ bedeutet. Hier wäre eine Regelung „wochentags, 9 – 17 Uhr“ denkbar. Desweiteren muss geklärt werden, wie der Tarif für eine Dienstnutzung ausgewählt werden soll, die zwar vor 17 Uhr beginnt, aber erst nach 17 endet (denkbare Vereinbarungen: anteilig/immer der günstigere bzw. teurere).

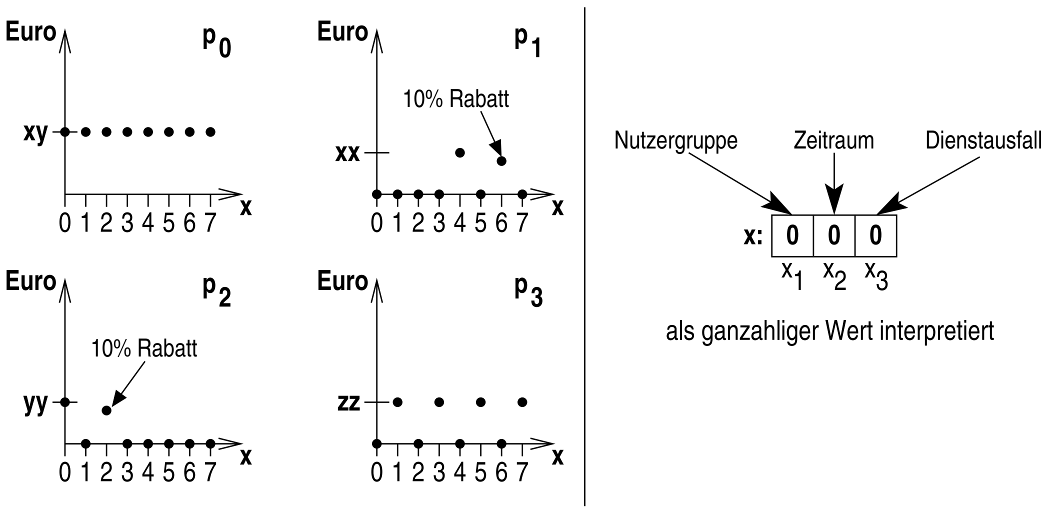

Abbildung 4.7: Preisfunktion

 Als Buch erhältlich im Verlag Dr. Hut, München, www.dr.hut-verlag.de (ISBN 3-934767-97-4)

In Abbildung 4.7 sind nun die sich aus den Vereinbarungen ergebenden Preisfunktionen dargestellt, die den jeweiligen Tarif in Abhängigkeit von x bestimmen. Hierbei setzt sich das x aus den einzelnen x_i wie folgt zusammen: Da die Wertebereiche der x_i binär sind, ergibt sich x durch das Zusammensetzen der einzelnen x_i zu einem Bitstring ($x = x_1x_2x_3$) und dem Anschließenden Interpretieren dieses Bitstrings als ganzzahliger Wert. Findet beispielsweise die erfolgreiche Dienstnutzung eines Agenturmitarbeiters ($x_1 = 1$) während der üblichen Arbeitszeit statt ($x_2 = 0$ und $x_3 = 0$, da offensichtlich erfolgreich), so ergibt das insgesamt: $x = 100_2 = 4_{10}$
Insgesamt ergibt sich damit das folgende Tarifmodell:
$G(x) = p_1(x)uebertrageneBytes + p_2(x)Dauer + p_3(x)Dienstausfallstunde + p_0(x)$

In Abbildung 4.6 ist der prinzipielle Ablauf des Teilprozesses der *Tarifierung* dargestellt. Zunächst werden Abrechnungseinheiten zwischen Kunde und Dienstleister vereinbart. Eine Abrechnungseinheit orientiert sich im nutzungsorientierten Fall immer an der zwischen Kunde und Dienstleister bereits vereinbarten *Dienstfunktionalität* und soll in aller Regel das Dienstnutzungsvolumen bzgl. einer Eigenschaft der Dienstfunktionalität widerspiegeln. Gemäß dem MNM Dienstmodell aus Abschnitt 2.1.1, in dem die Dienstfunktionalität in Nutzungs– und Managementfunktionalität aufgeteilt wird, bietet es sich der Übersichtlichkeit wegen an, die Abrechnungseinheiten ebenfalls nach diesem Schema aufzuteilen. Wichtig ist hierbei für den Dienstleister, dass die Abrechnungseinheiten mess– bzw. feststellbar sind, damit eine korrekte Rechnung ausgestellt werden kann. Im Dienstanalyse–Teilprozess (siehe Abschnitt 4.3.9) wird eine Identifizierung von Messeinheiten durchgeführt, die das Dienstnutzungsvolumen repräsentieren, so dass der dabei entstandene Messeinheitenkatalog als Grundlage für die Vereinbarung von Abrechnungseinheiten verwendet werden kann, falls es sich um die Neuverhandlung eines Tarifs handelt. Im einfachsten Fall wird demnach eine Teilmenge von (bekannten) Messeinheiten als Abrechnungseinheiten gewählt. Findet keine direkte Übernahme von Messeinheiten als Abrechnungseinheiten statt, so muss eine passende *Abbildungsvorschrift* definiert werden.

Für die Aktivität der Vereinbarung von Abrechnungseinheiten werden sowohl die Anforderungen des Kunden, die in Form des Kundentarifschemas vorliegen (siehe Abschnitt 4.3.2), als auch die Anforderungen der Dienstleister-Unternehmensleitung bzgl. des Tarifs in Form eines Dienstleistertarifschemas miteinbezogen. Das Dienstleistertarifschema enthält in aller Regel Unternehmenspolitiken bzgl. der Tarifgestaltung, die sich beispielsweise in durch Rabatte ausgedrückten Subventionen von bestimmten Diensten äußern können, um Marktanteile in einem bestimmten Sektor zu gewinnen. Allgemein werden durch ein Tarifschema sowohl (obligatorische) Bestandteile eines Tarifs resp. eines Tarifmodells, wie beispielsweise bestimmte Abrechnungseinheiten, als auch insbesondere das Verhältnis zwischen den einzelnen Tarifbestandteilen ausgedrückt. So kann beispielsweise allgemein festgelegt werden, dass die durch einen angewendeten Tarif erzeugte Gebühr zu 50 % durch eine bestimmte Abrechnungseinheit erbracht werden muss.

Nachdem eine Abrechnungseinheit festgelegt wurde, wird die dazugehörige *Preisfunktion* vereinbart. Die Preisfunktion legt die Gebühr für genau eine Abrechnungseinheit in Abhängigkeit von anderen Einflussfaktoren fest, die durch den Parameter x ausgedrückt werden. Diese Einflussfaktoren können grundsätzlich von beliebiger Art und Weise sein und sind somit explizit Gegenstand der Verhandlungen zwischen Kunde und Dienstleister. Das Spektrum der Einflussfaktoren reicht von Parametern wie der Dienstgüte, der Anzahl gleichzeitig zugreifender Nutzer, Ort der Dienstnutzung bis zu Parametern wie Nutzertyp. Grundsätzlich kann man in diesem Fall zwischen *statischen* Parametern und *dynamischen*, während der Dienstnutzung zu messenden Parametern unterscheiden. Im statischen Fall werden im Gegensatz zum dynamischen Fall demnach die Parameter nicht durch die Dienstnutzung beeinflusst und müssen folglich nicht während der Dienstnutzung festgestellt resp. gemessen werden. Als Beispiel ist die vereinbarte (entspricht statischem Parameter) im Gegensatz zur tatsächlich erfahrenen Dienstgüte (entspricht dynamischem Parameter) zu nennen. Wichtig ist hierbei, dass im Fall der dynamischen Parameter zusätzlich zu denen der Abrechnungseinheiten ebenfalls entsprechende Messkomponenten implementiert werden müssen.

Da insbesondere der Dienstleister an einem kostendeckenden Tarifmodell interessiert ist, wird die im vorhergehenden Teilprozess (siehe Abschnitt 4.3.3) erstellte Kostenprognose bei der Vereinbarung der Preisfunktion mit einbezogen. Hierbei ist aber zu beachten, dass im Vergleich zu einem Tarifmodell die Kostenprognose in der Regel immer eine Aufstellung der totalen Kosten enthält und Abhängigkeiten zwischen kostenverursachenden Komponenten außer Acht läßt. Im Gegensatz dazu ist aber eine Preisfunktion immer in Relation zu den übrigen Preisfunktionen eines Tarifmodells zu sehen, da die endgültige, in Rechnung zu stellende Gebühr durch die Summe der Einzelgebühren $p_i(x)A_i$ bestimmt wird und somit sowohl von der Wahl des Parameters x, der (zu erwartenden) Anzahl der Abrechnungseinheiten A_i als auch von $p_i(x)$ abhängig ist.

Das hierbei zwischen Dienstleister und Kunde vereinbarte Tarifmodell wird in die Dienstvereinbarung aufgenommen. Sofern ein nutzungsorientiertes Tarifmodell vereinbart wurde, das Abrechnungseinheiten enthält, für die noch keine Messkomponenten existieren, wird mit dem Teilprozess „Implementierung der Messkomponenten“ (siehe Abschnitt 4.3.5) fortgefahren. Ansonsten wird in Anschluss der Teilprozess „Data Rollout“ (siehe Abschnitt 4.3.6) ausgeführt.

4.3.5 Implementierung der Messkomponenten

Ziel des in Abbildung 4.9 visualisierten Teilprozesses, der in der Implementierungsphase des Dienstlebenszyklus stattfindet, ist es, Messkomponenten zu implementieren, die es ermöglichen, die Dienstnutzung in der Art und Weise zu messen, dass eine nutzungs– und volumenorientierte Abrechnung, wie sie durch das Tarifmodell im vorhergehendem Teilprozess festgelegt und vereinbart wurde, ermöglicht wird. Dieser Teilprozess ist folglich *nicht aktiv*, falls ein pauschales bzw. nicht an der Dienstnutzung orientiertes Tarifmodell vereinbart wurde.

 Als Buch erhältlich im Verlag Dr. Hut, München, www.dr.hut-verlag.de (ISBN 3-934767-97-4)

Für die Implementierung einer Messkomponente ist es zunächst notwendig, deren grundsätzliche Architektur festzulegen, um die zu implementierenden Bestandteile identifizieren zu können. Im Folgenden wird in Anlehnung an die IPDR Architektur (siehe Abschnitt 3.2.6) prinzipiell zwischen einem *Sensor*, der Messdaten über die auftretende Dienstnutzung bereit stellt, und einem *Messagenten* unterschieden, der die vom Sensor ermittelten Messdaten vorverarbeitet, um sie an einer wohldefinierten Schnittstelle an weiterverarbeitende Anwendungen, wie z.B. Kollektoren, zur Verfügung zu stellen. Damit überwindet der Messagent für die weiterverarbeitenden Anwendungen die Heterogenität der Sensoren. Sensor und Messagent bilden zusammen eine *Messkomponente* (siehe hierzu Abbildung 4.8). Sowohl Sensor als auch Messagent können als eigenständige Anwendung realisiert sein, so dass ein Messagent auch mit mehreren Sensoren assoziiert werden kann. Jedoch muss sinnvollerweise jedem Sensor eindeutig ein Messagent zugeordnet werden, da ansonsten die gemessenen Daten nicht an weiterverarbeitende Anwendungen weitergegeben werden können. Im Folgenden wird konzeptionell jeder Sensor mit genau einer Messfunktion assoziiert. Insbesondere ist für die nutzungsorientierte Abrechnung der sog. *Messeinheitenzähler* als Sensor relevant, der die Dienstnutzung in diskret abzählbaren Einheiten misst. Desweiteren können Sensoren mit weiteren Messfunktionen realisiert sein, die z.B. die erfahrene Dienstgüte während der Dienstnutzung feststellen. Da insbesondere der Messeinheitenzähler im Gegensatz zu anderen Sensoren dem Abrechnungsmanagement zugeordnet wird, bezieht sich die nachfolgende Darstellung v.a. auf die Betrachtung dieser Art von Sensoren.

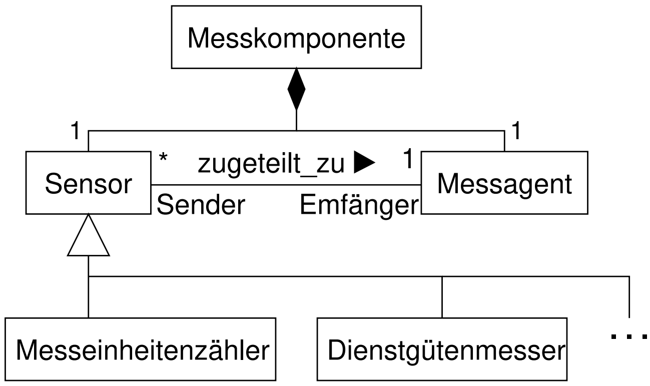

Abbildung 4.8: Bestandteile einer Messkomponente

Um die nutzungsorientierte Abrechnung zu ermöglichen, müssen demnach, falls nicht existent, geeignete Sensoren sowie Messagenten implementiert werden, die den Zugriff auf die Dienstfunktionalität in der Art und Weise überwachen, dass die im Tarifmodell vereinbarten Abrechnungseinheiten gezählt werden können. Hierfür ist eine Analyse der Dienstrealisierung hinsichtlich der zu überwachenden, am Dienstzugriff beteiligten Komponenten notwendig, um festzustellen, für welchen Komponententyp ein Sensor implementiert werden muss. Dieser Vorgang gestaltet sich z.T. schwierig, da der Dienst an sich in dieser Phase noch nicht installiert ist und somit die tatsächliche Dienstrealisierung, bezogen beispielsweise auf Verteilung, Redundanz und Laufzeitumgebung der einzelnen Dienstkomponenten, zu diesem Zeitpunkt noch nicht feststehen muss. Dies ist insbesondere bei Dienstkomponenten, die sich nach der Installationsphase außerhalb der dienstleister–eigenen Zuständigkeitsdomäne befinden, kritisch, da sicher gestellt werden muss, dass auf die zu überwachenden Komponenten während der tatsächlichen Messung in der Betriebsphase zugegriffen werden kann. Somit müssen für den in Abbildung 4.9 dargestellten Teilprozess der Implementierung einer Messkomponente die im Nachfolgenden

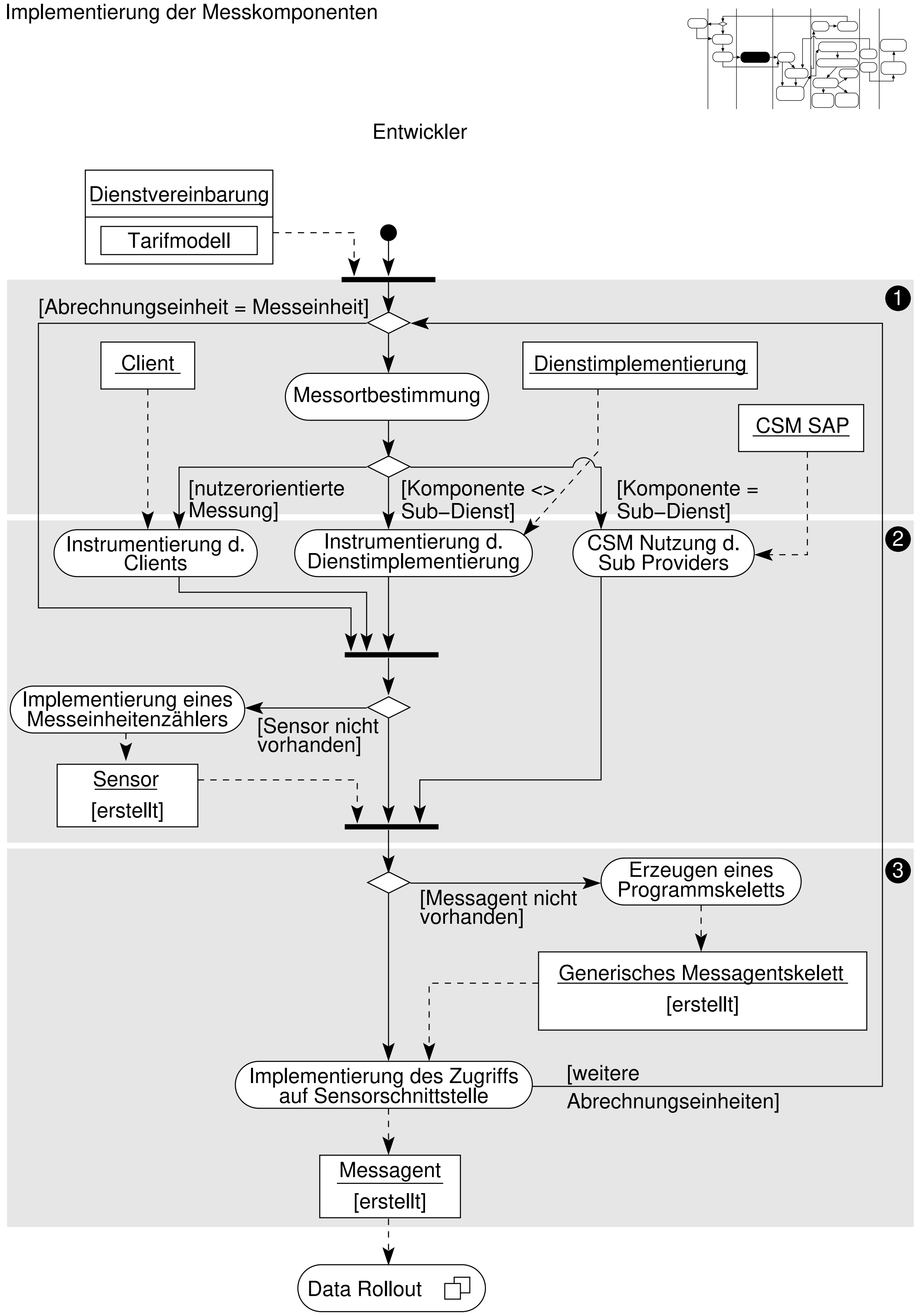

Abbildung 4.9: Implementierung der Messkomponenten

 Als Buch erhältlich im Verlag Dr. Hut, München, www.dr.hut-verlag.de (ISBN 3-934767-97-4)

beschriebenen drei Schritte durchgeführt werden (analog zur Abbildung 4.9 von ❶ – ❸ durchnummeriert).

❶ Identifizierung des zu überwachenden Komponententyps Da die zu überwachenden Komponenten letztlich durch die im vereinbarten Tarifmodell festgelegten Abrechnungseinheiten bestimmt werden, wird jede im Tarifmodell enthaltene Abrechnungseinheit auf den dazugehörigen, zu überwachenden Komponententyp hin untersucht. Unter Komponententyp wird eine ganze Klasse von gleichartigen Komponenten verstanden, die durch Einsatz eines gleichartigen Sensors überwacht werden können.

Beispiel: Sei zur Feststellung des Dienstnutzungsvolumens eines bereitgestellten Dienstes die Überwachung der jeweiligen Web Browser der User notwendig. Um die Überwachung zu ermöglichen, ist jeweils nur ein Sensor pro unterschiedlichem Browsertyp (also beispielsweise pro unterschiedlicher Browserversion eines Herstellers) zu implementieren.

Um die Komponententypen zu identifizieren, muss zunächst festgestellt werden, welche Komponenteninstanzen zu überwachen sind. Hierfür wird der *Messort* unter Zuhilfenahme des vollständigen Tarifs, d.h. aus der Kombination Preisfunktion und dazugehörige Abrechnungseinheit, ermittelt. Als Hilfestellung wird analog zu [HaRa 00, HaRa 01] die folgende Klassifikation der Messorte vorgeschlagen: Es wird eine *horizontale* und *vertikale Einteilung* der Messorte vorgenommen. Die horizontale Einteilung teilt den Messort in Dienstnehmer– und Dienstleisterseite ein. Die vertikale Einteilung sieht eine Messung auf Netz–, System– und Anwendungsebene vor. Im Prinzip kann aus der zu einer konkreten Abrechnungseinheit gehörenden Preisfunktion die horizontale Zuordnung des Messorts abgeleitet werden, je nachdem ob sich der Messort auf der Dienstnehmer– oder Dienstleisterseite befindet.

Beispiel: Sieht die Preisfunktion eines vereinbarten Tarifs eine dienstgüteabhängige Gebührenberechnung vor, wobei die Dienstgüte aus Nutzersicht gemessen werden soll, so kann die Messung nur auf Dienstnehmerseite stattfinden.

Um derartige Interpretationen des Messorts durchzuführen, ist eine semantische Auswertung des Wertebereichs des Parameters x der Preisfunktion notwendig. Im Gegensatz dazu bestimmt die vereinbarte Abrechnungseinheit in Kombination mit der Dienstimplementierung die vertikale Einteilung des auszuwählenden Messorts, je nachdem ob die Messung auf Netz–, System– oder Anwendungsebene durchgeführt werden muss.

Beispiel: Sei als Abrechnungseinheit eines vereinbarten Tarifs „Anzahl der Transaktionen" gewählt worden. Die Dienstrealisierung sieht eine verschlüsselte Kommunikation zwischen Client und Server vor. Aus diesem Grund sind weder durch Protokollanalyse noch durch Auswertung von Systemparametern die benötigten Daten erfassbar. Konsequenterweise muss auf Anwendungsebene gemessen werden, um die notwendigen Daten zu erfassen.

In [HaRa 01] wird gezeigt, dass der Aufwand zur Implementierung eines Sensors von der Netz– bis zur Anwendungsebene und von Dienstleister– zu Dienstnehmerseite steigt, so dass entsprechend dieser Reihenfolge der passende Messort gesucht werden sollte. In Tabelle 4.1 ist eine Übersicht über diese Aufteilung dargestellt, die zusätzlich eine Mindestabschätzung der zu überwachenden Komponententypen enthält.

vertikale Einteilung bestimmt durch Abrechnungseinheit	*horizontale Einteilung* bestimmt durch Preisfunktion	
	Dienstnehmerseite	**Dienstleisterseite**
Netzebene	pro angeschlossenem Subnetz eine Messkomponente	pro verbindendem Edge–Router eine Messkomponente
Systemebene	pro Client–Endsystem eine Messkomponente	pro Server–Endsystem eine Messkomponente
Anwendungsebene	pro Anwendungsclientinstanz ein Sensor	pro Anwendungsserverinstanz ein Sensor

Tabelle 4.1: Bestimmung des Messorts für die Implementierung von Messkomponenten

Zudem kann als weiterer Fall, der nicht explizit in Tabelle 4.1 mitaufgenommen wurde, auftreten, dass die Messung zwar auf Dienstleisterseite stattfinden soll, jedoch sich die zu überwachende Komponente außerhalb der dienstleister–eigenen Zuständigkeitsdomäne befindet. In diesem Fall wird die zu überwachende Teilfunktionalität der Dienstrealisierung durch Nutzung eines (Sub–)Dienstes eines Sub–Dienstleisters erbracht. Um dennoch die notwendigen Daten erfassen zu können, muss in diesem Fall mit dem Sub–Dienstleister vertraglich vereinbart werden, dass die benötigten Daten über die Customer Service Management (CSM) Schnittstelle zwischen Dienstleister und Sub–Dienstleister zur Verfügung gestellt werden.

Im günstigsten Fall stehen die zu überwachenden Bestandteile allerdings bereits fest und eine eigenständige Analyse ist nicht notwendig. Fand nämlich eine Neuverhandlung des Tarifmodells statt, so dass der Tarifierungsprozess (siehe Abschnitt 4.3.4) sich auf die Ergebnisse des Dienstanalyseprozesses (siehe Abschnitt 4.3.9) gründet, so sind die Abrechnungseinheiten des dabei vereinbarten Tarifs i.d.R. eine echte Teilmenge der Messeinheiten aus dem Dienstanalyseprozess und damit ist der zu einer Messeinheit respektive Abrechnungseinheit gehörende, zu überwachende Dienstbestandteil bereits identifiziert worden.

 Als Buch erhältlich im Verlag Dr. Hut, München, www.dr.hut-verlag.de (ISBN 3-934767-97-4)

❷ **Implementierung des Sensors** Nachdem im vorhergehenden Schritt der zu überwachende Komponententyp identifiziert wurde (hierzu gibt ebenfalls Tabelle 4.1 eine Übersicht, *was* in den einzelnen Fällen zu überwachen wäre), muss dafür ein geeigneter Sensor implementiert werden. Im Folgenden wird konzeptionell im Vergleich zum Messagenten angenommen, dass der Sensor *keinerlei* Vorverarbeitung im Sinne der weiterverarbeitenden Anwendungen des Abrechnungsprozesses durchführt. Jegliche Vor– respektive Weiterverarbeitung wird durch den zugehörigen Messagenten verwirklicht. Die explizite Implementierung eines Sensors ist allerdings nur dann notwendig, wenn die benötigten Daten nicht schon anderweitig zur Verfügung gestellt werden (wie z.B. durch SNMP MIBs auf Koppelementen oder Endsystemen, etc.). Insbesondere entfällt demnach eine Sensorimplementierung, wenn die benötigten Daten von einer CSM–Schnittstelle zur Verfügung gestellt werden.

Für die drei im vorhergehenden Abschnitt eingeführten Ebenen kommen nach [HaRa 01] die folgenden drei Messverfahren in Frage, die vom Sensor geeignet implementiert werden müssen:

1. *Protokollanalyse (Netzebene)*
   Grundsätzliches Vorgehen hierbei ist das vollständige Aufzeichnen des Netzverkehrs („Scannen") durch den Sensor. Hierbei muss am Netzinterface die vollständige Datenübertragung mitprotokolliert werden.
2. *Überwachung von Systemparametern (Systemebene)*
   Dieses Verfahren basiert auf der Überwachung von systemorientierten Parametern, wie beispielsweise die CPU Last, Anzahl der Dateizugriffe, Anzahl geöffneter Dateien, Anzahl offener Verbindungen etc. Üblicherweise kann diese Information durch Verwendung von (betriebs–)systeminternen APIs abgefragt werden.
3. *Anwendungsüberwachung (Anwendungsebene)*
   In diesem Fall erfolgt eine direkte Überwachung der Anwendung, d.h. es werden Informationen über den gegenwärtigen Zustand der Anwendung direkt aus der überwachten Anwendung geliefert. Hierzu ist eine *Anwendungsinstrumentierung* notwendig, d.h. dass managementrelevanter Programmcode in den eigentlichen Quellcode der Anwendung eingefügt wird. Hierbei liegt die Schwierigkeit v.a. im Auffinden der richtigen Stelle im Quellcode, in welcher der Managementcode eingefügt werden soll. Zudem muss für die erfolgreiche Anwendungsinstrumentierung auch der Zugriff auf den Quellcode der zu instrumentierenden Anwendung ermöglicht werden, so dass Legacy–Anwendungen im Nachhinein i.d.R. nicht instrumentiert werden können. In [Hauc 01] wird allerdings ein Ansatz für bausteinbasierte Anwendungen beschrieben, mit dem einerseits Legacy–Komponenten eingebunden werden können und andererseits die Instrumentierung fast vollständig automatisiert abläuft, so dass der Aufwand wesentlich reduziert wird.

Der zu implementierende Sensor stellt demnach für den Messagenten eine Schnittstelle zur Verfügung, an der Messdaten abgefragt werden können, die ein Ableiten der im Tarif vereinbarten Abrechnungseinheiten ermöglichen. Die Art und Weise der Realisierung des Sensors bestimmt insbesondere die Interaktion mit dem zugewiesenen Messagenten: Verfügt der

Sensor beispielsweise nur über flüchtigen Speicher, so muss darauf geachtet werden, dass die Messdaten rechtzeitig vom Messagenten abgefragt werden, um Datenverluste durch z.B. Speicherüberläufe zu vermeiden. Je nach der zu überwachenden tatsächlichen Komponente, den zu messenden Daten und der damit anfallenden Datenmenge, ist vom *Entwickler* somit zu entscheiden, ob die Schnittstelle zwischen Sensor und Messagent als interne oder externe Schnittstelle realisiert wird und welches Modell zur Messdatenübertragung eingesetzt wird (synchron/asynchron, Push/Pull/Hybrid).

❸ Implementierung des Messagenten Aufgabe des Messagenten ist es, die von einem Sensor an dessen Schnittstelle zur Verfügung gestellten Messdaten abzufragen, unter Umständen eine gewisse Vorverarbeitung, wie z.B. Ableiten von Abrechnungseinheiten, durchzuführen und schließlich diese Daten in Form von *Meter Records* an der eigenen Messagentenschnittstelle zur Verfügung zu stellen. Der Zugriff auf die Messdaten kann prinzipiell auf drei verschiedene Arten geschehen: Nutzung einer proprietären Sensorschnittstelle, Analyse von Logdateien und Nutzung standardisierter Schnittstellen, wie z.B. die Abfrage von Messdaten per SNMP. Im Folgenden ist ein Überblick über standardisierte Zugriffsmethoden zur Gewinnung von Messdaten in Bezug zu den drei Ebenen dargestellt:

1. *Netzebene*
   Neben dem Zugriff auf proprietäre Sensorschnittstellen von Koppelelementherstellern, wie z.B. Cisco Netflow, besteht im Allgemeinen auch die Möglichkeit, Messdaten in Form von SNMP MIBs, wie z.B. die *RMON MIB* [RFC 2819], zu gewinnen. Hierbei können entweder die Abrechnungseinheiten direkt (z.B. durch Zählen der übertragenen Nachrichteneinheiten wie IP Pakete) oder indirekt gewonnen werden. Im indirekten Fall wird durch semantische Analyse des überwachten Netzverkehrs versucht, auf andere Abrechnungseinheiten, wie z.B. Anwendungstransaktionen, zu schließen. Hierbei ist jedoch eine detaillierte Kenntnis des eingesetzten Anwendungsprotokolls notwendig.

2. *Systemebene*
   Auch in diesem Fall besteht die Möglichkeit, dass die Daten in Form von SNMP MIBs, wie beispielsweise die *Host Resources MIB* [RFC 1514], die *Network Services Monitoring MIB* [RFC 1565] und die *System–Level Application MIB* [RFC 2287], zur Verfügung gestellt werden, sofern das System diese unterstützt. Alternativ werden von einigen Betriebssystemen durch eigenständige Anwendungen, wie z.B. das Unix–Tool *acct*[3], auch Logfiles geführt, in denen relevante Messdaten herausgelesen werden können.

3. *Anwendungsebene*
   Die für einen Messagenten gängigsten Verfahren, um auf Anwendungsebene auf die notwendigen Messdaten zugreifen zu können, sind die Analyse von durch Anwendungen selbst angelegte Logfiles, der Zugriff auf SNMP MIBs sowie die direkte Abfrage über (standardisierte) APIs wie die *Application Response Measurement API (ARM)* [C807,

[3] In diesem Fall würde dieses Tool einem Sensor entsprechen.

 Als Buch erhältlich im Verlag Dr. Hut, München, www.dr.hut-verlag.de (ISBN 3-934767-97-4)

> C014], die sich auf die Methode der Anwendungsinstrumentierung gründet. Die bisher im Bereich Anwendungsmanagement standardisierten SNMP MIBs, wie die *Application Management MIB* [RFC 2564] und die *Application Performance Measurement MIB* [Wald 02], die eine Überwachung von anwendungsorientierten Parametern wie Transaktionen vorsehen, setzen allerdings immer eine Anwendungsinstrumentierung voraus und bieten somit nur eine weitere Möglichkeit, managementrelevante Daten abzufragen.

Die Hauptfunktionalität eines Messagenten ist demnach, die von einem Sensor zur Verfügung gestellten Messdaten verlustfrei einzusammeln, zu Meter Records geeignet zusammenzufassen und an einer wohldefinierten Schnittstelle weiterverabeitenden Anwendungen, wie z.B. Kollektorsoftware, zur Verfügung zu stellen. Da die Zugriffsschnittstelle gegenüber den weiterverarbeitenden Anwendungen einheitlich sein sollte, kann für die Implementierung des Messagenten ein *Messagentskelett* verwendet werden, in der die Signatur der Schnittstelle bereits enthalten ist. Damit muss vom Entwickler lediglich die dazugehörige, der Sensorschnittstelle und den Messdaten angepasste Funktionalität implementiert werden.

Nach Beendigung des letzten Schritts dieses Teilprozesses wird mit dem Teilprozess „Data Rollout", der im nachfolgenden Abschnitt beschrieben wird, fortgefahren.

4.3.6 Data Rollout

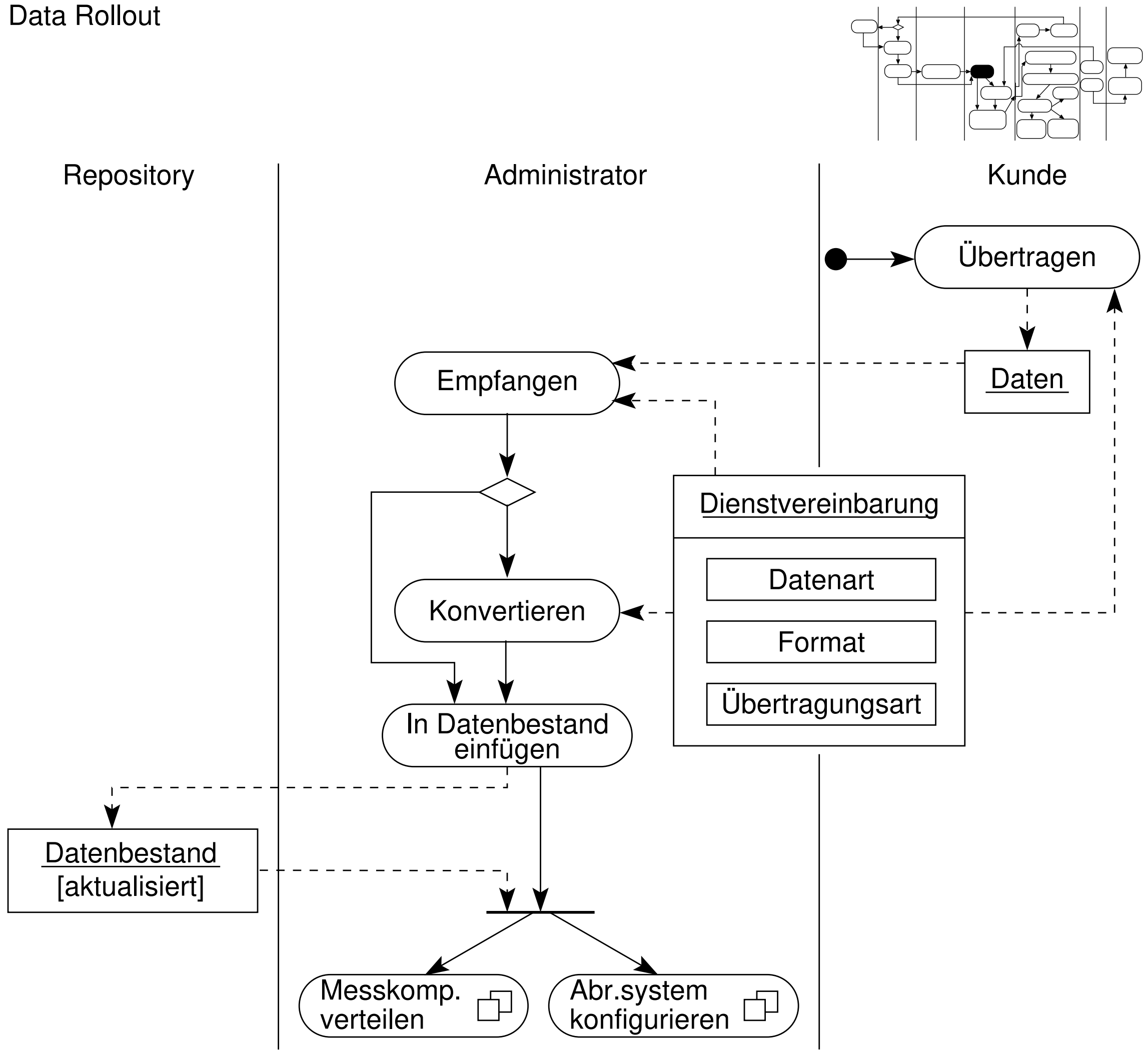

Abbildung 4.10: Data Rollout

Ziel des in Abbildung 4.10 dargestellten und in der Installations– und Testphase des Dienstlebenszyklus stattfindenden Teilprozesses „Data Rollout“ ist es, alle Daten, die auf Dienstleisterseite zum erfolgreichen Betrieb der Abrechnung benötigt werden und ohne direkten Zugriff beim Kunden anfallen, an den *Administrator* auf der Dienstleisterseite zu übergeben. Es handelt sich folglich um Daten, die außerhalb des Zuständigkeitsbereichs des Dienstleisters anfallen und damit explizit an diesen übergeben werden müssen. Bei diesen Daten handelt es sich *nicht* um Messdaten, wie sie bei der Dienstnutzungserfassung (siehe Abschnitt 4.3.11) durch die Messkomponenten anfallen. Die beim vorliegenden Teilprozess „Data Rollout“ ausgetauschten Daten sind zwar notwendig, um eine (korrekte) Abrechnung zu ermöglichen (z.B. Adresse

 Als Buch erhältlich im Verlag Dr. Hut, München, www.dr.hut-verlag.de (ISBN 3-934767-97-4)

der Rechnungsempfänger für die Rechnungszustellung, User Accounts für die Zuordnung von Messdaten, etc.), haben aber mit dem eigentlichen Dienstnutzungsvorgang nichts gemeinsam.

Zunächst werden hierfür die benötigten Daten vom Kunden aufbereitet. Sowohl die Art der Daten, d.h. welche Information vom Kunden zur Verfügung gestellt und vom Dienstleister zum erfolgreichen Betrieb benötigt werden, als auch das Format, in dem die Daten vorliegen, müssen in der Dienstvereinbarung festgelegt sein. Die Festlegung des Datenformats ist insbesondere dann relevant, wenn die vom Kunden gelieferten Daten nicht in einem standardisierten, vom Administrator unterstützten Format vorliegen. Dies ist momentan der weitaus üblichste Fall, da die vom Kunden in diesem Fall verwendeten Datenquellen, wie Bestandsführungssysteme, betriebswirtschaftliche Software, etc. meist nur proprietäre Datenformate unterstützen. Wie bereits kurz angedeutet, können die vom Kunden gelieferten Informationen vielfältiger Natur sein: Hierbei kann es sich um Koppelelementidentifikatoren, Endsystemadressen, User Accounts, Adressen der Rechnungsempfänger, etc. handeln. Dies ist vom vereinbarten Dienst und insbesondere vom vereinbarten Tarifmodell abhängig und kann nicht allgemein angegeben werden. Nachdem der Administrator die Daten per vereinbarter Übertragungsart empfangen hat, müssen diese eventuell erst in ein Format konvertiert werden, welche die zu aktualisierenden Systeme und Anwendungen unterstützen. Um eine geeignete Abbildungsvorschrift zu definieren, wird hierzu das in der Dienstvereinbarung festgelegte Format herangezogen. Nach einer eventuell durchgeführten Konvertierung, werden die Daten in den eigenen Datenbestand eingefügt bzw. bestehende aktualisiert.

In Anschluss an diesen Teilprozess werden entweder die Messkomponenten verteilt (siehe Abschnitt 4.3.7) oder, falls dies nicht notwendig ist, direkt das Abrechnungsystem konfiguriert (siehe Abschnitt 4.3.8).

4.3.7 Verteilung der Messkomponenten

In Abbildung 4.11 ist der Teilprozess der Verteilung der Messkomponenten dargestellt, der in der Installations– und Testphase des Dienstlebenszyklus stattfindet. Ziel dieses Teilprozesses ist es, ausreichend viele Messkomponenten zu instantiieren und zu verteilen, so dass alle für die Nutzungserfassung relevanten Komponenten überwacht werden. Die zu überwachenden Komponenten werden durch das in der Dienstvereinbarung ausgehandelte Tarifmodell bestimmt und werden bereits im Teilprozess der Implementierung der Messkomponenten identifiziert, der in Abschnitt 4.3.5 beschrieben ist. Folglich ist dieser Teilprozess nicht aktiv, falls ein pauschales bzw. ein nicht an der Dienstnutzung orientiertes Tarifmodell vereinbart wurde. Im Folgenden werden die einzelnen durchzuführenden Schritte näher erklärt.

Wie in Abbildung 4.11 dargestellt, muss pro zu überwachender Komponente eine Messkomponente instantiiert werden. Hierzu wird zunächst ein dem Komponententyp entsprechender Sensor, der die vereinbarten Abrechnungseinheiten zählt, instantiiert und der zu überwachenden Komponente zugewiesen. Anschließend wird ein den Sensorcharakteristika (bezogen auf Zugriffsschnittstelle, Abfrageintervall, etc.) entsprechender Messagent instantiiert und dem Sensor

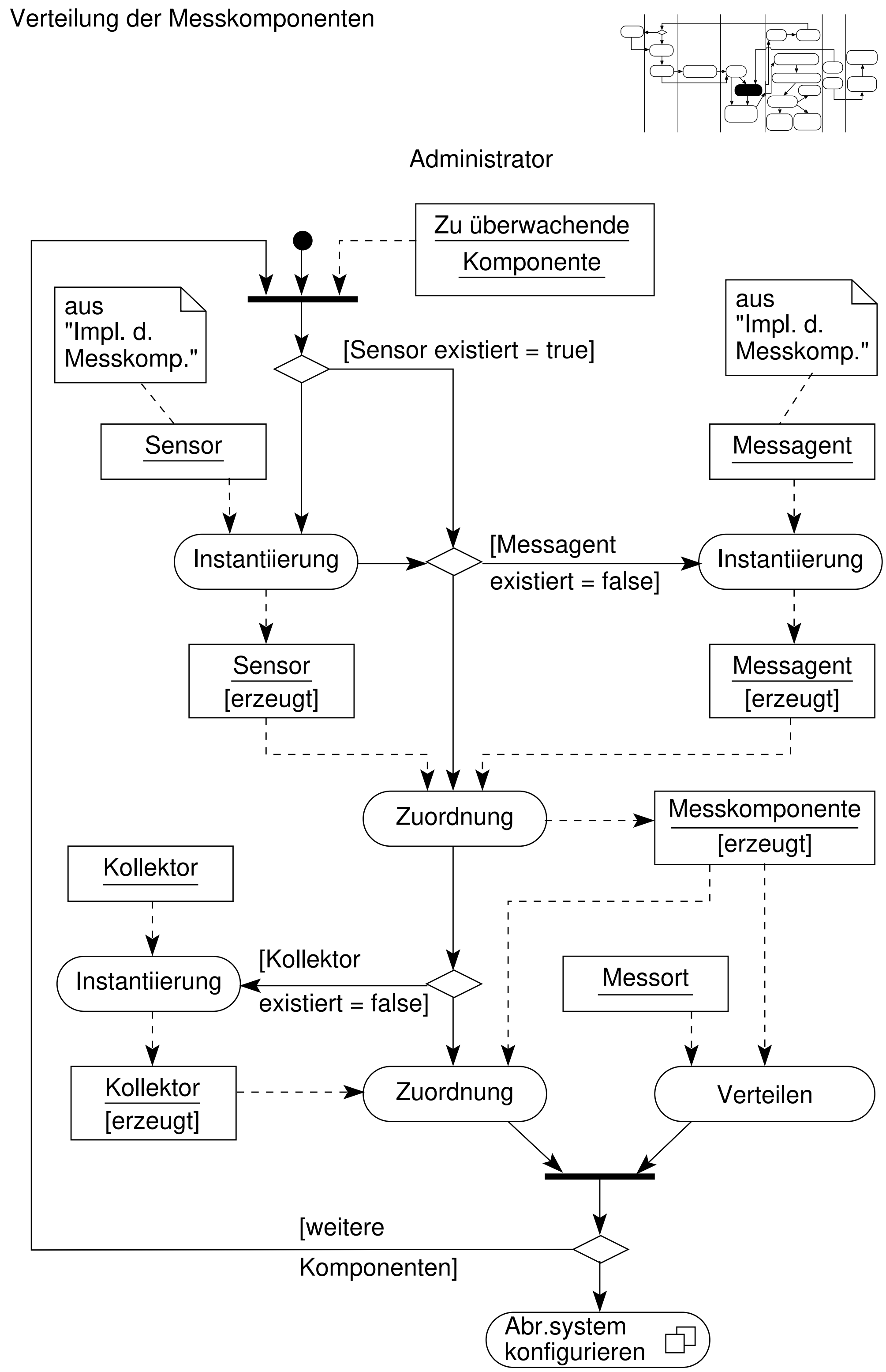

Abbildung 4.11: Verteilung der Messkomponenten

zugeordnet. Mit der Zuordnung von Sensor zu Messagent ist konzeptionell eine Messkomponente erstellt worden, die noch entsprechend dem Messort der zu überwachenden Komponente verteilt werden muss. Desweiteren wird die Messkomponente schließlich einem Kollektor zugeordnet, der in regelmäßigem Abstand die von der Messkomponente erzeugten Meter Records einsammelt und auf Basis dieser Daten ein Usage Record erzeugt (der genaue Ablauf der Nutzungserfassung und damit die Aufgaben der einzelnen daran beteiligten Komponenten wird in Abschnitt 4.3.11 beschrieben).

In Anschluss an diesen Teilprozess folgt die Konfiguration des Abrechnungssystems, das nachfolgend beschrieben wird.

4.3.8 Konfiguration des Abrechnungssystems

Ziel des in Abbildung 4.12 dargestellten Teilprozesses, der in der Installations– und Testphase des Dienstlebenszyklus stattfindet, ist es, alle an der Abrechnung eines Dienstes beteiligten Komponenten in der Art und Weise zu konfigurieren, dass die Abrechnung erfolgreich, d.h. entsprechend der zwischen Diensterbringer und Kunde geschlossenen Dienstvereinbarung, durchgeführt werden kann. Im Folgenden werden die einzelnen, durchzuführenden Konfigurationsschritte, jeweils nach den zu konfigurierenden Komponenten gegliedert, erklärt.

❶ Messkomponente Die Konfiguration einer Messkomponente, bestehend aus einem oder mehreren Sensoren und genau einem Messagenten, erfordert Festlegungen bzgl. folgender Elemente:

- *Felder der zu generierenden Meter Records*
  Der Messagent einer Messkomponente stellt weiterverarbeitenden Anwendungen, wie z.B. Kollektoren, sog. *Meter Records* zur Verfügung. Meter Records enthalten geeignet aufbereitet Messdaten, die von den zugeordneten Sensoren geliefert werden. Ein Eintrag innerhalb des Meter Records entspricht einem Messdatum. Damit weiterverarbeitende Anwendungen die Meter Records und insbesondere die darin enthaltenen Einträge analysieren können, müssen die Felder eines Eintrags sowie deren Semantik (z.B. entspricht der Eintrag einem aggregierten Wert oder einem tatsächlichem Messwert, entspricht der Zeitstempel dem Zeitpunkt des Einsammelns oder der Messung, etc.) bestimmt werden. Pro Feld eines Eintrags muss die Datenquelle, also z.B. der Sensor festgelegt werden, von welcher der Feldinhalt zur Verfügung gestellt wird. Desweiteren können vorverarbeitende Aggregationsfunktionen ausgewählt werden, Zeitstempelfelder spezifiziert werden, etc.
- *Zeitpunkt der ersten Messung und Zeitabstand zur Generierung der Meter Records*
  Damit wird das Zeitintervall zwischen den Zeitpunkten angegeben, zu denen die Messdaten durch den Messagenten eingesammelt und die dazugehörigen Meter Records erstellt werden.
- *Aufbewahrungszeit*
  Um ein nachträgliches Überprüfen sowie Reproduzieren von durch andere Komponenten erstellten Dokumenten wie Usage Records, Customer Detailed Records und Rechnungen

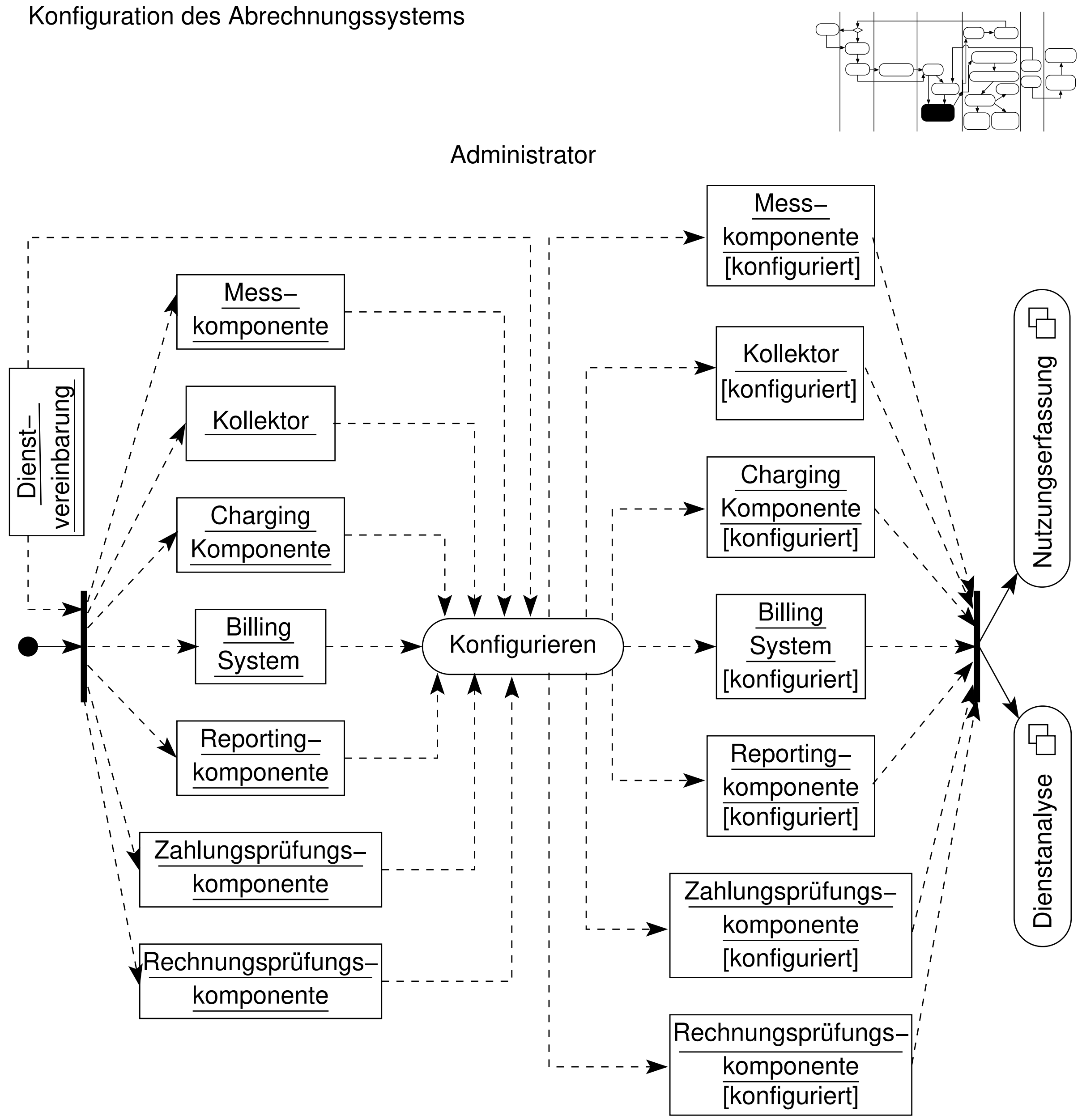

Abbildung 4.12: Konfiguration des Abrechnungssystems

zu ermöglichen, müssen die erstellten Meter Records mindestens die angegebene Zeit aufbewahrt werden. Dies wird meist in der Dienstvereinbarung zwischen Dienstleister und Kunde festgelegt.

- *Einsammelmodell*
  In der Regel werden Meter Records per Pull–Modell von weiterverarbeitenden Anwendungen, wie z.B. Reporting–Tools oder Kollektoren, angefordert. Um die Robustheit ge-

 Als Buch erhältlich im Verlag Dr. Hut, München, www.dr.hut-verlag.de (ISBN 3-934767-97-4)

genüber Verlust von generierten Meter Records zu erhöhen, kann es allerdings sinnvoll sein, ein Push– bzw. ereignis–basiertes Pull–Modell zu realisieren. Dies kann insbesondere im Falle von großen Datenmengen zweckmäßig sein, wenn Datenverlust aufgrund von begrenztem Speicher droht. Wird das Push– bzw. ereignis–basierte Pull–Modell gewählt, so müssen ebenfalls die Empfänger der Meter Records respektive der Ereignisse spezifiziert werden.

- *Zugewiesene, weiterverarbeitende Anwendungen*
  Da unter Umständen sicherheitssensible und vertrauliche Daten durch die Messkomponente gemessen und zusammengestellt werden, soll der Zugriff auf Meter Records nur für autorisierte Anwendungen ermöglicht werden. In diesem Fall muss ein entsprechend eindeutiger Identifikator aller autorisierten Komponenten konfiguriert werden.

❷ **Kollektor** Eine Festlegung bezüglich der folgenden Elemente ist für die erfolgreiche Konfiguration eines Kollektors erforderlich:

- *Identifikatoren der zugeordneten Messkomponenten*
  Da in der Regel das Pull–Modell verwendet wird, um Meter Records von Messkomponenten abzufragen, werden die eindeutigen Identifikatoren aller Messkomponenten benötigt, die dem Kollektor zugeordnet sind. Die Identifikatoren müssen in der Art und Weise beschaffen sein, dass damit eine Adressierung der Messkomponenten möglich ist. Im Regelfall ist die Zuordnung von Messkomponenten zu einem Kollektor bereits während der Verteilung der Messkomponenten (siehe Abschnitt 4.3.7) geschehen.
- *Einsammelintervall*
  Mit diesem Parameter wird der Zeitpunkt des ersten Einsammelns der Meter Records als auch die Zeitintervalle zwischen den Einsammelzeitpunkten festgelegt. Die Generierung der Usage Records wird nach Abschluss des Sammelvorgangs angestoßen.
- *Einsammelmodell*
  Wie bereits unter ❶ erklärt, können die Meter Records auch abweichend vom Pull–Modell im Push– oder im Event–basierten Pull–Modell von den Messkomponenten eingesammelt werden. Das gewählte Einsammelmodell wird bezüglich jeder Messkomponente festgesetzt, sofern dieses vom Pull–Modell abweicht.
- *Felder eines zu generierenden Usage Records*
  Ähnlich wie bei den Meter Records müssen bei den zu generierenden Usage Records die Felder der einzelnen Einträge festgelegt werden, die dann durch Aggregation verschiedener Meter Records gefüllt werden. Obligatorische Felder sind: Accounting User Identifikator und das in Anspruch genommene Dienstnutzungsvolumen. Ein Usage Record bezieht sich immer auf genau einen Dienst und genau einen Zeitraum.
- *Interpretation der Meter Records*
  Die zentrale Aufgabe eines Kollektors ist die Generierung von Usage Records auf Basis der eingesammelten Meter Records (siehe hierzu auch Abschnitt 4.3.11). Diese Aktivität hängt unmittelbar mit einer Aggregation der in Meter Records enthaltenden Daten zusammen. Um dies allerdings durchführen zu können, ist eine Analyse der Meter Records

notwendig und damit die Kenntnis sowohl der Struktur der Einträge als auch der Semantik der einzelnen Felder. Beispielsweise muss dem Kollektor bekannt gegeben werden, welche Felder den Nutzer, das Nutzungsvolumen oder den Zeitstempel repräsentieren. Ebenfalls muss beispielsweise festgelegt werden, in welcher Einheit das Nutzungsvolumen, der Zeitstempel, etc. vorliegt und wie eine gegebene Aggregationsfunktion darauf angewendet werden kann. Im Prinzip legt man in diesem Konfigurationsschritt die Abbildung zwischen den Eingabedaten, die als Meter Records vorliegen, und den Ausgabedaten, die durch die zu generierenden Usage Records vorliegen, fest.

- *Dienstidentifikator für die Haltung von Nutzer– und Kundendaten*
  Die zu generierenden Usage Records enthalten das durch einen Nutzer in Anspruch genommene, (aggregierte) Dienstnutzungsvolumen in einem Zeitintervall. Hierbei wird innerhalb eines Usage Records das Dienstnutzungsvolumen jeweils nach eindeutigen *Accounting User IDs* aufgeschlüsselt. Die Eindeutigkeit der Accounting User IDs bezieht sich jeweils auf eine Abrechnungs– und Rechnungsstellungsdomäne. Um die Abbildung der in einem Meter Record verwendeten User IDs[4] auf die Accounting User IDs zu ermöglichen, muss dem Kollektor der Dienst angegeben werden, der diese Abbildung realisiert. In der Regel handelt es sich hierbei um eine Datenbank bzw. um einen Verzeichnisdienst.
- *Aufbewahrungszeit*
  Äquivalent zu den Messkomponenten müssen die erstellten Meter Records mindestens die angegebene Zeit aufbewahrt werden, um nachträgliches Überprüfen respektive Reproduzieren zu ermöglichen. Dies wird i.d.R. in der Dienstvereinbarung zwischen Dienstleister und Kunde festgelegt.
- *Zugewiesene, weiterverarbeitende Anwendungen*
  Um nur autorisierten Anwendungen den Zugriff auf Usage Records zu erlauben, müssen entsprechend eindeutige Identifikatoren aller autorisierten Komponenten festgelegt werden.

❸ Charging Komponente Die Charging Komponente führt die eigentliche Berechnung der Gebühren durch und erstellt *Customer Detailed Records (CDR)*. Folgende Konfigurationen sind hierfür notwendig:

- *Identifikatoren der zugeordneten Kollektoren*
  Auf Basis von Usage Records, die Kollektoren generieren, werden von der Charging Komponente die Customer Detailed Records erstellt. Hierzu müssen die Identifikatoren der zugeordneten Kollektoren spezifiziert werden. Es muss darauf hingewiesen werden, dass die Kollektoren nicht auf das Bereitstellen von Daten des in Anspruch genommenen Dienstnutzungsvolumen beschränkt sind, sondern auch Daten über beispielsweise die Dienstgüte zur Verfügung stellen können.
- *Einsammelintervall*
  Der Zeitraum zwischen den Einsammelzeitpunkten der Usage Records wird mit diesem

[4] Dies kann beispielsweise ein Rechnername, IP Adresse, Login Account, etc. sein.

 Als Buch erhältlich im Verlag Dr. Hut, München, www.dr.hut-verlag.de (ISBN 3-934767-97-4)

Parameter festgelegt. Die Generierung der Customer Detailed Records wird nach Abschluss des Sammelvorgangs angestoßen und entspricht meist den vereinbarten Intervallen der Rechnungsstellung (beispielsweise einmal im Monat). Unter Umständen kann aber eine Generierung von CDRs von diesen Zeitintervallen abweichen (z.B. im Falle von Hot Billing).

- *Felder eines zu generierenden Customer Detailed Records*
  Äquivalent zu den bisher beschriebenen, zu erstellenden Datensätzen (siehe Meter Records und Usage Records) müssen auch in diesem Fall die Elemente eines Eintrags innerhalb des Customer Detailed Records festgelegt werden. Hierbei wird mindestens pro abzurechnenden Dienst das in Anspruch genommene Dienstnutzungsvolumen und die dafür in Rechnung gestellte Gebühr aufgenommen. Maximal kann aber die Granularität der CDRs auf die Genauigkeit der Usage Records erhöht werden.
- *Dienstidentifikator für die Haltung von Kundendaten*
  Jeder Kunde wird innerhalb einer Abrechnungs– und Rechnungsstellungsdomäne durch eine eindeutige *Accounting Customer ID* identifiziert. Um u.a. Dienstnutzer zu Kunden zuordnen zu können, wird an dieser Stelle der Dienst spezifiziert, der diese Information vorhält. In der Regel handelt es sich hierbei um eine Datenbank resp. um einen Verzeichnisdienst.
- *Datenbasis für Tarifdaten*
  Bei der Gebührenberechnung wird ein vereinbarter Tarif auf die in den Usage Records aufgeschlüsselten Nutzungsdaten angewendet. Jedem Kunden ist pro Dienst genau ein Tarif zugeordnet, der in einer Datenbasis abgelegt ist. Somit muss der Charging Komponente die Datenbasis genannt werden, in welcher der Tarif zu finden ist.
- *Aufbewahrungszeit*
  Ähnlich zu den bisher vorgestellten Komponenten müssen die erstellten Customer Detailed Records mindestens die angegebene Zeit aufbewahrt werden, um ein nachträgliches Überprüfen und Reproduzieren zu ermöglichen. Die Aufbewahrungszeit wird in der Dienstvereinbarung zwischen Dienstleister und Kunde geregelt.
- *Zugewiesene, weiterverarbeitende Anwendungen*
  Um nur autorisierten Anwendungen, wie Billing Systemen oder Reporting Tools, den Zugriff auf CDRs zu erlauben, müssen entsprechend eindeutige Identifikatoren aller autorisierten Komponenten festgelegt werden.

❹ Billing System Das Billing System ist für die Ausstellung der Rechnung sowie für deren Zusendung an die Rechnungsempfänger verantwortlich. Hierfür sind die folgenden Konfigurationen notwendig:

- *Identifikatoren der zugeordneten Charging Komponente*
  Die Rechnung wird auf Basis der von einer Charging Komponente gelieferten Customer Detailed Records (CDR) erstellt. Damit muss dem Billing System die ihm zugeordnete Charging Komponente genannt werden.
- *Einsammelintervall*

Hierbei wird der Einsammelzeitpunkt der CDRs bestimmt. Mit dem Erhalt der CDRs wird mit der Zusammenstellung der Rechnung begonnen.

- *Datenbasis für Rechnungsempfänger*
  Einem Dienstnehmer können mehrere Rechnungsempfänger zugeordnet werden, deren Daten in einer entsprechenden Datenbasis gespeichert werden. Ausschlaggebend ist allerdings, dass immer nur genau eine Rechnung eine Zahlungsaufforderung enthält.
- *Felder der zu generierenden Rechnung(en)*
  Sowohl die in eine Rechnung aufgenommene Information als auch die Art und Weise der Aufbereitung dieser Information (Layout) kann pro Rechnungsempfänger unterschiedlich sein.
- *Aufbewahrungszeit*
  Auch das Billing System muss Daten über die eigenen, erstellten Rechnungen für eine spätere Überprüfung aufbewahren.

❺ Reportingkomponente

- *Identifikatoren der am Reporting beteiligten Komponenten*
  Üblicherweise vollführen Reportingkomponenten die gleichen Schritte wie Kollektoren, allerdings weisen die erstellten Reports eine andere (meist feinere) Granularität auf. Somit sind in den meisten Fällen der Reportingkomponente die IDs der Messkomponenten zu übergeben. Allerdings ist es durchaus denkbar, dass auch Komponenten, die ausschließlich an der Dienstbereitstellung beteiligt sind, in das Reporting mit aufgenommen werden.
- *Felder der zu erstellenden Reports*
  Es muss das Layout der generierten Reports festgelegt werden.
- *Datenbasis für Reportempfänger*
  Einem Dienstnehmer können mehrere Reportempfänger zugeordnet werden, deren Daten in einer entsprechenden Datenbasis gespeichert werden.

❻ Rechnungsprüfungskomponente Zentrale Aufgabe der Rechnungsprüfungskomponente ist die Überprüfung der ausgestellten Rechnungen auf deren Korrektheit. Hierbei sind die folgenden Konfigurationen notwendig:

- *Identifikatoren und Konfigurationen der Messkomponenten, Kollektoren, Charging Komponente und Billing System*
  Grundsätzlich werden von der Rechnungsprüfungskomponente die gleichen Schritte durchgeführt, wie sie zur erfolgreichen Rechnungsstellung notwendig sind. Hierbei liegt aber die zentrale Aufgabe bei der Korrektheitsüberprüfung der Ausgabe jeder an der Abrechnung beteiligten Komponente bei gegebener Eingabe. Hierbei wird immer ein eindeutiges Urteil (engl.: *verdict*) gefällt. Die Anzahl der zu überprüfenden Komponenten können auch reduziert werden, so dass beispielsweise auf der untersten Ebene der Messkomponenten nur Stichproben durchgeführt werden.

 Als Buch erhältlich im Verlag Dr. Hut, München, www.dr.hut-verlag.de (ISBN 3-934767-97-4)

❼ **Zahlungsüberwachungskomponente** Diese Komponente überwacht, ob der in Rechnung gestellte Betrag von Kundenseite auch in den vereinbarten Zeitintervallen gezahlt wird. Hierzu werden folgende Konfigurationen durchgeführt:

- *Vereinbartes Zahlungsintervall*
  In Kombination mit der eindeutigen Accounting Customer ID wird das jeweilige Zahlungsintervall eingetragen.
- *Identifikator des Buchungssystems*
  Das Buchungssystem registriert alle Zahlungseingänge und kann damit die für die Zahlungsüberprüfung notwendige Information liefern.

Im Anschluss an diesen Teilprozess wird der Dienst und damit der Abrechnungsvorgang in Betrieb genommen. Als nächste Teilprozesse werden die „Dienstanalyse“ (siehe Abschnitt 4.3.9) und die „Nutzungserfassung“ (siehe Abschnitt 4.3.11) ausgeführt.

4.3.9 Dienstanalyse

Der in Abbildung 4.13 visualisierte Teilprozess der *Dienstanalyse*, der in der Betriebsphase des Dienstlebenszyklus stattfindet, umfasst Aktivitäten zur Bestimmung von *Messeinheiten*, die es ermöglichen, die bei einer Dienstnutzung in Anspruch genommene Ressourcennutzung auszudrücken. Dieser Teilprozess unterscheidet sich zu der während des Teilprozesses „Implementierung der Messkomponenten“ (vgl. Abschnitt 4.3.5) stattfindenden Dienstanalyse dahingehend, dass diese unabhängig von den vereinbarten Abrechnungseinheiten durchgeführt wird und somit wesentlich allgemeiner ist. Die auf diese

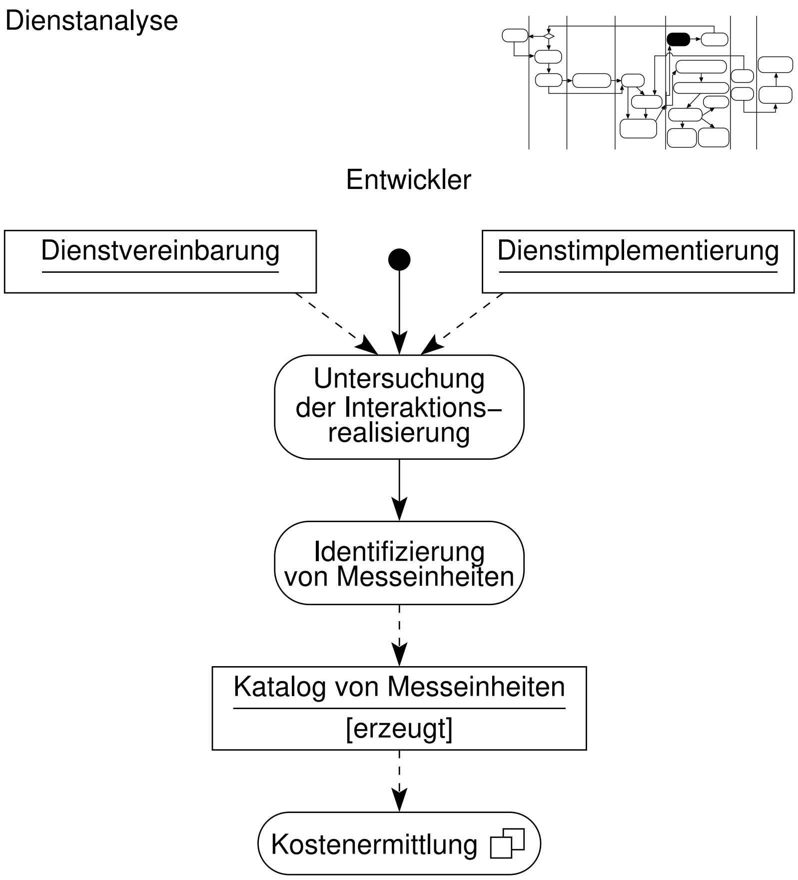

Abbildung 4.13: Dienstanalyse

Weise bestimmten Messeinheiten werden im Teilprozess „Kostenermittlung“ für die Aufschlüsselung der tatsächlich entstandenen Kosten verwendet und können auch als Grundlage für Tarifneuverhandlungen dienen.

Zur Identifizierung der Messeinheiten wird die Realisierung der vereinbarten Dienstfunktionalität analysiert: Jede in der Dienstvereinbarung enthaltene Funktionalitätseigenschaft, repräsentiert durch die Beschreibung einer Interaktionsbeziehung zwischen Dienstnehmer– und Dienstleisterseite, wird auf deren tatsächliche Realisierung durch eine Ressource hin untersucht. Anschließend wird die *Messbarkeit* der Interaktion festgestellt. Falls möglich, werden einer untersuchten Interaktion Messeinheiten zugewiesen. Ergebnis dieses Schritts ist ein *Katalog von Messeinheiten*, die jeweils bezogen auf eine Interaktion die Inanspruchnahme von Ressourcen ausdrücken können. Als Struktur des Messeinheitenkatalogs wird eine Aufteilung gemäß des Dienstmodells aus Abschnitt 2.1.1 vorgeschlagen: Die Messeinheiten werden jeweils nach Nutzungs– und Managementinteraktionen gegliedert.

Nicht unerwähnt sollte bleiben, dass die Ausführung dieses Teilprozesses grundsätzlich auch bei Fehlen genau einer der beiden Entitäten Dienstvereinbarung respektive Dienstimplementierung möglich ist. Die Identifizierung von Messeinheiten ohne Dienstimplementierung wird u.a. in [Schm 01] untersucht.

4.3.10 Kostenermittlung

Der Teilprozess der *Kostenermittlung*, der in Abbildung 4.14 dargestellt ist und in der Betriebsphase des Dienstlebenszyklus stattfindet, beschäftigt sich mit der Erstellung einer *Kostenaufstellung*, welche die Kosten, die durch die Dienstbereitstellung beim Diensterbringer entstehen, aufschlüsselt. Hierbei wird als Basis für die Identifizierung der Kosten ein *Kostenmodell* [ITIL 01], die Dienstimplementierung sowie der Katalog der Messeinheiten aus dem Dienstanalyseprozess herangezogen. Grundsätzlich kann jede an der Dienstbereitstellung beteiligte Entität auf die speziell durch diese Entität verursachten Kosten untersucht werden. Ein Kostenmodell dient hierbei als Hilfestellung: Dieses wird von einem *IT Finance Manager* aufgestellt und enthält u.a. Richtlinien, welche Entitäten grundsätzlich in die Kostenaufstellung miteinbezogen werden sollen. Bezüglich jeder zu betrachtenden Entität wird eine *Kostenfunktion* aufgestellt, die im günstigsten Fall die Kosten in Relation zu weiteren Parametern wie Güte und Nutzungsvolumen darstellt. Im einfachsten Fall ist die Kostenfunktion eine Konstante, die z.B. den Anschaffungspreis einer Entität, wie einem Endsystem, darstellt. Die Art und Weise, wie eine Kostenfunktion aufzustellen ist und welche Parameter aufgenommen werden, wird ebenfalls in einem Kostenmodell vorgeschrieben und basiert i.d.R. auf betriebswirtschaftlichen Analysen. Zum Schluss werden die einzelnen Kosten zu einer *Kostenaufstellung* zusammengeführt. Auch in diesem Fall liefert das Kostenmodell Anhaltspunkte, wie diese Aufstellung zu erfolgen hat. Denkbar ist z.B. die Aufschlüsselung der Kosten nach Dienst, Kunde, Monat, etc.

Insgesamt ist dieser Prozess der betriebswirtschaftlichen Domäne zuzuordnen, so dass die Rolle des IT Finance Managers diesem Bereich zufällt.

 Als Buch erhältlich im Verlag Dr. Hut, München, www.dr.hut-verlag.de (ISBN 3-934767-97-4)

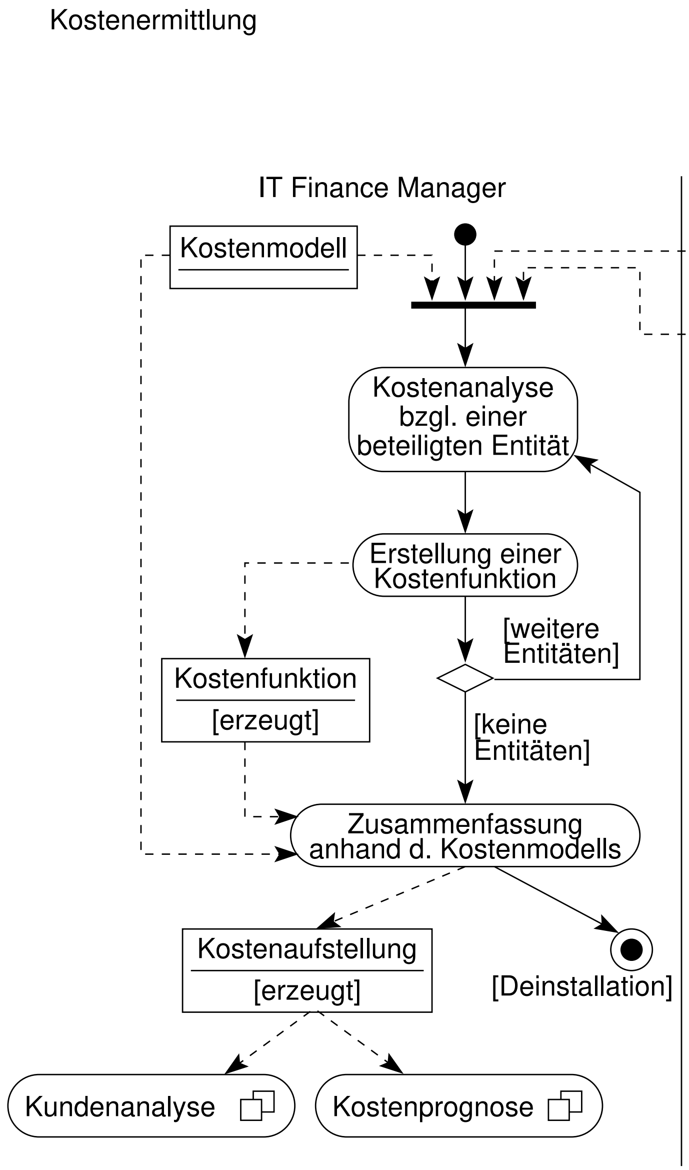

Abbildung 4.14: Kostenermittlung

Sofern eine Neuverhandlung des Tarifmodells vorgesehen ist, kann entweder mit der „Kundenanalyse“ (siehe Abschnitt 4.3.2) oder mit der „Kostenprognose“ (siehe Abschnitt 4.3.3) fortgefahren werden. Die erstellte Kostenaufstellung dient hierbei der „Kostenprognose“ als Eingabeentität.

4.3.11 Nutzungserfassung

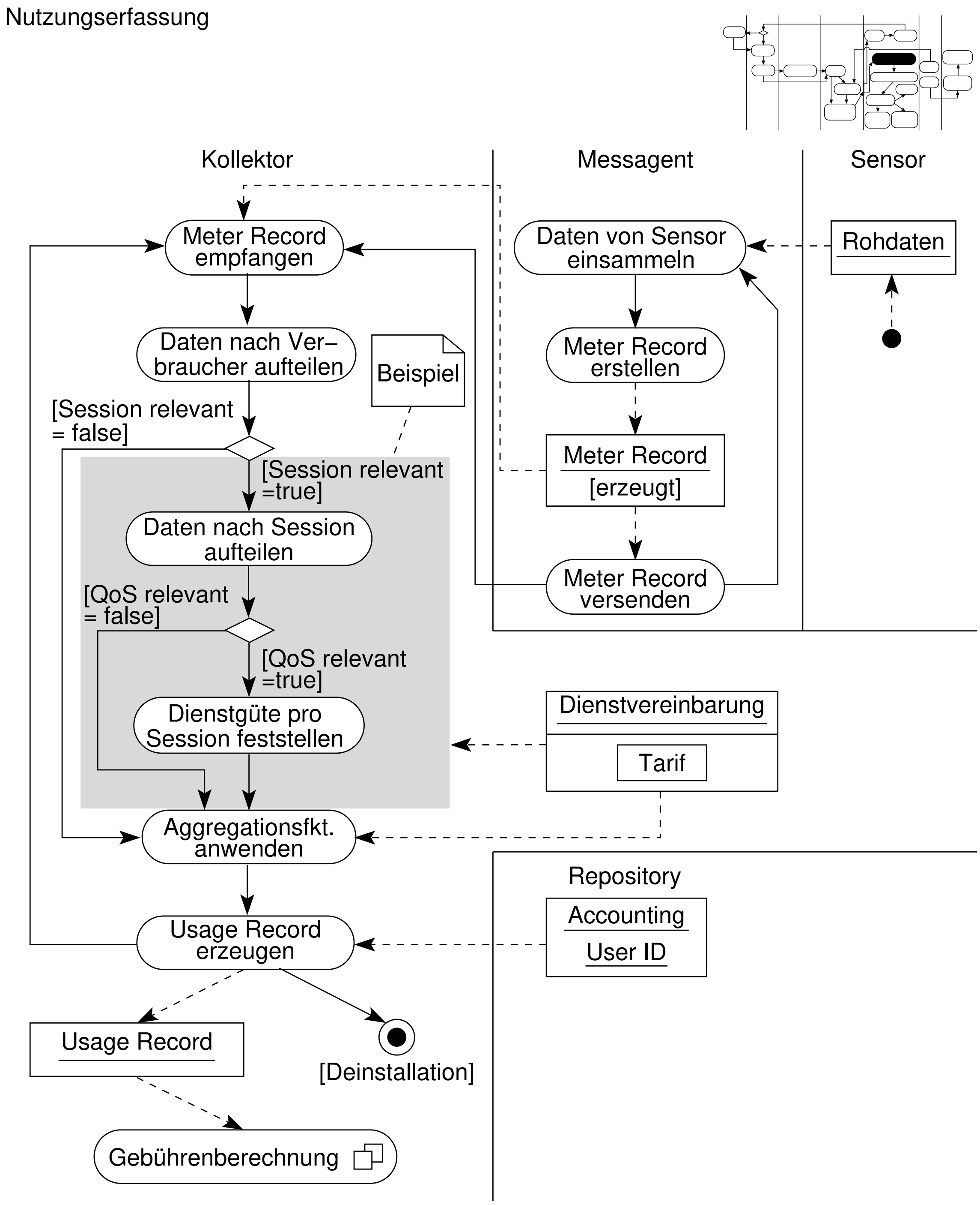

Abbildung 4.15: Nutzungserfassung

Ziel des in Abbildung 4.15 visualisierten Teilprozesses, der in der Betriebsphase des Dienst-

 Als Buch erhältlich im Verlag Dr. Hut, München, www.dr.hut-verlag.de (ISBN 3-934767-97-4)

lebenszyklus stattfindet, ist es, auf Basis von Messdaten sogenannte *Usage Records*, die einer aggregierten, komprimierten Form der gemessen Daten entsprechen, zu erzeugen. Im Folgenden wird der dargestellte Ablauf skizziert.

Zunächst müssen die von einem Sensor gemessenen Daten über die Dienstnutzung durch den zugewiesenen Messagenten eingesammelt werden. Die Form und Granularität der hierbei erfassten *Rohdaten* ist vom Sensortyp abhängig. Unter dem Begriff Rohdaten werden im Folgenden *alle* Daten verstanden, die für die Durchführung der vereinbarten Abrechnung notwendig sind. Dies schließt demnach nicht nur Daten über das Dienstnutzungsvolumen mit ein, sondern auch Daten über z.B. die erfahrene Dienstgüte. Die Festlegung, welche Daten gesammelt und damit welche Sensoren eingesetzt werden, hängt vom vereinbarten Tarifmodell respektive von den vereinbarten Abrechnungseinheiten A_i und den Parametern x_i ab. Der Messagent erstellt schließlich in den jeweils konfigurierten Zeitintervallen (siehe Abschnitt 4.3.8) Meter Records, indem dieser die Rohdaten in der konfigurierten Art und Weise aufbereitet und mit zusätzlichen Daten anreichert (z.B. indem der Sensor– und Messagentort, der Name des Dienstes, etc. hinzugefügt wird).

Im nächsten Schritt werden die erzeugten Meter Records in den konfigurierten Zeitintervallen vom Kollektor eingesammelt. Daraufhin werden die in den Meter Records enthaltenen Messdaten nach *Verbraucher* aufgeteilt. Die für diese Aktivität benötigte Information, mit der verschiedene Verbraucher unterschieden werden (z.B. IP–Adresse, Betriebssystemkennung, etc.) und die in den Meter Records ausgedrückt wird (z.B. Stelle in einer *Comma Separated List (CSL)*), wird ebenfalls im vorhergehenden Teilprozess während der Konfiguration des Abrechnungssystems (siehe Abschnitt 4.3.8) eingestellt.

Während die Aufteilung der Meter Records nach Verbraucher für die Erzeugung von Usage Records zwingend notwendig ist, sind weitere Schritte, wie die Aufteilung nach *Sitzungen (Sessions)* vom vereinbarten Tarifmodell abhängig und somit nicht allgemein festlegbar. Falls beispielsweise das durch den Verbraucher in Anspruch genommene Dienstnutzungsvolumen in Abhängigkeit von Zeitabschnitten wie Tageszeiten, Wochentag, etc. abgerechnet werden soll, müssen die Verbrauchsdaten in Sitzungen aufgeteilt werden (in Abbildung 4.15 mit einem grauen Kasten hinterlegt). Eine Sitzung wird im Folgenden mit dem Beginn und Ende einer Dienstnutzung gleichgesetzt, in der die Kriterien zur Entscheidungsfindung, welcher Preis für eine Abrechnungseinheit anzuwenden ist, sich nicht ändern. Damit ist eine Sitzung gleichzusetzen mit der *eindeutigen* Abbildung des u.U. mehrdimensionalen Parameters x auf einen Preis pro Abrechnungseinheit A_i durch Anwendung der Preisfunktion p_i[5]. Somit ist der Sitzungsbegriff insbesondere auch Voraussetzung für die Abrechnung nach der erfahrenen Dienstgüte. Im einfachsten Fall können die Messdaten in genau eine Sitzung eingeteilt werden, welches dem Fall entspricht, dass die Einteilung in unterschiedliche Sitzungen nicht relevant ist. Der Beginn und das Ende einer Sitzung werden durch Zeitstempel festgestellt. Damit müssen die Messdaten entsprechende Zeitangaben aufweisen und es muss die Möglichkeiten bestehen, die gemessenen Verbrauchsdaten Sitzungen zuordnen zu können.

[5] Abschnitt 4.3.4 erklärt die verwendeten Parameter.

Um aus den Messdaten die tatsächlichen *Abrechnungeinheiten* zu destillieren, wird im Anschluss eine *Aggregationsfunktion* angewendet. Die Anwendung einer Aggregationsfunktion entspricht der Abbildung der in den Meter Records enthaltenen *Messeinheiten* auf die in den Usage Records enthaltenen Abrechnungseinheiten. Dies umfasst sowohl die Bildung der Summe über die Messeinheiten einer Session als auch eventuell die Anwendung anderer Funktionen.

Insgesamt werden die auf diese Weise gewonnenen Daten zu sogenannten *Usage Records* zusammengefasst. Hierzu werden die Verbrauchsdaten, die in den gezählten Abrechnungseinheiten vorliegen, einem eindeutigen *Accounting User ID* zugewiesen, der innerhalb einer Abrechnungs– und Rechnungsstellungsdomäne den Dienstnutzer eindeutig identifiziert. Ausdrücklich sei an dieser Stelle darauf hingewiesen, dass nicht unbedingt eine 1:1–Beziehung zwischen tatsächlichem Verbraucher (z.B. der tatsächlichen Person, die einen Dienst in Anspruch nimmt) und einem Accounting User ID bestehen muss. Beispielsweise kann ein Verbraucher auch mehreren Accounting User IDs entsprechen, wenn ein– und dieselben Messdaten zur Erstellung von unterschiedlichen Rechnungen verwendet werden (z.B. Abrechnung nach Mitarbeiter und/oder Abteilung: die Messdaten bleiben dieselben, jedoch einmal kumuliert zu Mitarbeitern und einmal zu Abteilungen).

In Anschluss an die Nutzungserfassung erfolgt der Teilprozess der „Gebührenberechnung“, der in Abschnitt 4.3.12 beschrieben ist.

 Als Buch erhältlich im Verlag Dr. Hut, München, www.dr.hut-verlag.de (ISBN 3-934767-97-4)

4.3.12 Gebührenberechnung

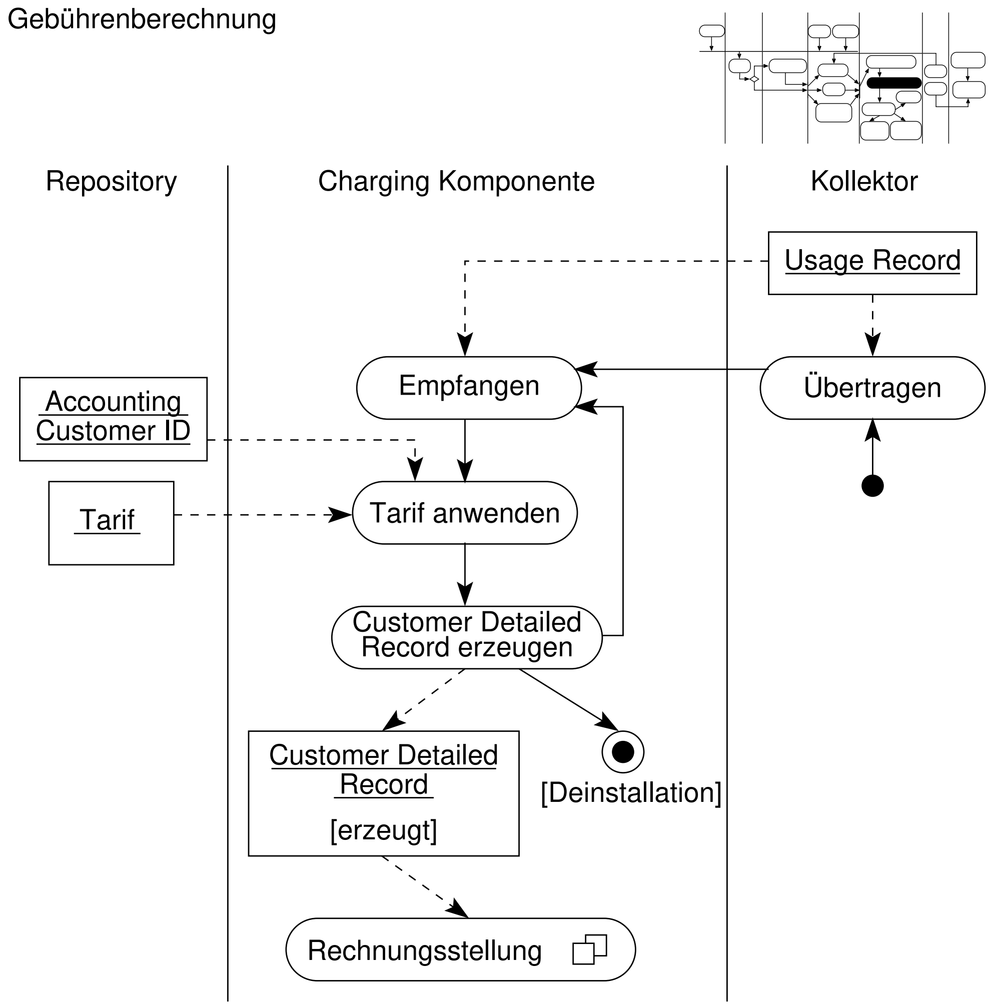

Abbildung 4.16: Gebührenberechnung

Ziel des in Abbildung 4.16 dargestellten Teilprozesses, der in der Betriebsphase des Dienstlebenszyklus stattfindet, ist es, auf Basis von Usage Records, einen sogenannten *Customer Detailed Record*, das einer Aufstellung des im vereinbarten Abrechnungszeitintervalls in Anspruch genommenen Dienstnutzungsvolumens sowie den dabei entstandenen Gebühren entspricht, zu erzeugen.

Zunächst werden alle Usage Records von Kollektoren eingesammelt, die der betrachteten Charging Komponente zugeordnet sind. Auf Basis der in den Usage Records enthaltenden Daten wie den Accounting User IDs sowie den jeweiligen Dienstzugangspunkten, wird die dazu passende *Accounting Customer ID* ermittelt, die den Dienstnehmer eindeutig innerhalb einer Abrechnungs– und Rechnungsstellungsdomäne identifiziert. Das einzusetzende Tarifmodell wird schließlich mit Hilfe der Accounting Customer ID und dem Diensttyp ermittelt. Anschließend wird das Tarifmodell auf jeden Eintrag eines Usage Records angewendet. Mit den in den Usage Records enthaltenen Daten ist die Bestimmung genau eines Tarifs möglich, d.h. es wird der Parameters x eindeutig festgelegt. Damit entspricht ein Eintrag in einem Usage Record immer der eindeutigen Anwendung der Preisfunktionen $p_i(x)$ und damit der eindeutigen Abbildung des Parameters x auf einen Preis pro Abrechnungseinheit A_i. Im Anschluss daran wird pro Eintrag im Usage Record die in Rechnung zu stellende Gebühr berechnet.

Das von der Charging Komponente erzeugte Customer Detailed Record kann neben den einzeln berechneten Gebühren auch Einzelheiten über die Zusammensetzung einer Gebühr, d.h. die Belegung des Parameters x, enthalten. Die Elemente eines CDRs werden i.d.R. in der Dienstvereinbarung festgelegt und im Konfigurationsteilprozess (siehe Abschnitt 4.3.8) eingestellt.

 Als Buch erhältlich im Verlag Dr. Hut, München, www.dr.hut-verlag.de (ISBN 3-934767-97-4)

4.3.13 Rechnungsstellung

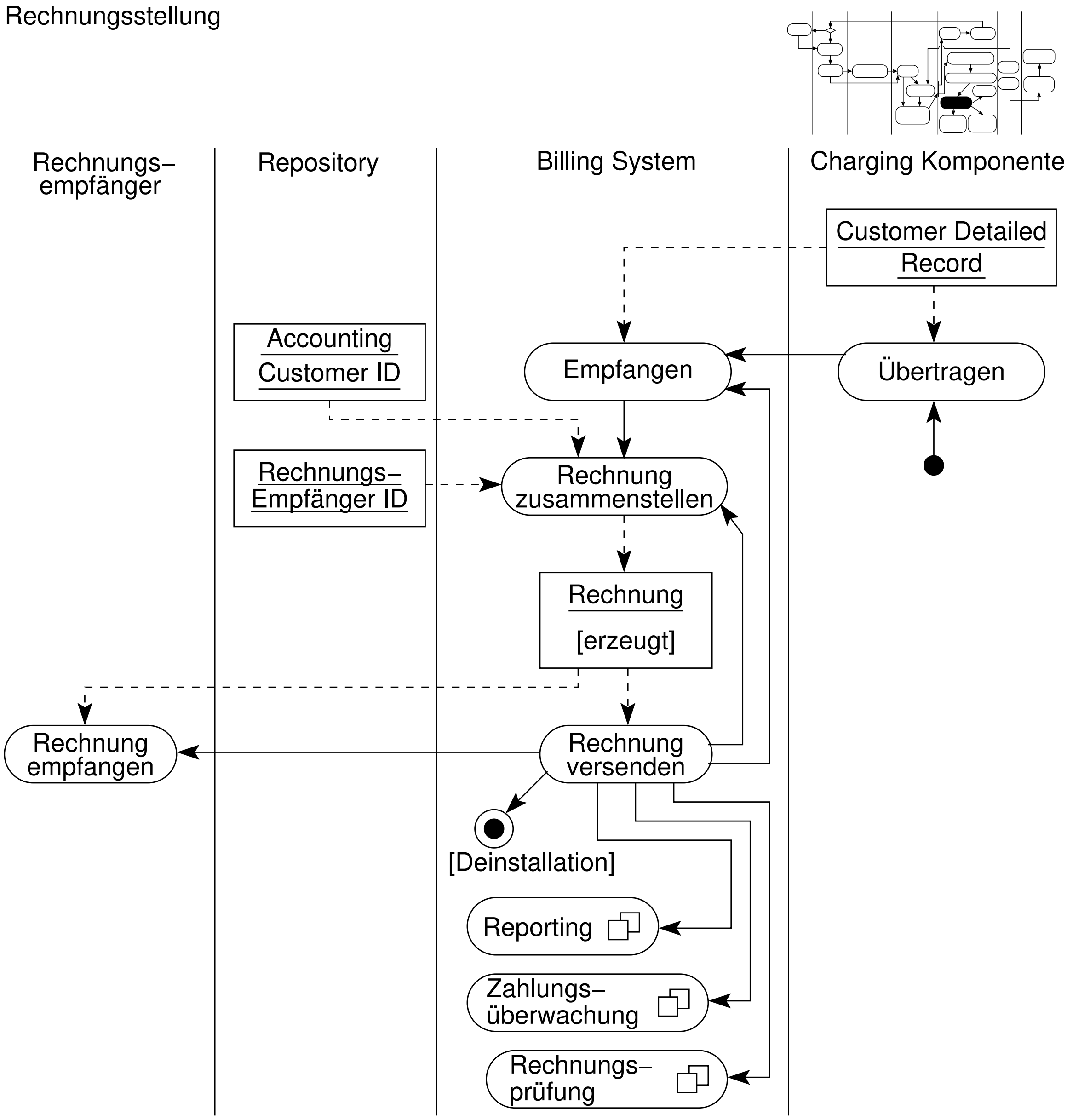

Abbildung 4.17: Rechnungsstellung

Der in Abbildung 4.17 dargestellte Teilprozess der Rechnungsstellung, der in der Betriebsphase des Dienstlebenszyklus stattfindet, stellt auf Basis von Customer Detailed Records Rechnungen aus und versendet diese an die entsprechenden Empfänger.

Im Detail werden in den vereinbarten Abrechnungszeitintervallen CDRs von der zugewiesenen Charging Komponente eingesammelt. Mit Hilfe der in den CDRs enthaltenen Accounting Customer ID werden die Rechnungsempfänger IDs aus der Kundendatenbank abgefragt. Pro

Rechnungsempfänger wird eine dem Rechnungsempfänger entsprechende Rechnung zusammengestellt. Das heißt, dass sich grundsätzlich die Rechnungen sowohl im Inhalt als auch in der Gestaltung in Abhängigkeit von den Rechnungsempfängern unterscheiden können. Die Rechnungsgestaltung wie auch deren Inhalte werden in der Dienstvereinbarung ausgehandelt und im Konfigurationsteilprozess (siehe Abschnitt 4.3.8) eingestellt. In aller Regel enthält allerdings nur genau eine Rechnung eine Zahlungsaufforderung. Die ausgestellte Rechnung wird schließlich in der vereinbarten Art und Weise an den Rechnungsempfänger versendet.

4.3.14 Reporting

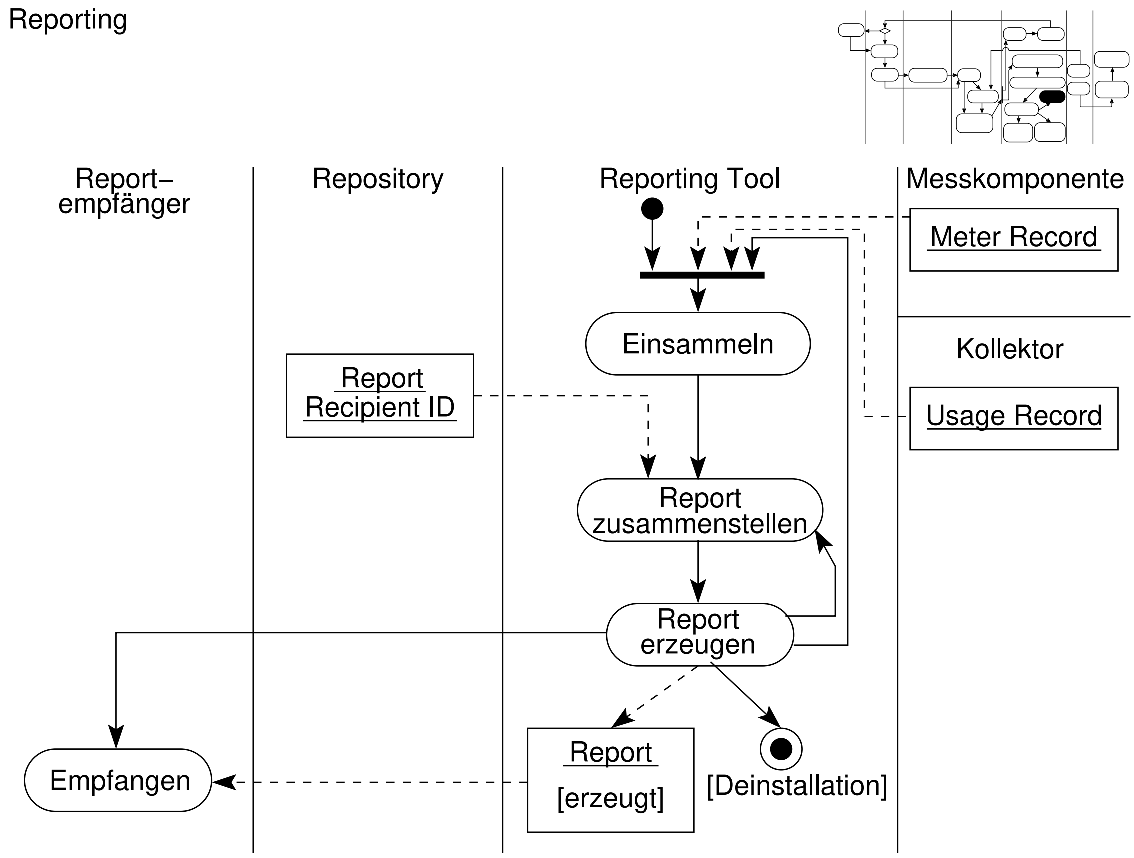

Abbildung 4.18: Reporting

Ziel des in Abbildung 4.18 dargestellten Teilprozesses, der in der Betriebsphase des Dienstlebenszyklus stattfindet, ist die den jeweiligen Kundenwünschen entsprechende Erstellung und Versendung von Reports.

Reports enthalten sowohl eine detaillierte Aufstellung als auch Aufbereitung (z.B. grafische Darstellung) der in einem Zeitintervall tatsächlich stattgefundenen Dienstnutzung. Mit Hilfe

 Als Buch erhältlich im Verlag Dr. Hut, München, www.dr.hut-verlag.de (ISBN 3-934767-97-4)

von Reports sollen einerseits von Dienstnehmerseite die Einhaltung von vertraglich zugesicherten Diensteigenschaften („Service Levels“) überprüft als auch andererseits ausgestellte Rechnungen nachvollzogen werden können. Prinzipiell können in Reports beliebige Informationen über die Dienstnutzung mitaufgenommen werden. In der Regel werden Reports aber auf der Basis derselben Daten erzeugt wie eine Rechnung, nur dass der Detailgrad wesentlich feingranularer ist. Somit werden von dem *Reporting Tool* sowohl Meter Records von den Messkomponenten als auch Usage Records von den Kollektoren eingesammelt. Anschließend wird ein Report nach den vertraglichen Vereinbarung zusammengestellt. Die Art und Weise der Report–Zusammenstellung wurde bereits im Konfigurationsteilprozess (siehe Abschnitt 4.3.8) eingestellt. Zum Schluß erfolgt die Zusendung des Reports an den *Reportempfänger*. Bei mehr als einem Reportempfänger können sich unter Umständen die Inhalte der Reports unterscheiden.

4.3.15 Rechnungsprüfung

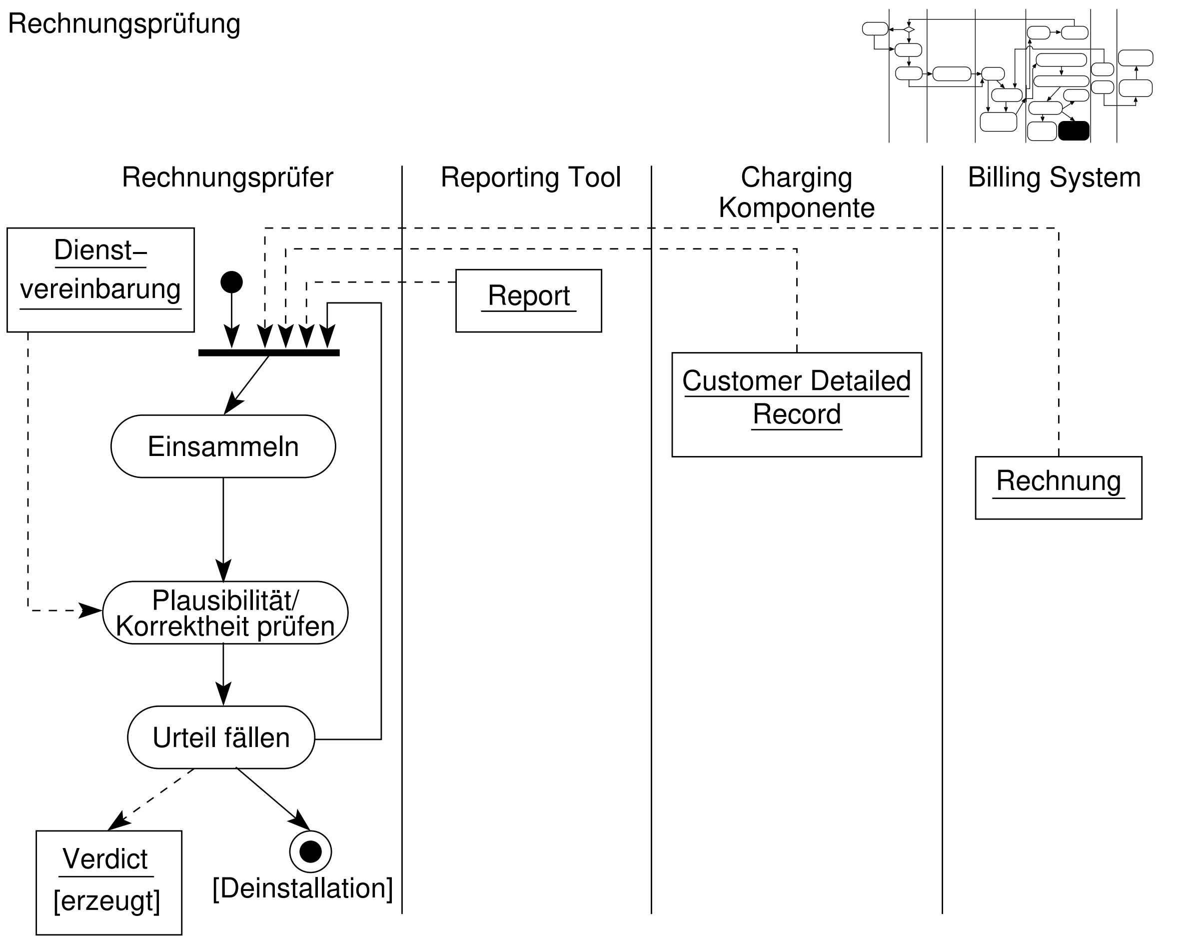

Abbildung 4.19: Rechnungsprüfung

Der in Abbildung 4.19 dargestellte Teilprozess der Rechnungsprüfung, der in der Betriebsphase des Dienstlebenszyklus stattfindet, prüft die vom Dienstleister ausgestellten Rechnungen auf Plausibilität und Korrektheit.

Hierzu werden vom *Rechnungsprüfer* die in einem Zeitraum ausgestellten Reports, Customer Detailed Records und Rechnungen eingesammelt. Auf Basis der Dienstvereinbarung und den in den genannten Dokumenten enthaltenen Daten wird die vom Dienstleister erbrachte und die von Dienstnehmerseite in Anspruch genommene Leistung verglichen sowie die ausgestellte Rechnung auf Korrektheit und Plausibilität geprüft und ein Urteil (auch *Verdict* genannt) gefällt. Die Dienstvereinbarung kann u.U. Kriterien und Maßnahmen enthalten, welche die Korrektheit der Rechnung feststellbar machen (z.B. durch nochmaliges Nachrechnen des in Rechnung gestellten Betrags auf Basis der CDRs). Ein negatives Urteil, d.h. im Falle, dass eine Rechnung als inkorrekt bezeichnet wird, kann wiederum Auslöser für weitere Aktionen sein, wie z.B. dass Aktivitätsprotokolle aller an der Abrechnung und Dienstnutzung beteiligten Komponenten angefordert werden, die Zahlung der Rechnung verweigert wird, etc.

Der Rechnungsprüfer wird als eine Rolle verstanden, die „domänenübergreifend", d.h. sowohl auf Dientnehmer– als auch auf Dienstleisterseite, eingenommen werden kann. Während der Dienstleister daran interessiert ist, dass nicht zu wenig in Rechnung gestellt wird, ist der Dienstnehmer daran interessiert, dass die Rechnung keinen zu hohen Betrag enthält. Somit verfolgen beide Seiten das Ziel, dass alle an der Abrechnung beteiligten Komponenten korrekt arbeiten. In einigen Fällen wird die Rechnungsprüfung auf Dienstnehmerseite sogar auf Basis eigener Messungen durchgeführt, welche mit den Reports des Dienstleisters verglichen werden.

4.3.16 Zahlungsüberwachung

Ziel des in Abbildung 4.20 dargestellten Teilprozesses, der in der Betriebsphase des Dienstlebenszyklus stattfindet, ist die Überwachung des Zahlungsverkehrs.

In der zwischen Dienstleister und Dienstnehmer geschlossenen Dienstvereinbarung werden neben der Art und Weise der Zahlung und des Zahlungsintervalls auch Eskalationsmechanismen vereinbart, die zum Einsatz kommen, wenn die Zahlungen nicht in vereinbarter Form erfolgen. Für den betrachteten Teilprozess ist nur relevant, dass die Zahlungsüberwachungskomponente das Buchungssystem auf Zahlungseingänge überwacht und bei Überschreiten des Zahlungsintervalls, in dem eine korrekte Zahlung zu erfolgen hat, entsprechende Eskalationsereignisse generiert werden, woraufhin vorkonfigurierte Eskalationsmechanismen ausgelöst werden können. Nach erfolgter Zahlung wird das Eskalationsniveau wieder gesenkt, sofern dieses vorher erhöht wurde.

 Als Buch erhältlich im Verlag Dr. Hut, München, www.dr.hut-verlag.de (ISBN 3-934767-97-4)

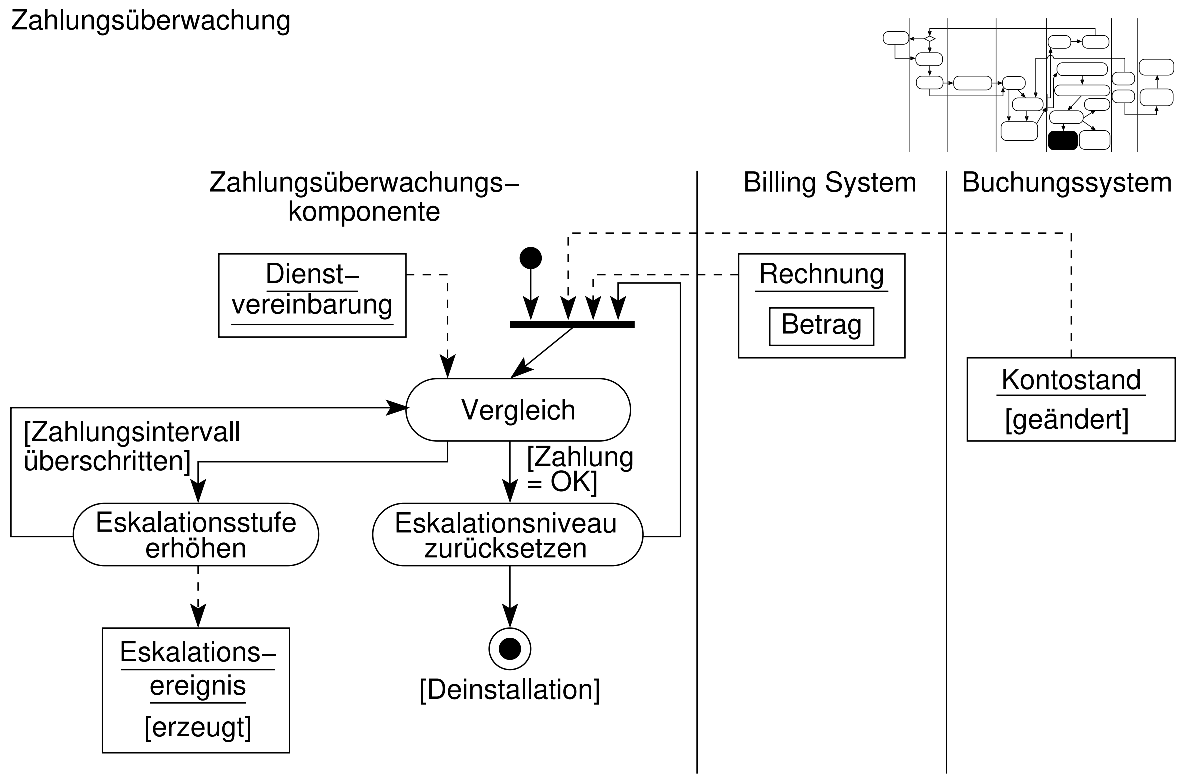

Abbildung 4.20: Zahlungsüberwachung

4.3.17 Deinstallation

Der in Abbildung 4.21 dargestellte Teilprozess der Deinstallation, der in der gleichnamigen Dienstlebenszyklusphase stattfindet, beschäftigt sich mit dem Abbau von abrechnungsbezogenen, kundenspezifischen Konfigurationen.

Im Einzelnen wird üblicherweise beim Auflösen der Dienstvereinbarung, beispielsweise nach Ablauf der Vertragslaufzeit, die kundenspezifische Konfiguration des Abrechnungssystems als auch des Abrechnungsmanagementsystems entfernt. Sind hierbei dedizierte Systeme verwendet worden, so wird das vollständige System abgebaut. Dies ist insbesondere bei Messkomponenten üblich, die in der Dienstnehmerdomäne ausschließlich das Dienstnutzungsvolumen messen und nur zu Abrechnungszwecken eingesetzt werden.

Mit der Deinstallation aller an der Abrechnung beteiligten Komponenten endet die Bereitstellung der Abrechnungsfunktionalität.

4.3.18 Change–Phase

Für die Change–Phase sind keine allgemeinen Schrittfolgen anhand von Aktivitäten bzw. Teilprozessen angebbar. Die auftretenden Aktivitäten und deren Auswirkungen sind immer von

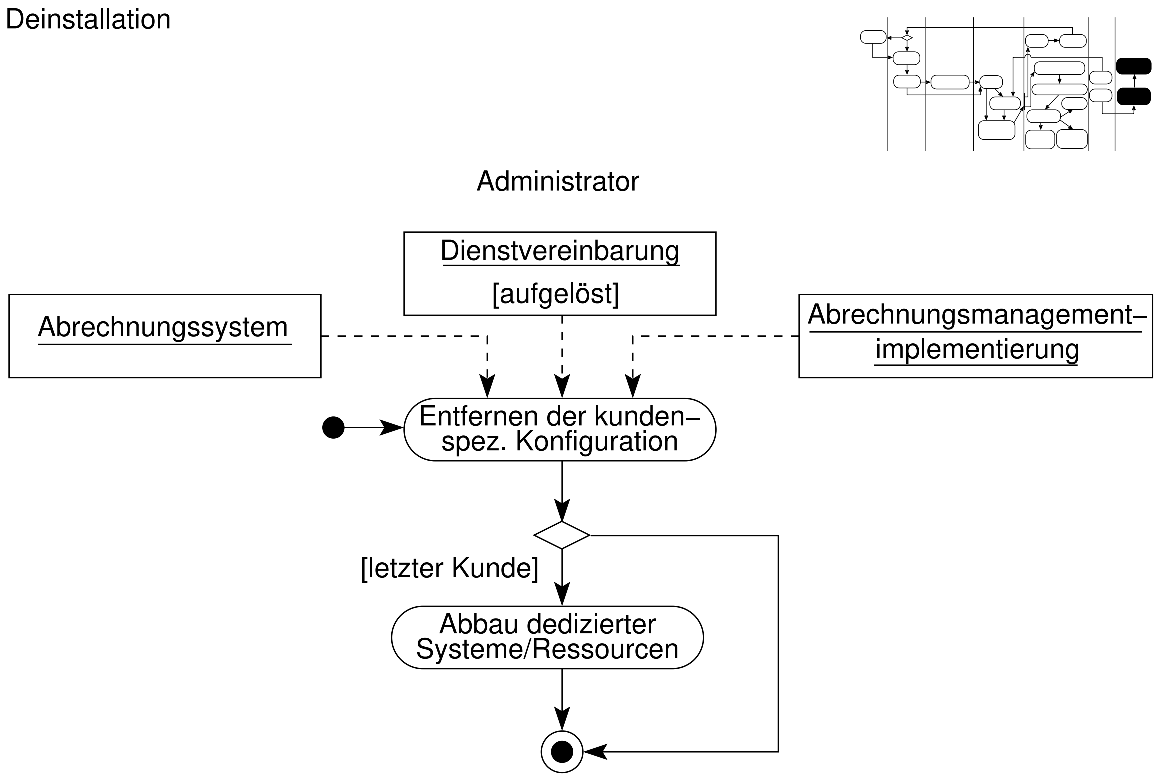

Abbildung 4.21: Deinstallation

der tatsächlich betriebenen Dienstimplementierung und Dienstmanagementimplementierung abhängig.

Allerdings sind für das Abrechnungsmanagement nur diejenigen Änderungsaktivitäten von Interesse, die entweder *direkt* das Abrechnungssystem bzw. Abrechnungsmanagementsystem betreffen oder die *indirekt* eine Auswirkung auf die Abrechnung respektive auf das Abrechnungs(management)system haben. Derartig indirekte Auswirkungen sind Aktivitäten, die ausgeführt werden müssen, um beispielsweise wieder einen konsistenten Zustand aus Sicht des Abrechnungsvorgangs zu erlangen. Konsequenterweise sind für diese Arbeit lediglich diejenigen Änderungsaktivitäten relevant, die Entitäten betreffen, die an der Abrechnung beteiligt sind. Das sind demnach die bisher identifizierten Entitäten wie Rollen, Ein–/Ausgabeentitäten (z.B. Dienstvereinbarung, Usage Records, etc.) sowie Komponenten, die das Abrechnungs– und Abrechnungsmanagementsystem realisieren (z.B. Messkomponenten, Kollektoren, etc.).

Um Auswirkungen von Änderungen an Entitäten untersuchen zu können, ist die Analyse der Abhängigkeiten resp. der Assoziationen einer Entität zu anderen Entitäten notwendig. Bisher wurden durch die Prozessanalyse dynamische Gegebenheiten untersucht und somit die relevanten Entitäten phasenweise entlang der Zeitachse identifiziert. Assoziationen einer Entität zu anderen Entitäten können damit zunächst nur auf Ebene der Teilprozesse identifiziert werden. Folglich fehlt bisher eine *phasen–* resp. *teilprozess–übergreifende* und damit *zeitunabhängige*

Analyse der Assoziationen. Um dies zu ermöglichen, kann im Kontrast zu den bisher in Form der Teilprozessmodelle entwickelten dynamischen Modelle ein *statisches Modell* verwendet werden, welches die Zeitdimension ausblendet. Damit ist es dann möglich, alle für die Abrechnung relevanten Assoziationen zwischen Entitäten auf einem Blick zu erkennen. In Konsequenz wird das Identifizieren von potentiell von Änderungen betroffenen Entitäten erheblich erleichtert. Dies wird u.a. in Kapitel 5 für die Spezifikation von Managementanweisungen verwendet.

In dem nun folgenden Abschnitt 4.4 wird zu diesem Zweck ein statisches Abrechnungsdienstmodell entworfen, das eine Erweiterung des generischen MNM Dienstmodells aus Abschnitt 2.1.1 ist.

4.4 Abrechnungsdienstmodell

Wie bereits im vorhergehenden Abschnitt 4.3.18 kurz erläutert wurde, erleichtert ein statisches Modell die Analyse der Auswirkungen von Änderungsaktivitäten, da es alle für den Abrechnungsvorgang relevanten Entitäten (bei ausgeblendeter Zeitdimension) in Beziehung zueinander setzt. Bei auftretenden Änderungen müssen in aller Regel lediglich die mit einem Objekt assoziierten Entitäten auf mögliche Auswirkungen untersucht werden. Desweiteren dient das statische Modell auch dazu, um zu bestimmen, welche Entitäten definiert und spezifiziert werden müssen, um eine dienst– und kundenorientierte Abrechnung zu realisieren. Hierbei sind diese Spezifikationen v.a. für die abrechnungsbezogenen Festlegungen in der Dienstvereinbarung relevant. Auf Basis der Spezifikationen innerhalb der Dienstvereinbarung kann die Realisierung des Abrechnungsprozesses auf Dienstleisterseite unterstützt werden, indem Anforderungen abgeleitet werden, die für Auswahl, Implementierung und Installation der notwendigen Komponenten und Prozesse erfüllt werden müssen. Hierfür muss allerdings erst festgestellt werden, welche Entitäten direkt oder indirekt von der Dienstvereinbarung abhängig sind.

Im Folgenden wird ein derartig statisches *Abrechnungsdienstmodell* entworfen, das eine Erweiterung des in Abschnitt 2.1.1 vorgestellten generischen MNM Dienstmodells ist. Es muss ausdrücklich betont werden, dass, auch wenn für die Notation des Abrechnungsdienstmodells statische UML Klassendiagramme verwendet werden, diese nicht, wie sonst üblich, automatisch zur Code–Generierung genutzt werden können und sich damit nicht direkt für die Implementierung in einer Programmiersprache eignen. Vielmehr dienen die Diagramme zur Analyse des Gesamtsystems, wie sie üblicherweise zu Beginn eines Software–Engineering–Prozesses durchgeführt wird. Auf Basis dieser Analyse ist es allerdings anschließend möglich, Klassendiagramme für die Implementierung zu entwerfen, wie es in [GHH+ 02] vorgeschlagen und in Kapitel 6 auch tatsächlich ausgeführt wird.

Das Abrechnungsdienstmodell wird zusammengestellt, indem die während der Analyse der Teilprozesse in Abschnitt 4.3 identifizierten Objekte und Rollen analog zu der in [GHH+ 02] beschriebenen Methodik in das MNM Dienstmodell eingefügt werden. Auftretende Interaktionen werden als Assoziationen in das Modell mit aufgenommen. Um die Darstellung übersichtlicher zu gestalten, ist das Abrechnungsdienstmodell in drei Teile, die in den folgenden drei Abschnitten vorgestellt werden, aufgeteilt. In Abschnitt 4.4.1 wird ein Überblick über das Abrechnungsdienstmodell dargestellt. Der in Abschnitt 4.4.2 vorgestellte Teil des Abrechnungsdienstmodells fokussiert auf die für die Dienstvereinbarung relevanten Entitäten. Abschnitt 4.4.3 konzentriert sich hingegen auf die Realisierung des Abrechnungsvorgangs auf Dienstleisterseite.

4.4.1 Dienstsicht

In Abbildung 4.22 ist ein Überblick über das Abrechnungsdienstmodell als Erweiterung des in Abbildung 2.2 auf Seite 14 dargestellten MNM Dienstmodells zu sehen. Die Unterschiede beschränken sich hierbei auf Erweiterungen der Dienstvereinbarung, der Managementfunktio-

 Als Buch erhältlich im Verlag Dr. Hut, München, www.dr.hut-verlag.de (ISBN 3-934767-97-4)

nalität des Dienstes und des CSM Zugangspunktes um abrechnungsrelevante Aspekte.

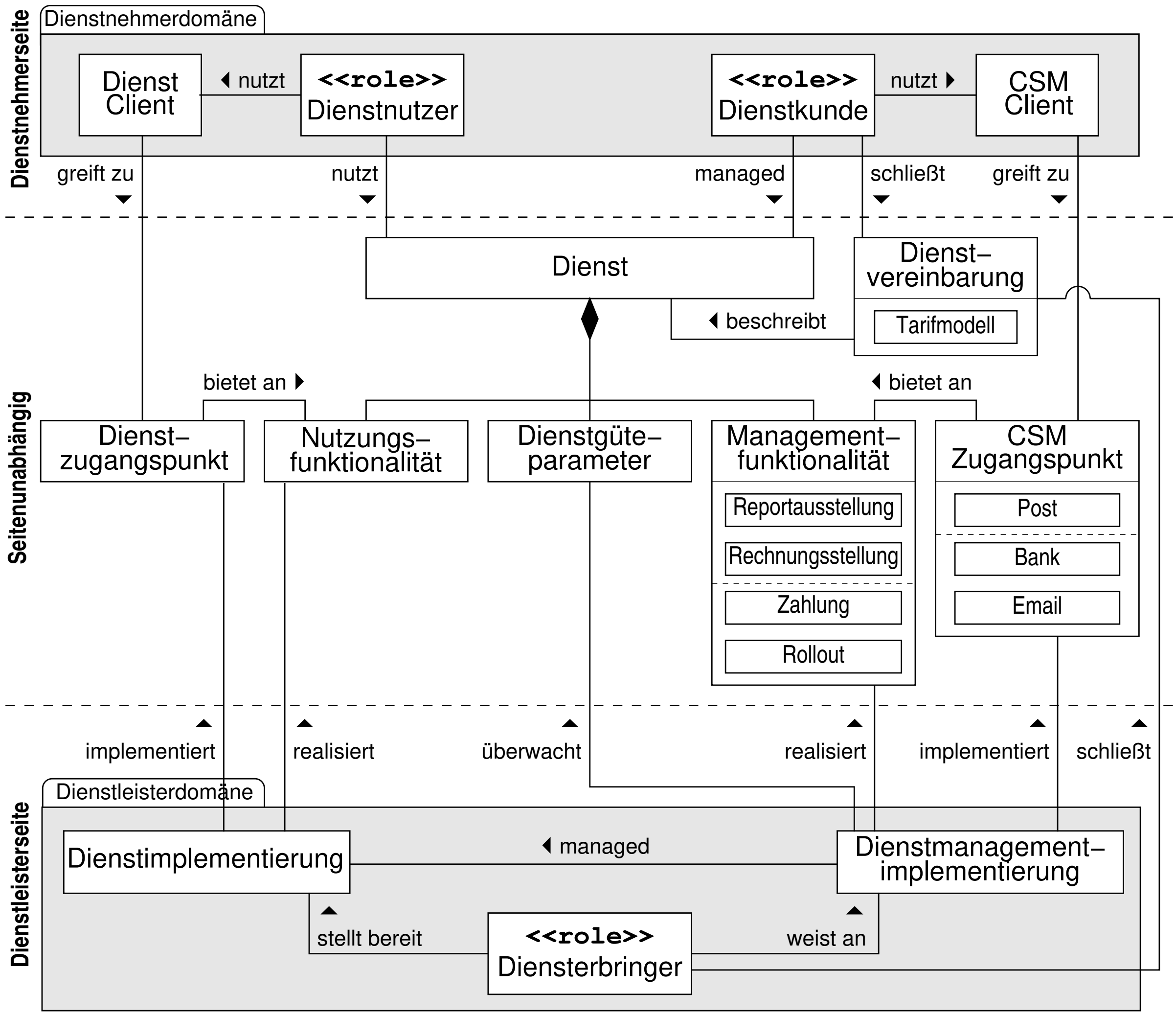

Abbildung 4.22: Abrechnungsdienstmodell: Dienstsicht

Die Managementfunktionalität beschreibt diejenigen Eigenschaften des Dienstes, die zwar für den erfolgreichen Betrieb des Dienstes erforderlich sind, aber nicht zur Nutzungsfunktionalität des Dienstes gezählt werden. Aus Abrechnungssicht muss demnach die Managementfunktionalität eines Dienstes die Erstellung und Zusendung einer korrekten Rechnung als auch eines Reports an entsprechende Empfänger auf Dienstnehmerseite vorsehen. Desweiteren werden der Managementfunktionalität auch notwendige Interaktionen zugeordnet, die von Dienstnehmerseite initiiert werden. Hierzu zählt die Zahlung des in Rechnung gestellten Betrags durch die Dienstnehmerseite sowie die Übergabe der für den Betrieb des Dienstes erforderlichen Kundendaten. Der CSM Zugangspunkt legt fest, wie die Managementfunktionalität genutzt werden kann. Beispielhaft ist für die Report– und Rechnungszusendung die Post als CSM Zugangspunkt angegeben sowie die Bank resp. ein Bankkonto für die Zahlung des Rechnungsbetrags und eine Email–Adresse für den Austausch der Kundendaten. Die Dienstvereinbarung muss nun neben der möglichst exakten Beschreibung der Managementfunktionalität (also beispielsweise die Abrechnungszeiträume, Zahlungsintervalle, etc.) und des dazugehörigen CSM Zugangspunktes auch ein Tarifmodell vorsehen, das zur Berechnung des in Rechnung zu stellenden Betrags verwendet wird.

Im nachfolgenden Abschnitt wird auf die Bestandteile der Dienstvereinbarung näher eingegangen.

4.4.2 Vereinbarungssicht

In Abbildung 4.23 sind die abrechnungsrelevanten Bestandteile einer Dienstvereinbarung zusammen mit denjenigen Entitäten abgebildet, die die Gestaltung der Dienstvereinbarung beeinflussen.

Im Einzelnen beinhaltet eine Dienstvereinbarung aus Abrechnungssicht die folgenden Elemente:

- *Tarifmodell*
  Bezüglich des Tarifmodells müssen nach Abschnitt 4.3.4 die folgenden Bestandteile zwischen Kunde und Diensterbringer vereinbart werden:
  - *Abrechnungseinheiten und Preisfunktionen*
    Ein Tarifmodell besteht aus Preisfunktionen und Abrechnungseinheiten, über die sich die beiden Parteien einigen müssen. Beeinflußt werden hierbei die Verhandlungen von dem *Dienstleister*– und *Kundentarifschema*, die jeweils die Anforderungen der beiden Seiten an das Tarifmodell zusammenfassen. Die Tarifschemata drücken somit Rahmenbedingungen (*Constraints*) bezüglich des Tarifmodells aus, die oftmals aus Unternehmensrichtlinien abgeleitet werden. Neben dem Tarifschema wird von Dienstleisterseite die Tarifvereinbarung auch durch die Kostenprognose (siehe hierzu Abschnitt 4.3.3) beeinflußt. Je zutreffender und somit besser die Prognose ist, desto mehr Verhandlungsspielraum hat der Dienstleister.
  - *Messgenauigkeit*
    Die Preisfunktion bestimmt immer in Abhängigkeit eines Parameters x die Gebühr einer Abrechnungseinheit. Hierbei muss das Tarifmodell eine eindeutige Abbildung gewährleisten. Somit sind sowohl für die Abrechnungseinheiten als auch für zur Laufzeit zu bestimmende Wertebelegungen des Parameters x (z.B. Tageszeit, erfahrene Dienstgüte, etc.) die Messgenauigkeit festzulegen.
- *Rechnungsstellung*
  - *Art der Rechnungsstellung*
    Die Art und Weise der Rechnungsaus– und zustellung muss zwischen beiden Parteien vereinbart werden. Denkbar ist sowohl eine reine elektronische Rechnungsaus– und

 Als Buch erhältlich im Verlag Dr. Hut, München, www.dr.hut-verlag.de (ISBN 3-934767-97-4)

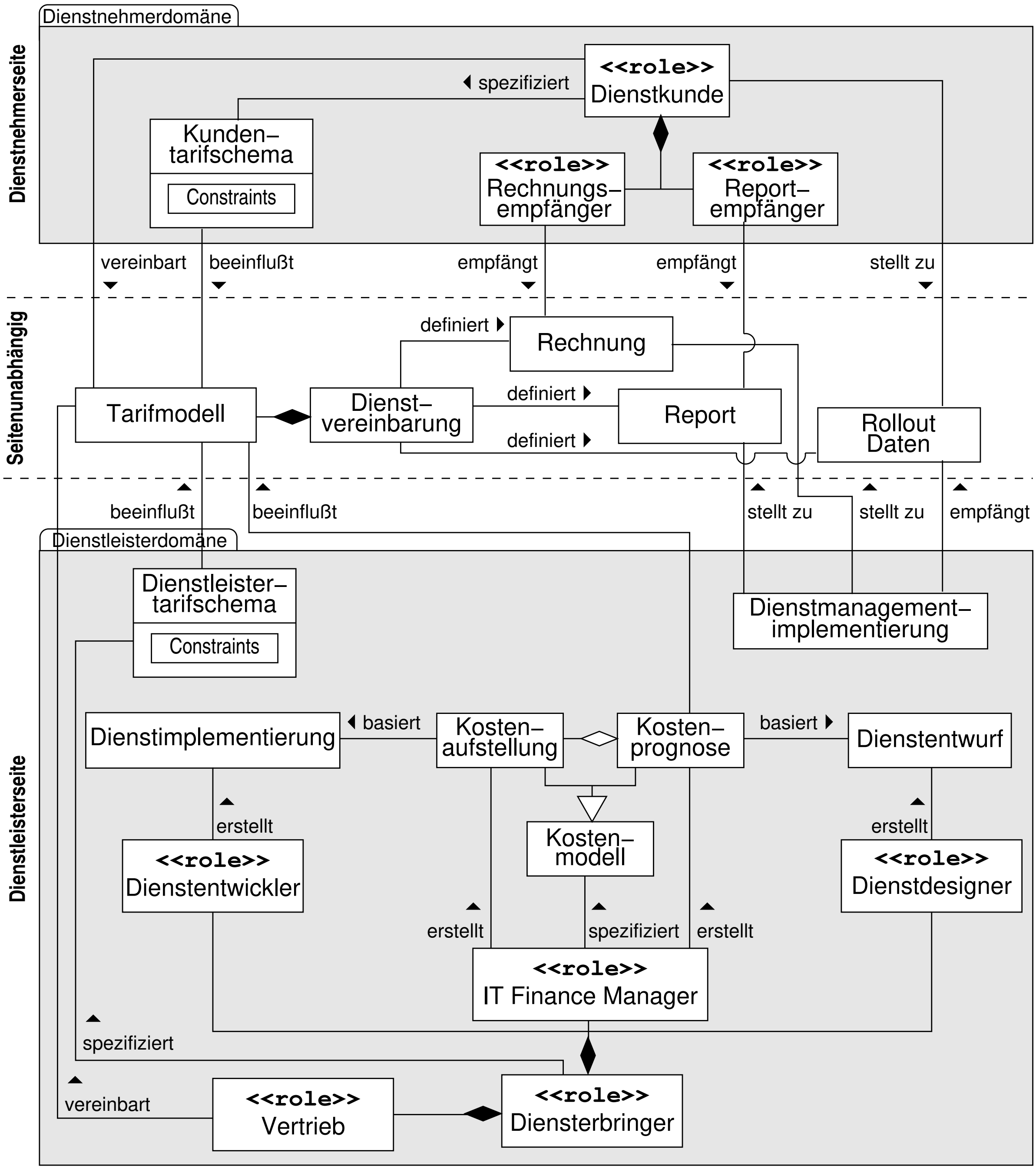

Abbildung 4.23: Abrechnungsdienstmodell: Vereinbarungssicht

zustellung (z.B. per Email, WWW Formular, etc.) als auch die klassische Variante, bei der die Rechnung gedruckt und per Post an die Empfänger verschickt wird. Ebenso wird das Zeitintervall zur Ausstellung einer Rechnung vereinbart (z.B. auf Anfrage (Hot–Billing), monatlich, etc.).

  - *Form und Inhalt der Rechnung*
    Die Form als auch der Inhalt der Rechnung sind ebenfalls Gegenstand der Verhandlungen zwischen beiden Seiten. Je nach Kundenwunsch kann der Detailgrad und die Art und Weise der Zusammenstellung der Rechnung variieren.
  - *Rechnungsempfänger*
    Für die Zustellung der Rechnung müssen die Rechnungsempfänger bekannt sein. Je nach Zustellungsart (elektronisch, per Post, etc.) müssen die hierfür notwendigen Daten (z.B. Adressen) vorliegen.
  - *Zahlungsmodalitäten mit Eskalationsmechanismen*
    Es wird auch festgelegt, wann der Kunde die in Rechnung gestellten Beträge zahlen muss sowie auf welche Weise (z.B. automatisch per Lastschrift, Überweisung, Kreditkarte, etc.). Zusätzlich werden auch Eskalationsmechanismen vereinbart, falls der Kunde die vereinbarten Zahlungsintervalle nicht einhält. Hierbei sind i.d.R. mehrere Stufen vorgesehen, die ein Spektrum vom Verschicken von Mahnungen bis zum Sperren des Dienstzugangspunktes abdecken.

- *Reportausstellung*
  Bezüglich der Reportausstellung sind die gleichen Vereinbarungen zu treffen, wie bei der Rechnungsstellung (mit Ausnahme der Zahlungsmodalitäten). D.h. es muss die *Art der Reportausstellung*, die *Form und der Inhalt eines Reports* (z.B. graphische Auswertung) und die *Reportempfänger* vereinbart werden.

- *Rollout*
  Bezogen auf den Daten–Rollout zwischen Dienstnehmer– und Dienstleisterseite sind folgende Vereinbarungen zu treffen:

  - *Art der Daten*
    Zunächst muss geklärt werden, welche Informationen von Dienstnehmerseite dem Diensterbringer zur Verfügung gestellt werden müssen, damit sowohl der Dienst als auch das Dienstmanagement in der geforderten Art und Weise betrieben werden kann. Bezogen auf das Abrechnungsmanagement können das beispielsweise Nutzerdaten, Rechnungs– und Reportempfängerdaten, Standortdaten, etc. sein.
  - *Art der Datenübertragung*
    Ebenso muss festgelegt werden, auf welche Weise die Übertragung der Rollout–Daten stattfindet. Hierbei ist eine enge oder auch lockere Kopplung von am Datenaustausch beteiligten Systemen denkbar. Sofern keine enge Kopplung (z.B. durch direkte Interaktion der beteiligten Systeme per Nutzungsschnittstellen) vorgesehen ist, sollte bereits zum Zeitpunkt des Vertragsabschlusses vereinbart werden, dass in regelmäßigen Zeitabständen ein Rollout durchgeführt wird, sofern von häufigen Änderungen der Daten in der Betriebsphase ausgegangen wird.
  - *Vereinbarung der Datenstruktur*
    Um ein Einfügen der Rollout–Daten in den Datenbestand des Diensterbringers zu ermöglichen, muss die Datenstruktur vereinbart werden.

 Als Buch erhältlich im Verlag Dr. Hut, München, www.dr.hut-verlag.de (ISBN 3-934767-97-4)

In [Schm 01] wird die Struktur von Dienstvereinbarungen analysiert und darauf basierend eine formale Darstellung für Dienstvereinbarungen entwickelt, die insbesondere Vereinbarungen bzgl. des Dienstmanagements als integralen Bestandteil enthält. Aus diesem Grund wird auf eine tiefergehende Betrachtung der Dienstvereinbarung verzichtet und auf [Schm 01] verwiesen.

In dem nun folgenden Abschnitt 4.4.3 wird die Realisierung des Abrechnungsvorgangs sowie des Abrechnungsmanagements dargestellt.

4.4.3 Realisierungssicht

In Abbildung 4.24 ist ein auf die Realisierung des Abrechnungsvorgangs fokussierter Ausschnitt des Abrechnungsdienstmodells dargestellt. Die in der Dienstvereinbarung festgelegte Managementfunktionalität des Dienstes beeinflusst die Dienstmanagementimplementierung und damit die Realisierung und das Management des Abrechnungsvorgangs, die integrale Bestandteile der Dienstmanagementimplementierung sind. Im Abrechnungsdienstmodell wird analog zu den Ausführungen in Abschnitt 4.2 die *Abrechnungsprozessimplementierung* explizit getrennt von der *Abrechnungsmanagementimplementierung* dargestellt. Bezüglich der Abrechnungsprozessimplementierung bestimmt die Dienstvereinbarung die Auswahl der Einzelkomponenten, welche die in Abschnitt 4.3 beschriebenen Abrechnungsteilprozesse umsetzen. Im Gegensatz dazu werden bezüglich der Abrechnungsmanagementimplementierung aus der Dienstvereinbarung Managementanweisungen abgeleitet, welche die Abrechnungsprozessimplementierung in der Art und Weise steuern und überwachen, dass die Vereinbarungen bzgl. der Abrechnung erfüllt werden.

Beispiel: Die Wahl eines konkreten Billing Systems ist von der vereinbarten Rechnungsstellungsfunktionalität abhängig: Ist z.B. Hot–Billing vereinbart, so kommen nur Billing Systeme in Frage, die diese Funktionalität in Zusammenarbeit mit den anderen an der Abrechnung beteiligten Komponenten auch tatsächlich unterstützen. Die Abrechnungsmanagementimplementierung würde in diesem Fall bei einem konkret ausgewählten Billing System eine Konfiguration dieser Komponente vorsehen, so dass nur die in der Dienstvereinbarung vorgesehenen Personen auf die Rechnungsdaten im Rahmen des Hot–Billings zugreifen können.

Gesondert erwähnt werden muss in diesem Zusammenhang die Wahl der Messkomponenten. Da eine Messkomponente die im Tarifmodell enthaltenden Abrechnungseinheiten während der Dienstnutzung erfassen muss, damit eine korrekte Gebührenberechnung ermöglicht wird, bestimmt insbesondere das vereinbarte Tarifmodell die Wahl und Implementierung der Messkomponente. Da der Sensor Aspekte der Dienstnutzung, wie z.B. die Dienstqualität, das in Anspruch genommene Dienstnutzungsvolumen, etc. misst, ist dieser insbesondere von der tatsächlichen Dienstimplementierung abhängig (siehe hierzu Abschnitt 4.3.5).

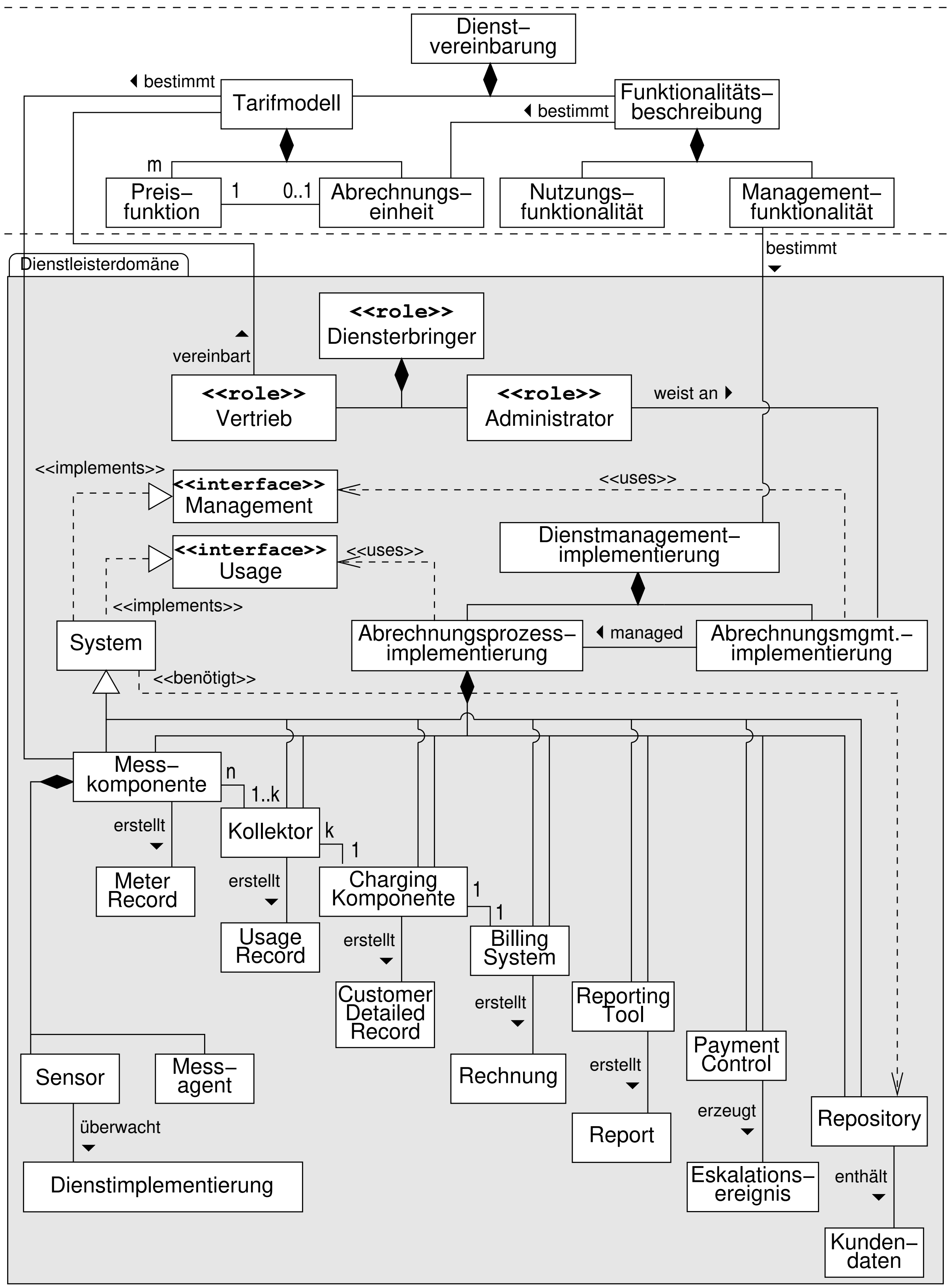

Abbildung 4.24: Abrechnungsdienstmodell: Realisierungssicht

 Als Buch erhältlich im Verlag Dr. Hut, München, www.dr.hut-verlag.de (ISBN 3-934767-97-4)

Um das Zusammenspiel der an der Abrechnung beteiligten Komponenten im Kontext der Abrechnungsprozessimplementierung und der Abrechnungsmanagementimplementierung nachvollziehen zu können, ist es wichtig, die Nutzungsschnittstelle einer beteiligten Komponente von deren Managementschnittstelle zu unterscheiden. Während die Abrechnungsprozessimplementierung durch Kopplung der Nutzungsschnittstellen realisiert wird, werden prinzipiell zunächst von der Abrechnungsmanagementimplementierung ausschließlich die Managementschnittstellen verwendet. Damit ist die Mächtigkeit des durchsetzbaren Managements bezüglich der Überwachung und Steuerung der Komponenten zunächst durch die an den Managementschnittstellen der Komponenten angebotene (Management–)Funktionalität beschränkt. Da allerdings die Komponenten über die Nutzungsschnittstellen miteinander agieren, läßt sich einerseits die Überwachungsgranularität dadurch erhöhen, indem das Management zusätzlich die Kommunikation zwischen den Komponenten überwacht, und andererseits läßt sich die Steuerungsgranularität dadurch erhöhen, indem die Abrechnungsmanagementanwendung in die Kommunikation aktiv eingreift. Diese Erkenntnis wird in den noch kommenden Kapiteln beim Design der Managementanwendung aufgegriffen.

Damit sind die drei Teile des Abrechnungsdienstmodells beschrieben, das eine statische Darstellung der an der Abrechnung und deren Management beteiligten Entitäten und Rollen repräsentiert.

4.5 Zusammenfassung

In diesem Kapitel wurde eine umfangreiche Analyse der dynamischen Aspekte der Abrechnung und des Abrechnungsmanagements durchgeführt, da eine derartige Untersuchung bisher, insbesondere für die in dieser Arbeit fokussierten Großkundenszenarios, fehlt. Hierzu wurde zunächst in strukturierter Art und Weise ein Prozessmodell entworfen (Abschnitt 4.2), indem entlang des Dienstlebenszyklus die für den Abrechnungsvorgang relevanten Teilprozesse identifiziert wurden. Anschließend wurden auf Basis der in dieser Arbeit durchgeführten Szenarioanalyse (siehe Kapitel 2) und in Teilen bereits bestehender Spezifikationen in diesem Bereich die Teilprozesse auf einzelne Aktivitäten, Ein–/Ausgabeentitäten sowie beteiligten Akteure untersucht und jeweils in eigenen Diagrammen dargestellt (Abschnitt 4.3). Neben der reinen Visualisierung der Teilprozesse, wurden diese ausführlich beschrieben. Sofern während der Ausführung eines Teilprozesses Entscheidungen zu treffen sind, wurden, wo möglich, allgemeine Kriterien zur Entscheidungsfindung angegeben. Somit ist es gelungen, eine detaillierte, aber dennoch generische Beschreibung der für die dienst– und kundenorientierte Abrechnung notwendigen Schritte zu erstellen. Anschließend wurde basierend auf dieser Prozessanalyse ein Abrechnungsdienstmodell als Erweiterung des generischen MNM Dienstmodells entwickelt (Abschnitt 4.4), das alle am Abrechnungsvorgang beteiligten Entitäten in Beziehung zueinander setzt. Damit weist das Abrechnungsdienstmodell direkte und indirekte Abhängigkeiten zwischen Entitäten bei ausgeblendeter Zeitdimension auf.

Das Prozessmodell sowie das Abrechnungsdienstmodell dienen als Grundlage für die zu entwickelnde Managementlösung in den nachfolgenden Kapiteln dieser Arbeit. Das Prozessmodell wird zur Festlegung der notwendigen Managementanweisungen benötigt, während das Abrechnungsdienstmodell u.a. als Grundlage für die Spezifikation von Managementschnittstellen verwendet wird.

 Als Buch erhältlich im Verlag Dr. Hut, München, www.dr.hut-verlag.de (ISBN 3-934767-97-4)

Kapitel 5

Anwendung policy–basierter Konzepte zur Steuerung des Abrechnungsprozesses

In diesem Kapitel wird der neu entwickelte Ansatz vorgestellt, der die aus dem dienst– und kundenorientierten Abrechnungsmanagement resultierenden Anforderungen erfüllt. Zunächst wird hierzu die grundsätzliche Lösungsidee skizziert, die eine Anwendung von policy–basierten Konzepten in Kombination mit einer strikt prozessorientierten Sicht auf den Abrechnungsvorgang und das Abrechnungsmanagement vorsieht. Da der grundsätzliche Ansatz allgemein auf andere Managementfunktionsbereiche anwendbar ist und nicht auf das dienstorientierte Abrechnungsmanagement beschränkt ist, wird eine Policy–Sprache für ein allgemeines, prozessorientiertes IT Management entworfen, welches auf Basis einer umfangreichen Analyse der dafür notwendigen Ausdrucksmächtigkeit durchgeführt wird. Anschließend wird eine im Kontext des prozessorientierten IT Managements allgemeine Methodik für die Spezifikation von Policies auf Grundlage von Prozess– und Dienstmodellen entwickelt. Basierend auf dieser Methodik werden die sowohl im vorhergehendem Kapitel spezifizierten Teilprozessmodelle für den Abrechnungsvorgang als auch das Abrechnungsdienstmodell für die Festlegung von Policy–Schablonen für das Abrechnungsmanagement verwendet. Schließlich wird der objektorientierte Entwurf eines policy–basierten, prozessorientierten Managementsystems vorgestellt. Abgerundet wird dieses Kapitel mit der Beschreibung einiger Beispielpolicies für das Abrechnungsmanagement.

5.1 Einführung

Aus der bisherigen Analyse folgernd muss insgesamt gesehen ein dienst– und kundenorientiertes Abrechnungsmanagement die Abrechnung entlang des vollständigen Dienstlebenszyklus, wie es in Kapitel 4 vorgestellt wurde, unterstützen. Demnach muss jeder Teilprozess jeder einzelnen Lebenszyklusphase geeignet gesteuert und überwacht werden. In der folgenden Aufzählung werden wiederholend diejenigen Gegebenheiten genannt, die v.a. bisher die Entwicklung einer Managementlösung für das dienst– und kundenorientierte Abrechnungsmanagement erschwerten:

- *Heterogene Implementierung des Abrechnungsprozesses*
  Heutzutage werden unterschiedliche Hard-/Software– und Anwendungskomponenten miteinander kombiniert, um die Abrechnung eines Dienstes zu ermöglichen. So wird beispielsweise die Nutzungserfassung von speziellen Messkomponenten in Kombination mit einer separaten Aggregations– und Kollektorsoftware realisiert. Die Berechnung der in Rechnung zu stellenden Gebühren wird oftmals von einer betriebswirtschaftlichen Standardsoftware durchgeführt. Für die tatsächliche Ausstellung von Rechnungen kann zudem ein Billing System eingesetzt werden. Wie in Abschnitt 2.2 gezeigt wurde, erfolgt die Auswahl der eingesetzten Werkzeuge sowie deren Kombination jeweils maßgeschneidert für ein gegebenes Szenario. Da sich bisher keine standardisierten Schnittstellen zwischen den einzelnen Komponenten zum Austausch von Daten– und Steuerungsinformationen etabliert haben, müssen entweder proprietäre Schnittstellen unterstützt werden oder Gateways implementiert werden. Dies ist insbesondere in der Implementierungs– sowie Installations– & Testphase für das Abrechnungsmanagement relevant.

- *Keine einheitlichen/standardisierten Managementschnittstellen*
  Trotz einiger Standardisierungsbemühungen im Bereich des Abrechnungsmanagements (siehe hierzu Abschnitt 3.2), haben sich bisher nur unzureichend Standards für Managementschnittstellen von Komponenten durchgesetzt, die an der Abrechnung beteiligt sind. Jede Komponente besitzt somit eine eigene, meist proprietäre Managementschnittstelle. Wird aufgrund geänderter Anforderungen eine Komponente durch eine andere ausgetauscht, so muss das Management der ausgetauschten Komponente ebenso angepasst, wenn nicht sogar vollständig neu implementiert werden.

- *Kein integriertes, ganzheitliches Abrechnungsmanagement*
  Bisher werden die an der Abrechnung beteiligten Komponenten isoliert ohne expliziten Bezug zu anderen Abrechnungskomponenten gesteuert. Dies ist z.T. auch eine Folge der im vorhergehenden Punkt erwähnten fehlenden standardisierten Managementschnittstellen. Damit können Inkonsistenzen insbesondere bei Konfigurationen entstehen, wenn Änderungen nur bei einer Komponente durchgeführt werden, die aber auch Auswirkungen auf andere Komponenten haben.

- *Unzureichende Automatisierung von Managementvorgängen*
  Wie in Abschnitt 2.2.2 erläutert, ergeben sich insbesondere in Großkundenszenarios oftmals Änderungen auf Kunden– als auch auf Providerseite, die Auswirkungen auf das Abrechnungsmanagement haben. Hierbei wäre ein hoher Automatisierungsgrad bei der Durchführung von wiederkehrenden Managementvorgängen wünschenswert. Dies würde zudem die Fehlerwahrscheinlichkeit senken. Allerdings fehlt heutigen Managementwerkzeugen eine derartige Unterstützung.

 Als Buch erhältlich im Verlag Dr. Hut, München, www.dr.hut-verlag.de (ISBN 3-934767-97-4)

Demnach ist ein integriertes, flexibles und möglichst automatisiertes Abrechnungsmanagement gefordert, das einerseits an unterschiedliche Szenarios und damit an unterschiedliche Kunden–, User– und Dienstprofile anpassungsfähig ist und andererseits auch die auftretende Änderungsdynamik unterstützt. Das Abrechnungsmanagementsystem muss demnach diese Flexibilität sowohl in der Architektur als auch im umgesetzten Managementansatz, der im Folgenden vorgestellt wird, unterstützen.

5.2 Lösungsidee und Vorgehensmodell

Die prinzipielle Lösungsidee basiert darauf, nicht mehr isoliert und unabhängig voneinander die einzelnen an der Abrechnung beteiligten Komponenten zu managen, sondern den Abrechnungsvorgang als solchen aus *Prozesssicht*, zunächst ungeachtet von den realisierenden Komponenten, zu steuern. Durch Anwendung der prozessorientierten Sicht ist es insbesondere möglich, auch die dynamischen Aspekte der Abrechnung und damit des Abrechnungsmanagements miteinzubeziehen, welches den bisher entwickelten Ansätzen fehlt. Zusammenfassend aus dem vorhergehenden Kapitel 4 resultiert die Dynamik einerseits aus den unterschiedlichen, szenarioabhängigen Variationen der Teilprozesse und andererseits durch Change-Managementaktivitäten, die Managementaktionen initiieren können, die u.U. bereits in früheren Phasen des Lebenszyklus aufgetreten sind. Um nun ein derartig prozessorientiertes Management tatsächlich zu realisieren und es damit nicht nur als eine reine Darstellungsweise für die Managementaufgabe zu belassen, wird der Einsatz von *policy–basierten Konzepten* vorgeschlagen.

Somit besteht die Lösungsidee darin, jeden in Kapitel 4 identifizierten Abrechnungsteilprozess durch eine geeignete (Abrechnungs–)Managementpolicy zu steuern und zu überwachen. Durch den Einsatz von Managementpolicies und den damit verbundenen, bereits entwickelten Konzepten in Kombination mit einer prozessorientierten Sicht ergeben sich bereits auf konzeptioneller Ebene mehrere positive Eigenschaften. Im Nachfolgenden wird auf diese Eigenschaften jeweils kurz eingegangen und es werden zudem bereits erfüllte Anforderungen (siehe hierzu den in Abschnitt 2.3.5 erstellten Anforderungskatalog) genannt:

- *Abstraktion wesentlicher Bestandteil policy–basierter Managementkonzepte*
  Ein Grundprinzip policy–basierter Managementkonzepte ist die Abstraktion von unwichtigen Details und die Konzentration auf die relevanten Aspekte in einem Managementbereich. Damit soll einerseits die Erstellung von Managementanweisungen wesentlich erleichtert werden als auch andererseits die Spezifikation von Fehlern vermieden werden. Diese Eigenschaft ist zudem Grundlage für die Erfüllung der im Anforderungskatalog unter den Punkten MGT 1, MGT 2 und MGT 3 spezifizierten Anforderungen.

- *Einheitliche Schnittstelle zur Spezifikation von Managementanweisungen*

Das policy–basierte Management bietet dem Manager in Form der Policy–Sprache eine einheitliche und flexible Schnittstelle zur Spezifikation von Managementanweisungen. Damit ist ein uneinheitliches und umständliches Management, wie es bisher der Fall ist, der an der Abrechnung beteiligten Komponenten aufgrund von unterschiedlichen Managementschnittstellen ausgeschlossen. Zusätzlich bildet dies auch die Basis dafür, die Komponenten im Zusammenspiel und nicht mehr isoliert zu managen. Somit werden die unter Punkt MGT 1 und Punkt MGT 2 formulierten Anforderungen erfüllt.

- *Explizite Speicherung von Managementwissen*
  Wie in Abschnitt 2.2 bei der Darstellung eines Großkundenszenarios erklärt wurde, werden bisher viele Managementaktivitäten manuell durch die verantwortlichen Administratoren durchgeführt. Damit findet keine Wissensverteilung über das Management einer bestimmten Komponente statt, sondern das Expertenwissen bleibt bei den jeweilig verantwortlichen Administratoren. Dies kann insbesondere bei Ausfall eines Administrators kritisch sein. Da ein Ursprung von policy–basierten Konzepten bei wissensbasierten Expertensystemen liegt, kann damit vorhandenes Managementwissen explizit gespeichert und weiterverarbeitet werden. Diese Eigenschaft ist Grundlage für die im nächsten Punkt genannte Eigenschaft und bildet damit die Basis für die Erfüllung der Anforderung MGT 3.

- *Automatisierung von Managementaktivitäten*
  Ein Manager muss lediglich Policies in der gegebenen Policy–Sprache spezifizieren. Die Durchsetzung der Policies wird automatisch durch entsprechende *Policy–Enforcer* durchgeführt. Damit wird die im Anforderungskatalog unter Punkt MGT 3 spezifizierte Anforderung erfüllt.

- *Mächtigkeit des Managements in direkter Relation zur Ausdrucksmächtigkeit der Policy–Sprache*
  Die Ausdrucksmächtigkeit der Policy–Sprache bestimmt die Mächtigkeit des damit durchsetzbaren Managements. Um die Komplexität von Policy–Sprachen zu beschränken, aber dennoch eine dem Anwendungsfall genügende Ausdrucksmächtigkeit zu gewähren, bietet sich die Möglichkeit, allgemeine, bereits entwickelte Policy–Sprachen anzupassen.

Es gilt festzustellen, dass der grundsätzliche Ansatz, ein prozessorientiertes Management durch Anwendung von policy–basierten Konzepten durchzusetzen, *allgemein* für alle Funktionsbereiche des Managements anwendbar ist und somit zunächst *nicht* alleinig auf das Abrechnungsmanagement beschränkt ist. Im weiteren Vorgehen der Arbeit wird deswegen versucht, so weit wie möglich generisch zu bleiben und nur in den absolut notwendigen Teilen der zu entwickelten Managementlösung abrechnungsmanagement–spezifisch zu werden.

 Als Buch erhältlich im Verlag Dr. Hut, München, www.dr.hut-verlag.de (ISBN 3-934767-97-4)

Um nun policy–basierte Managementkonzepte auf ein prozessorientiertes (Abrechnungs–)Management anzuwenden, wird das in Abbildung 5.1 visualisierte Vorgehen gewählt. Da die *Managementarchitektur* festlegt, wie das grundsätzliche Zusammenspiel zwischen den einzelnen, am Management beteiligten Entitäten funktioniert, wird in einem ersten Schritt auf Basis des in Kapitel 2 entwickelten Anforderungskatalogs ein Grobentwurf der policy–basierten Managementarchitektur entwickelt. Es wird demnach spezifiziert, wie der zu managende Prozess (im vorliegenden Fall: Abrechnungsprozess) aus Managementsicht dargestellt wird, so dass ein policy–basiertes Management, wie zu Beginn dieses Abschnitts beschrieben, möglich ist. Somit bestimmt die Managementarchitektur auch, auf welche Weise die in einer Policy–Sprache formulierten Managementanweisungen ausgeführt werden. Insbesondere wird demnach die Policy–Evaluierung und –Durchsetzung in Bezug zu den zu managenden Entitäten bestimmt und damit auch die Zielsprache(n) der vom Policy–Interpreter zu übersetzenden Managementanweisungen.

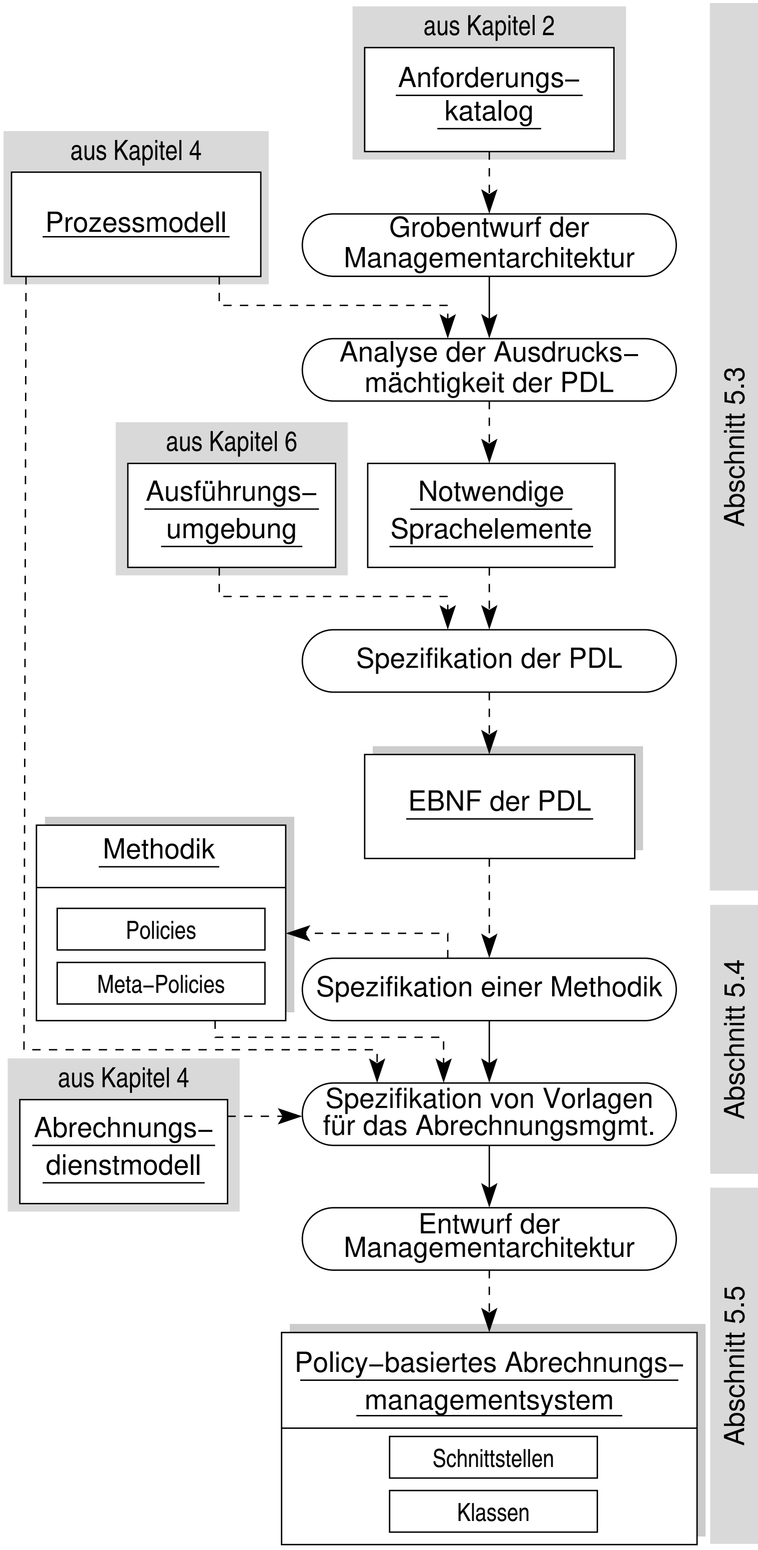

Abbildung 5.1: Vorgehensmodell und Ergebnisse des Kapitels 5

Der Grobentwurf der policy–basierten Managementarchitektur wird zusammen mit den grundsätzlichen Elementen des in Kapitel 4 entwickelten Prozessmodells dazu verwendet, die notwendige *Ausdrucksmächtigkeit* der Policy–Sprache zu untersuchen und die hierfür obligatorischen Sprachelemente zu identifizieren. Im nächsten Schritt werden die hierbei gewonnenen Analyseergebnisse dazu verwendet, um die *Grammatik* der Policy–Sprache (*Policy Description Language (PDL)*) für ein allgemeines prozessorientiertes IT Management zu spezifizieren. Da eine in der gegebenen PDL definierte Policy letztendlich auf zu managenden Ressourcen durchgesetzt werden muss, muss für die PDL–Spezifikation bereits die Ausführungsumgebung miteinbezogen werden, die allerdings erst im nächsten Kapitel 6 näher betrachtet wird.

In Anschluss an die Spezifikation der Grammatik der PDL wird eine allgemeine *Methodik* zur Spezifikation von Policies und Meta–Policies für das prozessorientierte IT Management festgelegt. Unter Zuhilfenahme des in Kapitel 4 entwickelten Abrechnungsdienstmodells und der entwickelten Teilprozessmodelle werden zusätzlich auf Basis der Methodik Policy–Schablonen speziell für das Abrechnungsmanagement angegeben. Sind entsprechende Teilprozess– und Dienstmodelle für andere Funktionsbereiche des Managements[1] vorhanden, so können in gleicher Art und Weise Policy–Schablonen festgelegt werden.

Zum Schluss erfolgt auf Basis der in diesem Kapitel gewonnenen Erkenntnisse der objektorientierte Entwurf eines policy–basierten Abrechnungsmanagementsystems. Auch hierbei wurde auf einen möglichst generischen, d.h. nicht auf das Abrechnungsmanagement beschränkten, Entwurf geachtet.

Insgesamt ist die entwickelte Policy–Sprache, die spezifizierte Methodik und der überwiegende Anteil des entworfenen Managementsystems allgemein für ein prozessorientiertes IT Management einsetzbar und nicht auf das Abrechnungsmanagement beschränkt. Der Abrechnungsvorgang sowie das Abrechnungsmanagement dienen im Rahmen dieses Kapitels v.a. zur vertiefenden Darstellung des entworfenen Konzepts.

5.3 Der Einsatz von Policies im prozessorientierten IT Management

Nach einem Überblick über den Status Quo im policy–basierten Management in Abschnitt 5.3.1, wird in Abschnitt 5.3.2 ein erster Grobentwurf der policy–basierten Managementarchitektur beschrieben. Anschließend werden in Abschnitt 5.3.3 Anforderungen an eine Policy–Sprache für das dienst– und prozessorientierte IT Management analysiert und grundlegende Sprachbausteine identifiziert. Darauf basierend wird in Abschnitt 5.3.4 eine Grammatik für die Policy–Sprache entworfen.

[1] Nach OSI wären das das Leistungs–, Fehler–, Sicherheits– und Konfigurationsmanagement.

 Als Buch erhältlich im Verlag Dr. Hut, München, www.dr.hut-verlag.de (ISBN 3-934767-97-4)

5.3.1 Status Quo im policy–basierten Management: Überblick

In diesem Abschnitt werden bisherig entwickelte Ansätze auf dem Gebiet des policy–basierten Managements vorgestellt. Aufgrund der großen Anzahl von Arbeiten in diesem Bereich konzentriert sich die nachfolgende Beschreibung auf die bedeutendsten Arbeiten.

Begriffe und Definitionen

Es können zahlreiche Definitionen für eine Policy in der Literatur gefunden werden, die sich v.a. dadurch unterscheiden, dass diese das grundsätzliche *Konzept*, welches zur Beschreibung einer Policy eingesetzt wird und das *Einsatzgebiet*, in dem Policies verwendet werden, miteinbeziehen. Insgesamt gesehen spiegelt jedoch die folgende, in [Dami 02] spezifizierte und allgemeingehaltene Definition den momentanen Konsens im Bereich des Managements bzgl. dieser Thematik wider:

> *Eine Policy ist die aus Managementzielen abgeleitete persistente, deklarative Spezifikation des Verhaltens eines Systems.*

In ihrer Bedeutung ähnliche Definitionen sind u.a. auch in [Wies 95, Koch 97, Mari 97, RFC 3060] zu finden. Mit Hilfe einer Policy soll demnach in deklarativer Art und Weise beschrieben werden, *welches* Verhalten von einem gegebenen System gewünscht wird, jedoch nicht, *wie* dieses Verhalten, im Sinne einer prozeduralen Beschreibung von einzelnen Programmanweisungen, erreicht wird. Somit ist es Ziel, dass der Manager nur das allernotwendigste spezifiziert und alle restlichen, notwendigen Schritte möglichst automatisiert durch ein geeignetes *Werkzeug* ausgeführt werden. Folglich soll die Komplexität der Durchführung bzw. Durchsetzung von Managementanweisungen vom Administrator in ein entsprechendes Policy–Werkzeug verlagert werden. Durch dieses generelle Prinzip wird ein höherer Abstraktionsgrad erreicht, wodurch zunächst primär eine erhebliche Erleichterung und Vereinfachung der Durchführung von Managementaufgaben geschaffen wird. Diese durch Policies gewünschte Erleichterung sowie Vereinfachung des Managements wird zudem als unabdingbare Voraussetzung angesehen [Wies 94], um ein immer komplexer werdendes Umfeld, bestehend aus mehreren tausend Managementobjekten, aus Sicht des Managements überhaupt erst handhabbar zu machen.

Jede Policy kann als *regelähnlicher Ausdruck* angesehen werden, der in einer gegebenen *Policy–Sprache*, welche eine eindeutige Grammatik vorweist, in der dafür vorgesehenen *Policy–Notation*, z.B. als Text, Graph, etc., vom Manager spezifiziert wird. Die Architektur eines Policy–Werkzeugs weist folgende Bestandteile auf:

- *Policy–Verwaltung*
  Die Policy-Verwaltung sieht neben der persistenten Speicherung von Policies auch die u.U. notwendige Verteilung von Policies auf unterschiedliche *Points of Decision (POD)* vor.

- *Point of Decision (POD)*
  Der POD wird als derjenige Teil der policy–basierten Managementarchitektur angesehen, der die in einem gegebenen Kontext relevanten Policies findet, diese evaluiert und schließlich im positiven Fall die Ausführung der in der Policy spezifizierten Aktionen anstößt. Zur Evaluierung einer Policy wird ein *Policy–Interpreter* verwendet. Hierbei bildet der *Policy–Parser* einen wichtigen Bestandteil des Policy–Interpreters.

- *Point of Execution (POE)*
  Die spezifizierten Policy–Aktionen werden tatsächlich vom POE ausgeführt, der sich in aller Regel in der Nähe der zu managenden Entität befindet.

Grundsätzlich können die eben erwähnten drei Teile zentral oder auch verteilt in unterschiedlichen Komponenten realisiert werden. Bei einer nicht zentralen Realisierung tritt allerdings das Problem der korrekten Verteilung von Policies auf unterschiedliche PODs auf. In [DLSD 01] wird hierfür ein Modell entwickelt.

Wie schon kurz zu Beginn dieses Abschnitts erwähnt, existieren mehrere Ansätze im policy–basierten Management, die sich v.a. in ihrem Einsatzgebiet und dem Sprachkonzept zur Formulierung einer Policy unterscheiden. Da mit einem hohen Abstraktionsniveau und einer hohen Vielfalt der zu unterstützenden Einsatzszenarios die Komplexität des Policy–Werkzeugs steigt, muss beim Design eines policy–basierten Managementwerkzeugs ein Kompromiss zwischen Werkzeugkomplexität auf der einen Seite sowie Implementierungstransparenz und Abstraktionsniveau auf der anderen Seite gefunden werden. Aus diesem Grund existieren für unterschiedliche Einsatzgebiete auch verschiedene, jeweils für das fokussierte Einsatzgebiet optimierte Policy–Ansätze und –Konzepte, die im Nachfolgenden exemplarisch erklärt werden.

Policy–Konzepte für das verteilte, heterogene Netz– und Systemmanagement

Ein inhärentes Problem im integrierten Netz– und Systemmanagement ist zum einen die hohe Anzahl der zu managenden Entitäten (in diesem Fall der Koppelemente und Endsysteme) und zum anderen deren hoher Grad an Verteilung und Heterogenität. Seit geraumer Zeit versucht man nun, diesem Problem durch Anwendung policy–basierter Konzepte Herr zu werden (vgl. [KKK 96, Mari 97]). Insbesondere wird versucht, die bereits beschriebene, in policy–basierten Konzepten enthaltene Abstraktion zur Verschattung sowohl der Heterogenität als auch der Verteilung zu nutzen. Ebenso soll die einheitliche bzw. konsistente Konfiguration von einer Vielzahl von verteilten Netz– und Systemkomponenten durch eine im Vergleich geringe Anzahl von Policies ermöglicht werden. Bezogen auf die sonst übliche, durch Setzen von Konfigurationsparametern, *statische* Konfiguration, ermöglicht der Einsatz von Policies eine dem Zustand einer Komponente angepasste *dynamische* Konfiguration.

Als Beispiel können für das Netzmanagement die Bestrebungen der *Internet Engineering Task Force (IETF)* genannt werden, die insbesondere den Einsatz von Konfigurationsmanagementpolicies im Bereich des Dienstgütemanagements in Zusammenhang mit

 Als Buch erhältlich im Verlag Dr. Hut, München, www.dr.hut-verlag.de (ISBN 3-934767-97-4)

DiffServ/IntServ [RFC 2749, RFC 2750, RFC 3084] untersuchen. Hierbei sieht man einfache *if–then–else*–Ausdrücke zur Policy–Formulierung vor, die immer dann ausgewertet werden, wenn beispielsweise eine Nutzeranfrage, ein Flow, etc. von einer zu managenden Komponente empfangen wird. Bei der Auswertung wird überprüft, welche Policy zutrifft (*if*–Teil), um daraufhin die spezifizierten Aktionen (*then*– bzw. *else*–Teil) auszuführen. Bei den auszuführenden Aktionen handelt es sich beispielsweise darum, IP–Pakete einer bestimmten Dienstgüteklasse zuzuordnen, Ressourcenreservierungen durchzuführen, das Aufzeichnen von Datenverkehr anzustoßen, etc.

Im Bereich des Systemmanagements hat sich der deklarative Ansatz „Policy–Objekt mit Attributen" in zahlreichen Arbeiten [Heil 98a, Mari 97, KKK 96, Koch 97] durchgesetzt. Hierbei werden innerhalb einer Policy neben den auszuführenden Aktionen auch Bedingungen für die Anwendbarkeit der Policy, die Zielobjekte der Aktionen sowie die ausführenden Objekte bestimmt. Desweiteren kann eine Policy mit weiteren Attributen versehen werden, wie z.B. die explizite Angabe eines Ereignisses, zu dem die Policy evaluiert werden soll.

Gesondert erwähnt werden soll an dieser Stelle die von Prof. Sloman am Imperial College London geleitete Forschergruppe[2], die seit mehr als 10 Jahren intensiv das policy–basierte Management verteilter Systeme untersucht. Mit *Ponder* [DDLS 00] wurden zahlreiche Spracherweiterungen für deklarative Sprachkonzepte entwickelt. Eine herausragende Weiterentwicklung bildet hierbei eine automatische Konflikterkennung und teilweise auch Konfliktlösung zwischen mehreren Sicherheitsmanagement–Policies auf Basis einer Syntaxanalyse. Auf eine tiefergehende Betrachtung der Sprache wird allerdings an dieser Stelle verzichtet.

In diesem Zusammenhang ist ebenfalls die in [DDGH 02] veröffentlichte Arbeit interessant, die u.a. eine Bewertung der prominentesten Policy–Sprachen bzgl. deren Einsatzgebietes durchführt.

Policy–Konzepte im Bereich des Sicherheitsmanagements

Ein weiteres Gebiet, in dem der Einsatz von Policies intensiv untersucht wurde, ist das Sicherheitsmanagement. Insbesondere im Bereich des Role–based Access Controls (RBAC), das sich mit der Rechtevergabe auf Basis von Rollen beschäftigt, können zahlreiche Arbeiten gefunden werden. Hier wurde insbesondere der Einsatz von Constraint– und Logik–basierten Ausdrücken zur Formulierung von Autorisierungspolicies untersucht. Auf eine tiefergehende Beschreibung wird an dieser Stelle verzichtet und auf [Dami 02] verwiesen, das eine detaillierte Darstellung aktueller Arbeiten in diesem Bereich enthält.

Bemühungen von Standardisierungs– und Industriegremien

Es beschäftigen sich insbesondere die Distributed Management Task Force (DMTF) und die Internet Engineering Task Force (IETF) mit der Spezifikation von Standards im Bereich des

[2] `http://www-dse.doc.ic.ac.uk/Research/policies/index.shtml`

policy–basierten Managements. Im Folgenden werden die wichtigsten Arbeiten der beiden Gremien kurz vorgestellt.

Distributed Management Task Force (DMTF) Ein wesentlicher Schwerpunkt der DMTF[3] liegt bei der Definition eines Management–Informationsmodells, dem sog. *Common Information Model (CIM)*. Bei CIM handelt es sich um ein objektorientiertes Informationsmodell, das die Integration von unterschiedlichen, bereits spezifizierten Informationsmodellen ermöglichen soll. Weiteres Ziel ist es einen Übergang von datentyporientierten Informationsmodellen, wie es beispielsweise das des Internet Managements ist, in ein objektorientiertes Modell zu ermöglichen, um damit die typischen Vorteile, wie bspw. Wiederverwendung durch Vererbungsmechanismen, zu nutzen. Im Rahmen der Entwicklung einer Vielzahl von spezifischen Informationsmodellen, beschäftigt sich insbesondere die *Policy Working Group* der DMTF mit der Definition eines Policy–Informationsmodells, dem sog. *CIM Core Policy Model* [DSP 0108], das mittlerweile in der Version 2.7 vorliegt. Das CIM Policy Model stellt die einzelnen Bestandteile einer Policy, wie beispielsweise eine Aktion, eine Bedingung, etc. als generische Objekte in Form eines Klassendiagramms in Beziehung zueinander. Auch wenn das CIM Policy Model *keine* Policy–Sprache definiert und damit auch keinerlei Festlegungen bzgl. der Policy–Deklaration und –Interpretation macht, ist ein einheitliches Informationsmodell Voraussetzung für die Vereinheitlichung der Policy–Verwaltung. Das heißt, dass CIM–konforme Policies, die in einer beliebigen Sprache spezifiziert sind, von beliebigen CIM–konformen Verwaltungsstrukturen verwaltet werden können. Da somit das CIM Policy Model auch keinerlei Einschränkungen bzgl. der Sprache und des Einsatzgebietes spezifiziert, wird das Policy Model in dieser Arbeit zur Repräsentierung von Policies und deren Bestandteilen verwendet. Auf eine tiefergehende Betrachtung des Informationsmodells wird an dieser Stelle verzichtet und auf die noch kommenden Abschnitte 5.3.4 und 5.5.4 dieses Kapitels verwiesen, in der das CIM Policy Informationsmodell umgesetzt wird.

Internet Engineering Task Force (IETF) Die IETF[4] engagiert sich in vielfältiger Weise im Bereich des policy–basierten Managements. Relativ früh wurde das eben erwähnte CIM Policy Model der DMTF adaptiert und zum sog. *Policy Core Information Model* [RFC 3060] angepasst, das einige kleinere Design–Schwächen der damaligen Version der DMTF behob. Ab CIM–Version 2.6 wurden allerdings von der DMTF die IETF–Verbesserungsvorschläge angenommen, so dass sich mittlerweile das CIM Policy Model der DMTF nicht von IETFs PCIM unterscheidet. Die Working Groups der IETF erweitern zum gegenwärtigen Zeitpunkt PCIM (allerdings in CIM–konformer Weise) dahingehend, dass Policies für das Dienstgüte–Management eingesetzt werden können. Desweiteren wird geprüft, wie das Informationsmodell durch ein LDAP–Schema geeignet repräsentiert werden kann. Neben den CIM–Arbeiten innerhalb der IETF, wurde auch das Protokoll *Common Open Policy Service (COPS) Proto-*

[3] `http://www.dmtf.org/`
[4] `http://www.ietf.org/`

 Als Buch erhältlich im Verlag Dr. Hut, München, www.dr.hut-verlag.de (ISBN 3-934767-97-4)

col [RFC 2748] zum Austausch von Policy–Informationen zwischen POD und POE[5] spezifiziert. Desweiteren wurde in [RFC 2750] RSVP bzgl. des Einsatzes von Policies erweitert sowie in [RFC 2749] das Zusammenspiel zwischen COPS und RSVP untersucht. Es existieren noch einige Internet Drafts der IPSec Working Group der IETF, die sich mit dem Einsatz von Policies im Sicherheitsmanagement beschäftigen, die allerdings an dieser Stelle nicht weiter betrachtet werden.

Zusammenfassung des Status Quo

Es wurde bereits zahlreiche, wertvolle Arbeit auf dem Gebiet des policy–basierten Managements geleistet, sowohl in Hinblick auf Policy–Sprachen als auch bzgl. einer generellen Architektur zum Durchsetzen von Policies. Hierbei ist zweifellos die am Imperial College London entworfene Policy–Sprache *Ponder* die am weitesten entwickelte Sprache für das policy–basierte Management von verteilten Systemen. Demnach würde es nahe liegen, Ponder als Basis für das prozessorientierte Management heranzuziehen. Allerdings weist Ponder in der aktuellsten Version (vgl. [Dami 02]) sehr viele Erweiterungen hinsichtlich des Sicherheitsmanagements (wie z.B. die Trennung von Obligations– und Autorisierungspolicies) auf, die für das allgemeine prozessorientierte Management keine Rolle spielen, die Implementierung einer Managementanwendung jedoch erheblich erschweren würden. Zudem wurden noch keinerlei Untersuchungen vorgenommen, die den Einsatz von Policies zum Management von IT (Geschäfts–)Prozessen vorsehen, so dass u.U. sowohl Ponder als auch andere existierende Policy–Sprachen den sich daraus ergebenden Anforderungen angepasst werden müssten. Ohne den weiteren Ausführungen dieses Kapitels vorgreifen zu wollen, wird im Weiteren eine Policy–Sprache entworfen, die sich auf Sprachelementen einer Vorversion von Ponder gründet. Grundsätzlich weist damit die in dieser Arbeit entwickelte Syntax der Policy–Sprache eine hohe Ähnlichkeit zu bisher entworfenen Policy–Sprachen, wie z.B. Ponder, auf. Allerdings zeigen sich bei der *Interpretierung* der Sprachbestandteile und damit in deren Semantik einige erhebliche Unterschiede, so dass keine existierende Policy–Sprache und die dazugehörige Implementierung 1:1 übernommen werden kann.

Alles in allem bieten dennoch die bisherigen Arbeiten in diesem Bereich eine sehr gute Basis für das in dieser Arbeit anvisierte Managementziel.

Mit PCIM resp. dem CIM Core Policy Model steht ein allgemein anwendbares Informationsmodell für Policies zur Verfügung, das ohne Weiteres übernommen werden kann, da keinerlei Einschränkungen bzgl. des Einsatzgebietes noch bzgl. der Sprache deklariert worden sind. PCIM wird in dieser Arbeit u.a. bei der Spezifikation der Grammatik der Policy–Sprache in Abschnitt 5.3.4 als auch beim objektorientierten Entwurf der Managementarchitektur in Abschnitt 5.5 beachtet.

[5] In der IETF–Terminologie wird allerdings der POD als *Policy Decision Point (PDP)* und der POE als *Policy Execution Point* (PEP) bezeichnet.

5.3.2 Entwurf einer policy–basierten Managementarchitektur für das prozessorientierte IT Management

Nachfolgend werden zunächst die grundlegenden Designentscheidungen sowie ein Grobentwurf der policy–basierten Managementarchitektur erläutert. Im Anschluss daran werden die sich daraus ergebenden Vorteile mit Bezug zum Anforderungskatalog aus Abschnitt 2.3.5 diskutiert. Der Architekturentwurf dient schließlich im nächsten Abschnitt 5.3.3 als eine der Grundlagen für die Spezifikation der Policy–Sprache.

Grundlegende Designentscheidungen

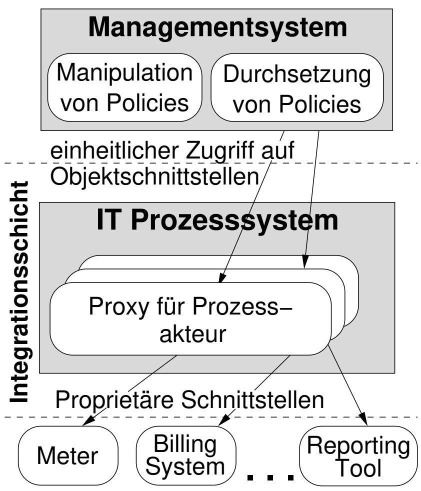

Abbildung 5.2: Policy–basierte, prozessorientierte Managementarchitektur

In Abbildung 5.2 ist der Grobentwurf der Basisarchitektur für ein policy–basiertes, prozessorientiertes IT Management abgebildet. Wie bereits erwähnt, bestimmt die Basisarchitektur insbesondere auch diejenigen Entitäten, die an der Ausführung der in einer PDL spezifizierten Managementanweisungen beteiligt sind. Um der Anforderung einer integrierten Sicht auf die zu managenden Entitäten einerseits und zusätzlich einem problemlosen Hinzufügen resp. Ersetzen von (proprietären) (Abrechnungs-) Komponenten andererseits zu genügen, sieht die Basisarchitektur eine *Integrationsschicht* zwischen dem Managementsystem und den zu managenden Entitäten, welche die bspw. an der Abrechnung beteiligten Teilprozesse tatsächlich realisieren, vor. Die Integrationsschicht repräsentiert auf diese Weise gegenüber dem Management den vollständigen zu steuerenden und zu überwachenden IT Prozess und stellt damit dem Management eine *prozessorientierte Sicht* zur Verfügung. Im Falle des Abrechnungsmanagements würde der vollständige Abrechnungsprozess, wie er in Abschnitt 4.3 vorgestellt wurde, durch die Integrationsschicht repräsentiert. Die in einer PDL spezifizierten Managementanweisungen werden im ersten Schritt folglich auf Vertretern (im Weiteren Proxies bzw. Proxy–Entität genannt) der tatsächlich zu managenden Entitäten ausgeführt. Somit ist aus Sicht des Managementsystems ein einheitlicher Zugriff auf Objektschnittstellen während der Durchsetzung von Policies möglich. Eine Proxy–Entität nimmt demnach im Gesamtsystem eine bivalente Rolle innerhalb der Integrationsschicht ein: Gegenüber der Managementanwendung agiert sie in der Rolle der *zu managenden Entität*, während sie den prozessrealisierenden Komponenten als *Manager* gegenübertritt.

Um die Flexibilität der Managementumgebung zu erhöhen, wird sowohl die Managementanwendung als auch die Integrationsschicht anhand von Funktionalitätsklassen aufgeteilt. Die Managementanwendung wird bzgl. der Funktionalität in Methoden, welche die Manipulation

 Als Buch erhältlich im Verlag Dr. Hut, München, www.dr.hut-verlag.de (ISBN 3-934767-97-4)

von Policies betreffen (z.B. Erzeugen/Ändern/Speichern/etc.), aufgeteilt und in Methoden, die die Durchsetzung von Policies betreffen (z.B. eine gegebene Policy interpretieren und die aufgeführten Managementanweisungen durchführen). Bezogen auf die Integrationsschicht eignet es sich gut, die partizipierenden *Akteure* eines IT Prozesses als Einteilung zu wählen. Damit wird gegenüber der Managementschicht die Funktionalität eines jeden Teilprozesses durch diejenigen Proxymodule der Integrationsschicht repräsentiert und teilweise implementiert, welche die jeweilig beteiligten Akteure repräsentieren. In der Regel wird der Aufruf einer Methode eines Proxymoduls auf die jeweilige Schnittstelle der tatsächlich zu managenden Ressource abgebildet. Dies kann beispielsweise in einem Aufruf einer proprietären API, einer SNMP Anweisung, einem Eintrag in einem Konfigurationsfile, etc. resultieren.

Alternativ wäre das Weglassen der Integrationsschicht denkbar. In diesem Fall ergeben sich allerdings einige Nachteile. Ohne die Integrationsschicht muss die prozessorientierte Sicht auf den Abrechnungsvorgang innerhalb der Managementanwendung, genauer gesagt innerhalb des Policy–Interpreters, realisiert werden. Das wiederum heißt, dass sich zwar nach außen hin nichts für den Manager ändert, da die Spezifikation von Policies in einer gegebenen PDL identisch bleibt, aber die Realisierung des Policy–Interpreters und –Enforcers als Kernstücke der Managementanwendung sich wesentlich schwieriger gestaltet. Dies ist darin begründet, dass der Policy–Interpreter bei Fehlen der Integrationsschicht alle möglichen Managementschnittstellen der zu managenden Entitäten kennen und zusätzlich der Policy–Enforcer die unterschiedlichen Zugriffsarten auf die Schnittstellen implementieren muss. Desweiteren müssten sowohl Policy–Interpreter als auch –Enforcer jedesmal erweitert und angepasst werden, wenn eine neue Komponente mit noch unbekannter Managementschnittstelle hinzugefügt bzw. eine bestehende ersetzt wird. Mit Einführung der Integrationsschicht treten diese Nachteile nicht auf.

Sofern nicht eine Standardmanagementarchitektur, wie z.B. das Internet/SNMP–Management, als Zielarchitektur der in Policies spezifizierten Managementanweisungen vorausgesetzt wird, tritt grundsätzlich in *jedem* policy–basierten Managementsystem eine Art Abstraktionsschicht auf, welche die Heterogenität der Managementschnittstellen gegenüber der policy–basierten Managementanwendung verschattet. In *Ponder* werden beispielsweise die Policies zunächst in einen einheitlichen Zwischencode übersetzt, der dann durch managementziel–spezifische *Module* in tatsächlich durchsetzbare Managementanweisungen übersetzt wird. Neu an der in diesem Abschnitt vorgestellten Integrationsschicht ist allerdings die Forderung, dass die Integrationsschicht aus *Proxies für Prozessakteure* besteht und damit gegenüber der policy–basierten Managementanwendung die prozessorientierte Sicht auf das Managementziel umsetzt. Diese Forderung muss u.a. beim Entwurf des Managementsystems (siehe Abschnitt 5.5) und bei der Implementierung beachtet und umgesetzt werden.

Eigenschaften der gewählten Managementarchitektur bezogen auf das Abrechnungsmanagement

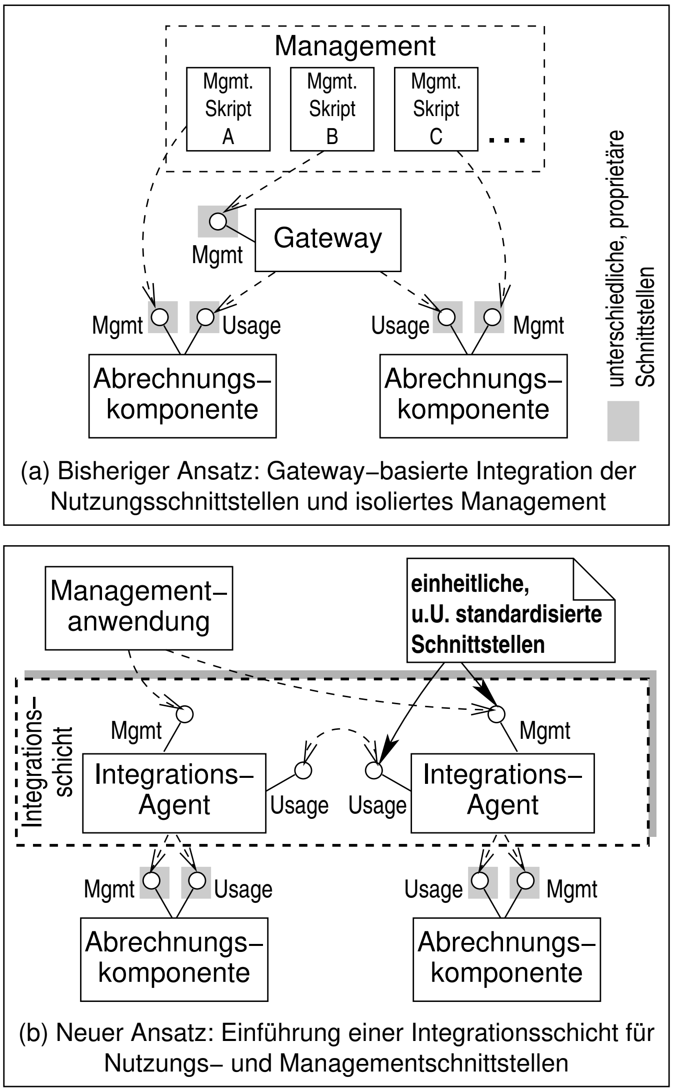

Abbildung 5.3: Vereinheitlichung der Nutzungs– und Managementschnittstellen

Um die besonderen Charakteristika der entworfenen Managementarchitektur speziell für das Abrechnungsmanagement zu erläutern, ist in Abbildung 5.3 die bisherige, gateway–basierte Architektur (Abb. 5.3 a)) der neuen, auf einer umfassenden Integrationsschicht basierenden Architektur (Abb. 5.3 b)) gegenübergestellt. Unschwer ist zu erkennen, dass es mit dem gateway–basierten Ansatz bisher nur gelungen ist, abrechnungsrealisierende Komponenten, die uneinheitliche, proprietäre Nutzungsschnittstellen besitzen, per vermittelnden Gateways zu einer funktionierenden Abrechnungslösung zusammenzustellen. Da sich die Gateways ausschließlich auf die Nutzungsschnittstellen der abrechnungsrealisierenden Komponenten beziehen, blieb der Zugriff auf die (proprietären) Managementschnittstellen der Abrechnungskomponenten (und der Gateways) unberührt, so dass sich aus Managementsicht eine uneinheitliche, heterogene, aus vielen Einzelschnittstellen bestehende Abrechnungslösung darbietet. In Konsequenz war das bisherige Management in isolierter und uneinheitlicher Art und Weise realisiert. Sofern das Management bisher überhaupt automatisiert und nicht durch den manuellen Eingriff des Administrators gestaltet wurde, war dieses meist durch Skripte, die zudem in unterschiedlichen Sprachen verfasst wurden, verwirklicht. Durch die auftretende Dynamik war es bislang unter diesen Umständen weder möglich noch sinnvoll eine ganzheitliche, „integrierende“ Managementanwendung zu implementieren.

Mit dem neuen Ansatz hingegen, der aus einer Integrationsschicht aus *Integrationsagenten* besteht, wird sowohl eine Vereinheitlichung der Management– als auch der Nutzungsschnittstellen erreicht. Die Integrationsagenten abstrahieren nicht nur von der tatsächlichen Management-

 Als Buch erhältlich im Verlag Dr. Hut, München, www.dr.hut-verlag.de (ISBN 3-934767-97-4)

schnittstelle, sondern auch von der tatsächlichen Nutzungsschnittstelle. Die Kombination der abrechnungsrealisierenden Komponenten zu einer funktionierenden Abrechnungslösung wird durch die Kooperation der dazugehörigen Integrationsagenten erreicht. Durch die Vereinheitlichung der Nutzungsschnittstellen der Integrationsagenten wird die Kooperation der Agenten wesentlich vereinfacht. Zusätzlich schafft die Vereinheitlichung der Managementschnittstellen eine einheitliche Managementsicht auf die Abrechnungslösung, so dass die Entwicklung einer ganzheitlichen Managementanwendung (im vorliegenden Fall policy–basiert) ermöglicht wird und auch sinnvoll ist. Somit erfüllt ein Integrationsagent gegenüber Manager und Abrechnungskomponente nicht nur die Funktionalität eines Proxys, sondern auch weiterhin die eines Mediators/Gateways zwischen den abrechnungsrealisierenden Komponenten.

Die entworfene Managementarchitektur wurde auf Grundlage des in Abschnitt 2.3.5 spezifizierten Anforderungskatalogs, mit besonderem Augenmerk auf die in Abschnitt 2.3.4 an das Abrechnungsmanagement formulierten Anforderungen, entwickelt. Nachfolgend werden die Eigenschaften der vorgestellten Managementarchitektur in Kombination mit den damit bereits erfüllten Anforderungen explizit genannt:

- *Trennung des Abrechnungsvorgangs vom Abrechnungsmanagement*
  Insgesamt wird mit der Integrationsschicht die geforderte klare Trennung zwischen dem Abrechnungsmanagement, in Form einer policy-basierten Managementanwendung, und dem zu managenden Abrechnungsvorgang, der in einer prozessorientierten Sicht durch die Integrationsschicht repräsentiert wird, vollzogen. Damit wird Anforderung ALL 2 erfüllt. Existierende Ansätze und Werkzeuge vermischen diese beiden getrennt zu betrachtenden Schichten und erschweren damit ein integriertes Abrechnungsmanagement, da das Management dadurch immer isoliert, auf eine Komponente bezogen, angewendet wird.

- *Keine Respezifikation von Policies bei Änderungen der Abrechnungsprozessimplementierung notwendig*
  Änderungen an Komponenten, die am Abrechnungsvorgang beteiligt sind, haben aufgrund der Schichtenarchitektur weder Auswirkungen auf bereits festgelegte Policies noch auf den Policy–Interpreter/Enforcer, so dass aus Managementsicht eine hohe Flexibilität bzgl. derartigen Änderungen gewährleistet ist. Damit wird die unter Punkt MGT 4 formulierte Anforderung erfüllt.

- *Integrationsschicht bietet einheitliche Managementschnittstelle*
  Die Integrationsschicht stellt in Form der Integrationsagenten eine einheitliche Zugriffsschnittstelle für die Durchsetzung von in Policies spezifizierten Managementanweisungen zur Verfügung. Damit werden heterogen zusammengestellte Abrechnungslösungen per se unterstützt und somit die unter MGT 5 spezifizierte Anforderung erfüllt. Da die Kombination des Policy–Managementkonzepts mit der Integrationsschicht auch eine integrative, einheitliche Sicht auf den Abrechnungsvorgang bietet, wird die Anforderung MGT 1 erfüllt.

- *Integrationsagent kann fehlende Funktionalität implementieren*
  Im einfachsten Fall bildet ein Integrationsagent einen Aufruf 1:1 auf die jeweilige Schnittstelle der tatsächlich zu managenden Entität ab und fungiert damit als Proxy im eigentlichen Sinne. Da aber der Funktionalitätsumfang von Softwarekomponenten, die zur Realisierung des Abrechnungsvorgangs in Betracht gezogen werden können, stark variieren kann, kann ein Integrationsagent auch dazu eingesetzt werden, um notwendige, aber fehlende Funktionalität bereit zu stellen. Somit ist der Funktionalitätsumfang einer Sofware/Hardware–Komponente kein K.O.–Kriterium mehr, wenn es um die Zusammenstellung einer Abrechnungslösung geht. Zudem wird die unter Punkt IMP 9 formulierte Anforderung bzgl. der Erweiterbarkeit der abrechnungsrealisierenden Komponenten erfüllt.

5.3.3 Analyse der Ausdrucksmächtigkeit einer Policy–Sprache für das prozessorientierte IT Management

Die *Policy Description Language (PDL)* legt allgemein die Syntax und implizit auch die Semantik der Sprache fest, in der Policies zur Spezifikation einer Managementaufgabe deklariert werden. In der Regel werden in Abhängigkeit von der jeweiligen Managementaufgabe und dem Managementumfeld einzelne Sprachelemente wie Schlüsselwörter, erlaubte Zeichenketten, etc. definiert. Die PDL legt damit insbesondere auch die Ausdrucksmächtigkeit einer Policy fest, so dass beim Entwurf der PDL darauf zu achten ist, dass diese weder zu allgemein ist, um Fehlspezifikationen zu vermeiden, noch zu einschränkend ist, um zu verhindern, dass gewünschte Managementaktivitäten nicht ausgedrückt werden können. Wie in Abschnitt 5.3.1 bereits dargestellt wurde, existieren mittlerweile zahlreiche Policy–Konzepte, die für unterschiedliche Anwendungsgebiete, wie dem Netz– und Systemmanagement oder dem Sicherheitsmanagement, entwickelt wurden. Um auf den Ergebnissen bereits existierender Arbeiten im Policy–Umfeld aufbauen zu können, wird zunächst analysiert, was mit einer Policy im prozessorientierten IT Management ausgedrückt werden soll, um anschließend ein bestehendes Konzept zur Spezifikation von Policies und der damit verbundenen PDL zu übernehmen bzw. diese, falls nötig, zu adaptieren.

In den nächsten beiden Abschnitten werden auf Basis des grundsätzlichen Aufbaus eines Prozesses (vgl. Kapitel 4) die Sprachelemente einer PDL zur Spezifikation von Policies und Meta–Policies analysiert und festgelegt. Anschließend wird in Abschnitt 5.3.4 die resultierende PDL in der *Extended Backus Naur Form (EBNF)* [ISO 14977] vorgestellt.

Analyse des Einsatzes von Policies und Meta–Policies

Wie bereits eingangs erwähnt, bestimmt die Managementaufgabe die notwendige Ausdrucksmächtigkeit einer Policy und stellt damit Anforderungen an die PDL und ihre Sprachelemente. Policies sollen im vorliegenden Fall einerseits dazu verwendet werden, einen IT Prozess, wie den Abrechnungsvorgang, geeignet zu steuern und zu überwachen, andererseits,

 Als Buch erhältlich im Verlag Dr. Hut, München, www.dr.hut-verlag.de (ISBN 3-934767-97-4)

das Management an sich soweit wie möglich zu vereinfachen und zu automatisieren. Folglich müssen alle am IT Prozess und an dessen Management beteiligten Entitäten ausgedrückt werden können, die aus Sicht eines prozessorientierten Managements beeinflussbar sein sollen.

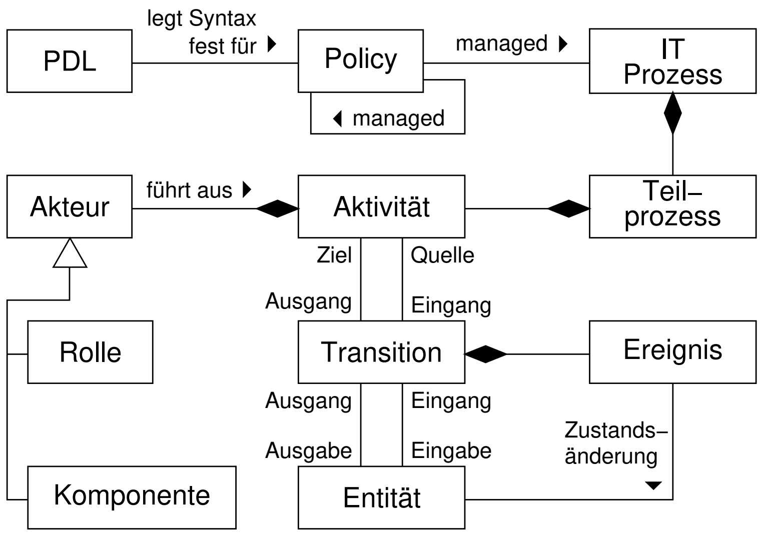

Abbildung 5.4: Statisches Diagramm der Beziehungen zwischen Policies und Prozessen

Zur Identifikation dieser Entitäten ist in Abbildung 5.4 ein statisches Diagramm der beteiligten Objekte dargestellt. Hierbei sind die Bestandteile eines IT Prozesses im rechten Teil der Abbildung auf Basis des schematischen Überblicks eines Prozessdiagramms in Abbildung 4.2 auf Seite 76 abgeleitet. Wie aus dieser Abbildung ersichtlich ist, legt die PDL die Syntax einer Policy zum Management des IT Prozesses als auch zum Management von Policies fest (in diesem Fall spricht man von einer *Meta–Policy*). Der IT Prozess besteht aus Teilprozessen, die wiederum aus Aktivitäten und Transitionen aufgebaut sind. Jede Aktivität ist hierbei zwingend einem Akteur zugeordnet, der die Aktivität tatsächlich ausführt. Der Beginn einer Transition ist immer mit einem Ereignis assoziiert, wobei die Beendigung einer direkt vorangestellten Aktivität als Standardereignis dient. Transitionen können nicht nur zwischen Aktivitäten auftreten, sondern auch zwischen Entitäten und Aktivitäten. Die Entitäten treten hierbei als Ein– bzw. Ausgabeobjekte von Aktivitäten auf. Eine Transition bzw. das damit assoziierte Ereignis kann in diesem Zusammenhang eine Zustandsänderung bei den assoziierten Entitäten hervorrufen. In den folgenden beiden Abschnitten werden basierend auf diesen Feststellungen die Anforderungen an die Ausdrucksmächtigkeit von Policies und Meta–Policies spezifiziert, die durch Anwendung derselben PDL deklariert werden sollen.

Anforderungen an die Ausdruckmächtigkeit von Policies Die Zielsetzung des Einsatzes von Policies (*ohne* das Meta–Policy–Konzept) im prozessorientierten IT Management ist die Überwachung und Steuerung der Prozessausführung eines Vorgangs, wie z.B. der Abrechnung. Das heißt, dass im vorliegenden Fall als die kleinsten vorkommenden Einheiten auf der gewählten Abstraktionsebene die Ausführung der Aktivitäten der Teilprozesse geeignet gesteuert werden sollen. Da während der Prozessanalyse nicht nur die Aktivitäten des Vorgangs selber, sondern auch die des „klassischen“ technischen Managements (wie z.B. die Überwachung der Dienstnutzung) untersucht werden, beschränkt sich die Einflußnahme von Managementseite auf das explizite Anstoßen von den beispielsweise in Kapitel 4 bereits identifizierten Aktivitäten (siehe hierzu auch die erklärenden Ausführungen in Abschnitt 4.3 auf Seite 77). Wie zu Beginn dieses Abschnitts bereits erwähnt wurde, wird die Ausführung einer Aktivität immer mit einer vorangegangenen Transition angestoßen. Diese wiederum hängt entweder mit einem expliziten Ereignis oder mit der Beendigung einer vorhergehenden Aktivitätsausführung zusammen. Somit muss aus Managementsicht innerhalb einer Policy die Ausführung von Aktivitäten sowohl durch *direkten Aufruf* (z.B. Methodenaufruf) als auch durch *Auslösen eines Ereignisses* ermöglicht werden. Die jeweilige Ausführung der innerhalb einer Managementpolicy formulierten Aktivitäten wird analog ebenfalls immer durch ein auftretendes Ereignis, das beispielsweise ebenfalls mit einer Transition assoziiert ist, angestoßen. Somit müssen innerhalb der Policy auch die *Ereignisse* angebbar sein, zu dem die spezifizierten Aktivitäten ausgeführt werden sollen. Wie bereits erwähnt, ist die Ausführung jeder Aktivität einem Akteur zugeordnet, so dass konsequenterweise die Policy auch diejenigen *Akteure* beinhalten muss, welche die innerhalb der Policy formulierten Aktivitäten ausführen. Um die tatsächliche Anwendung einer Policy und damit die Ausführung der innerhalb der Policy spezifizierten Aktivitäten feingranularer steuern zu können, sollte es ebenfalls möglich sein, *Bedingungen* für das Zutreffen einer Policy anzugeben. Desweiteren müssen Policies auch geeignet verwaltet werden können. Somit sollten Policies auch Verwaltungsinformationen, wie die zugewiesene *Zuständigkeitsdomäne*, enthalten. Somit muss eine PDL die Spezifikation von (Management–)Aktivitäten, Ereignissen, Bedingungen, Akteuren und Zuständigkeitsdomänen unterstützen, um geeignet im prozessorientierten IT Management eingesetzt werden zu können.

Anforderungen an die Ausdruckmächtigkeit von Meta–Policies Wie bereits zu Beginn dieses Abschnitts erwähnt wurde, sollen Meta–Policies dazu eingesetzt werden, um einen noch höheren Automatisierungsgrad zu erreichen. Meta–Policies spezifizieren Aktionen, die sich auf „normale“, u.U. bereits bestehende Policies auswirken. Damit werden mit Hilfe von Meta–Policies Managementaktivitäten, wie die Aktivierung/Deaktivierung und das Löschen/Erstellen von Policies, spezifiziert. Diese Aktivitäten werden ebenso wie nicht–Meta–Policies durch eintretende Ereignisse angestoßen und werden von einem Akteur tatsächlich ausgeführt (in diesem Fall von der Policy–Verwaltungsinstanz). Analog zum vorangegangenen Abschnitt ist es auch für Meta–Policies sinnvoll, dass diese sowohl Bedingungen für die Evaluierung als auch die zugewiesene, verantwortliche Zuständigkeitsdomäne enthalten.

Insgesamt ergeben sich demnach keine neuen Anforderungen an die Policy–Sprache selber, da die bereits identifizierten Sprachbausteine für die gewünschte Ausdrucksmächtigkeit von Meta–Policies ausreicht. Prinzipiell ändern sich lediglich die Managementaktivitäten und deren Ziel, welches Policies sind, im Gegensatz zu Entitäten, die am Abrechnungsvorgang direkt beteiligt sind.

 Als Buch erhältlich im Verlag Dr. Hut, München, www.dr.hut-verlag.de (ISBN 3-934767-97-4)

Analyse der PDL–Sprachbausteine

Die in den vorhergehenden Abschnitten identifizierten Anforderungen an die Policy–Sprache müssen nun geeignet durch die Definition von Sprachbausteinen repräsentiert werden. Hierfür bietet es sich an, bereits existierende Sprachkonzepte zu übernehmen oder, falls nötig, das am besten geeignete Konzept den Bedürfnissen des prozessorientierten IT Managements anzupassen.

Neben den bereits genannten, zu unterstützenden Anforderungen an eine Policy–Sprache sind die entscheidenden Kriterien für die Wahl eines bestehenden Policy–Konzepts und der damit verbundenen PDL, dass grundsätzlich Erweiterungen vorgesehen und zugelassen sind und auch bereits Konzeptumsetzungen in Form von bestehenden Implementierungen existieren und damit die Tragfähigkeit des Ansatzes in realen Umgebungen nachgewiesen ist. Damit eignet sich ausschließlich das Konzept der *Policy–Objekte mit Attributen*, das in zahlreichen Arbeiten bereits erfolgreich im Netz– und Systemmanagement eingesetzt wurde [Heil 00, Koch 97, KKK 96, MaSl 96]. Dieses Konzept sieht dabei eine *deklarative* Sprache zur Repräsentation einer Policy vor. Eine Policy wird konzeptionell als Objekt mit ausgewählten Attributen, wie beispielsweise *subject*, *target*, *action* and *constraints* angesehen. Ein *subject* wird im Rahmen dieser Arbeit als eine Menge von Objekten verstanden, für welche die mit *action* spezifizierten Managementaktivitäten auf der Menge der Zielobjekte (*target*) ausgeführt werden, sofern die Auswertung des in *constraints* angegebenen Ausdrucks wahr ergibt. Das *subject* legt damit fest, für welche Objekte eine Policy gültig ist, während das *target* lediglich das Ziel der Ausführung der spezifizierten Aktivitäten angibt. Um die geforderten Ereignisse angeben zu können, wird als Erweiterung das Attribut *event* dem Policy–Objekt hinzugefügt. Die damit spezifizierten Ereignisse stoßen letztendlich die Auswertung der (Meta–)Policy an.

Zusammenfassend enthält demnach eine Management–Policy für das dienstorientierte Abrechnungsmanagement mindestens die folgenden Bestandteile:

`POLICY` *ID* `FOR` *Subject+* `ON` *Target+* `ON EVENT` *Event+* `DO` *Action+* `CONSTRAINT` *ConstExpr+;*

Damit sind die notwendigen Basis–Sprachbausteine einer Policy für das prozessorientierte IT Management festgelegt, um die formulierten Anforderungen zu erfüllen. Damit die Spezifikation der PDL vervollständigt wird, müssen als nächster Schritt sowohl die Semantik der einzelnen, zu spezifizierenden Policy–Felder als auch die erlaubten Zeichenketten und Datentypen exakt festgelegt werden. Da eine Policy letztendlich maschinell durch einen Policy–Enforcer durchgesetzt wird, muss für die Definition der PDL bereits auch die Ausführungs– und Entwicklungsumgebung miteinbezogen werden. Ohne den Abschnitten des noch folgenden Kapitels 6, in dem die prototypische Implementierung detaillierter beschrieben wird, vorgreifen zu wollen, ist die Implementierung der Managementanwendung in eine CORBA–Umgebung eingebettet. Das heißt, dass sowohl die Managementanwendung, in welcher der Policy–Enforcer ein Bestandteil ist, als auch die Integrationsagenten der Integrationsschicht als CORBA–Objekte realisiert sind. Im Weiteren wird jeweils die Semantik der eben bereits identifizierten Sprachbausteine erklärt sowie zusätzlich, basierend auf dieser Semantikanalyse und den spezifischen

Charakteristika der CORBA–basierten Ausführungsumgebung, ein erster schematischer Entwurf der PDL–Syntax getätigt. In Abschnitt 5.3.4 wird schließlich auf dieser Grundlage die Grammatik der PDL formal in EBNF–Notation vorgestellt.

Nachfolgend wird jeweils eine verfeinerte, schematische Übersicht über die Syntax jedes identifizierten Sprachbausteins vorgestellt. Diese schematische Übersicht verwendet intuitive Darstellungsmechanismen, um das Nachvollziehen der umgesetzten Anforderungen an die Sprachbausteine zu ermöglichen und ist *nicht* als eine formale Spezifikation gedacht (diese erfolgt in Abschnitt 5.3.4). In der schematischen Übersicht werden Schlüsselwörter, die Teil der PDL sind, fettgedruckt und Daten, die durch den Policy–Ersteller zu spezifizieren sind, in kursiver Schrift notiert. Teile, die optional sind, werden in eckige Klammern ([ ]) gesetzt. Eine Wahlmöglichkeit wird durch einen senkrechten Strich (|) zwischen den Optionen visualisiert.

Policy–Feld Das Policy–Feld enthält Daten, die zur Verwaltung einer spezifizierten Policy benötigt werden. Zu den obligatorischen Daten gehören ein eindeutiger Identifikator und die Information, ob die Policy aktiviert wurde. Weitere nützliche Informationen, die allerdings als optional anzusehen sind, sind Erstellungsdatum, Datum der letzten Änderung und von wem diese Änderung durchgeführt wurde sowie der Gültigkeitszeitraum der Policy. Da eine Strategie zur Auflösung von *Policy–Konflikten* [LuSl 99, LuSl 97, MoSl 93] die Zuweisung von Prioritäten an Policies ist, sollte dies prinzipiell ermöglicht werden[6]. Ebenso erscheint es sinnvoll, Policies mit einer optionalen Versionsnummer zu versehen, welche die Version der PDL kennzeichnet. Die sich daraus ergebenden Syntaxbausteine der PDL sind in Tabelle 5.1 schematisch dargestellt.

Policy	**ID** =*id* [**Comment** = *Kommentar in Prosa*] [**Descriptor** { *dateOfCreation, modifiedBy, lastModified, expires* }] **Attributes** { *isEnabled, [priority, version]* }

Tabelle 5.1: Schematische Darstellung des *Policy*–Feldes

Subject–Feld Mit dem *Subject* wird eine Menge von Objekten angegeben, für welche die spezifizierte Policy gültig ist. Im Gegensatz zu bisher entwickelten deklarativen Policy–Sprachen, wie z.B. Ponder, wird das *Subject* *nicht* dazu verwendet, um anzugeben, welches Objekt die spezifizierten Managementaktionen tatsächlich ausführt. Das *Subject*–Feld wird im vorliegenden Fall vor allem dazu verwendet, Policies bestimmten Zuständigkeitsdomänen (z.B. einem Kunden, einer dienstleisterinternen Organisationseinheit, etc.) zuordnen zu können. Diese Information kann einerseits zur Umsetzung der Mandantenfähigkeit im Management (Anforderung MGT 11) verwendet werden, indem während der Policy–Evaluierung das Subjekt als Bedingung für die Anwendbarkeit der Policy aufgefasst wird. Andererseits kann das Subjekt auch zu Verwaltungszwecken als Suchargument oder Einordnungskriterium verwendet werden. Hierbei kommen insbesondere die während der Prozessanalyse in Abschnitt 4.3 identifizierten

[6]Dennoch sei an dieser Stelle angemerkt, dass das *Erkennen* von Policy–Konflikten immer noch ein z.T. ungelöstes Problem ist (siehe hierzu auch [Kemp 02]).

 Als Buch erhältlich im Verlag Dr. Hut, München, www.dr.hut-verlag.de (ISBN 3-934767-97-4)

Akteure als Subjekte einer Policy in Frage[7]. Demnach müssen Rollen, wie z.B. Kunde, Administrator, etc. und (technische) Komponenten, die den Abrechnungsvorgang (teilweise) implementieren, angebbar sein. Um möglichst flexibel zu sein, werden zur *Subject*–Spezifikation die folgenden Notationsmöglichkeiten unterstützt:

- *Domänennotation*
  Eine naheliegende Weise, mehrere zusammengehörende Policy–Subjekte zu spezifizieren, ist die Einteilung der Subjekte in (organisatorische) Domänen. Damit können Organisationsstrukturen nachgebildet und diese in nachvollziehbarer Art und Weise direkt in der Policy angegeben werden. Die Subjektspezifikation sollte in diesem Fall in der gewohnten Pfadnotation ermöglicht werden.

- *Rollennotation*
  Rollen können in komfortabler Art und Weise dazu eingesetzt werden, Subjekte über Organisationsstrukturen und damit Domänengrenzen hinweg zusammenzufassen (z.B. für die Policy–Spezifikation für Nutzer einer Dienstart, etc.). Hierbei kommen insbesondere die aus der Prozessanalyse bekannten Rollen, wie z.B. Kunde, Administrator, etc. in Frage.

- *Gruppennotation*
  Um neben der Zusammenfassung von Subjekten in Form von Domänen und Rollen auch eine völlig beliebige Gruppierung zu unterstützen, sollten Entitäten, die als Subjekte einer Policy in Frage kommen, auch explizit alle einzeln als Gruppe zusammengefasst werden können.

Es liegt auf der Hand, dass zur tatsächlichen Umsetzung der obig genannten Notationen entsprechende Verwaltungs– und Auflösungsmechanismen, wie z.B. Verzeichnisdienste, existieren müssen, um die Zuordnung eines Objekts zu einer Rolle und Domäne realisieren zu können. Um allerdings eine fehlerhafte Spezifikation automatisch entdecken zu können sowie den Rollen–/Domänen–/Namensauflösungsmechanismus zu vereinfachen, ist es sinnvoll, explizit die verwendete Notation anzugeben. Demnach werden in der PDL im Subjekt–Feld die Schlüsselwörter `Role` und `Group` mit aufgenommen, wobei die Domänennotation implizit durch Verwendung des Separators '`/`' angezeigt wird. Zusätzlich ist es zweckmäßig, auch Rollen in Domänen aufteilen zu können, so dass die Domänennotation zusammen mit der Rollennotation explizit erlaubt sein soll. Insgesamt ergibt sich die in Tabelle 5.2 schematisch dargestellte Notation für das Subjekt.

Target–Feld Mit dem *Target* werden die Default–Zielobjekte der in einer Policy im *Action*–Feld spezifizierten Managementanweisungen angegeben. Für das Target sind grundsätzlich dieselben Aussagen gültig wie für das Subjekt, so dass auch die gleiche Notation zur *Target*–Spezifikation herangezogen werden kann. In Tabelle 5.2 ist die Syntax des Target–Feldes schematisch notiert.

[7] Welche Akteure *sinnvollerweise* als Subjekte ausgewählt werden, wird bei der Spezifikation der Methodik in Abschnitt 5.4.1 besprochen.

Subject	*/Domäne/Subdomäne/...*[*/Entitätenname*]
	| [*/Domäne/Subdomäne/...*] **Role** *Rollenname*
	| **Group** { *Entität-ID, Entität-ID, ...* }

Tabelle 5.2: Schematische Darstellung des *Subject*–Feldes

Target	*/Domäne/Subdomäne/...*[*/Entitätenname*]
	| [*/Domäne/Subdomäne/...*] **Role** *Rollenname*
	| **Group** { *Entität-ID, Entität-ID, ...* }

Tabelle 5.3: Schematische Darstellung des *Target*–Feldes

Event–Feld Die innerhalb einer Policy angegebenen Ereignisse initiieren die Evaluierung der Policy. Die im *Action*–Feld der Policy spezifizierten Managementanweisungen werden genau dann ausgeführt, wenn die nach Auftreten eines spezifizierten Ereignisses angestoßene Auswertung des im Constraint–Feld angegebenen Ausdrucks wahr ergibt. Damit werden mit einem Ereignis die prinzipiellen Fälle angegeben, in welchen die Policy grundsätzlich anwendbar ist. Die Ereignisse werden in aller Regel von den Integrationsagenten generiert und versendet, z.B. wenn ein Aktivitätsübergang stattgefunden hat, eine neues Abrechnungsobjekt, wie z.B. ein Usage Record, erstellt oder der Datenbestand geändert wurde. Für die Spezifikation von Ereignissen muss das Ereignismodell der Ausführungsumgebung näher betrachtet werden. Wie eingangs bereits kurz erwähnt wurde, ist die Implementierung der Managementanwendung in eine CORBA–Umgebung eingebettet. Es wird eine Implementierung des *CORBA Event Services* [OMG 01-03-01] zum Versenden und Empfangen von Ereignissen verwendet. Innerhalb der Spezifikation des *CORBA Notification Service* [OMG 00-06-20] wird mit dem sog. *Structured Event* eine wohldefinierte Ereignis–Datenstruktur festgelegt, die den Aufbau von gesendeten und empfangenen Ereignissen festlegt. Der sog. *Event Header* eines Structured Events sieht die Spezifikation eines Ereignistyps und –namens vor und bietet damit Unterscheidungsmerkmale für Ereignisse. Der Ereignistyp sieht laut Spezifikation lediglich ein grobes Einteilungskriterium für die Art eines Ereignisses vor, wie z.B., dass ein Ereignis dem Netzmanagement zugeordnet wird. Der Ereignisname hingegen dient nicht, wie man fälschlicherweise annehmen könnte, der eindeutigen Identifizierung einer Ereignisinstanz, sondern der genauen Zuordnung zu einer Ereignisklasse. Somit werden im vorliegenden Fall die Ereignisnamen derjenigen Ereignisse in der Policy deklariert, die eine Evaluierung anstoßen sollen. Tabelle 5.4 enthält einen schematische Darstellung der Syntax des Ereignisfeldes einer Policy.

Event	{ *Ereignisname, Ereignisname, ...* }

Tabelle 5.4: Schematische Darstellung des *Event*–Feldes

 Als Buch erhältlich im Verlag Dr. Hut, München, www.dr.hut-verlag.de (ISBN 3-934767-97-4)

Action–Feld Auch für die Spezifikation des *Action*–Feldes muss die tatsächliche Ausführungsumgebung näher analysiert werden. Wie bereits in Abschnitt 5.3.2 erläutert wurde, werden die in einer Policy spezifizierten Managementanweisungen auf Integrationsagenten der Integrationsschicht ausgeführt. Im vorliegenden Fall werden die Agenten als CORBA–Objekte realisiert, die ihre Funktionalität an einer CORBA–Schnittstelle anbieten. Somit werden in einer Policy diejenigen Methoden der CORBA–Schnittstelle des Integrationsagenten angegeben, die bei Auftreten des spezifizierten Ereignisses aufgerufen werden sollen. Um dies in der gegebenen Ausführungsumgebung zu bewerkstelligen, benötigt man zumindest die folgenden Informationen: Eindeutige ID des CORBA–Objekts (entspricht der des Integrationsagenten), Name der aufzurufenden Methode sowie die zu übergebenden Parameter. Da die als *Target* spezifizierten Objekte die Default–Zielobjekte der auszuführenden Methoden sind, kann die ID des CORBA–Objekts auch weggelassen werden. Ansonsten wird als ID der entsprechende *CORBA–Objektname* des Integrationsagenten angegeben, der bei der Registrierung beim *CORBA Naming Service* [OMG 00-06-19] verwendet wird. Als Parameter der aufzurufenden Methode kommen entweder numerische oder Zeichen–Konstanten, die Ergebnisse von anderen Methodenaufrufen[8] und sog. Ereignis–Keys in Frage. Neben einem im sog. *Event Header* enthaltenen Ereignistyp und –namen, werden im sog. *Event Body* eines Structured Events detailliertere Informationen, die in Zusammenhang mit dem Ereignis stehen, als *Key–Value*–Paare mit übertragen. Der Key kann als eindeutiger Wert innerhalb des Event Bodys aufgefasst werden und dient der Identifizierung der übertragenen Werte (Values). Da insbesondere die Key–Value–Paare Daten enthalten können, die als Parameter für die spezifizierten Managementanweisungen dienen können, ist es zweckmäßig, die Event–Keys als Parameter mitangeben zu können. Desweiteren sollte es explizit ermöglicht werden, dass eine Policy auch die Erzeugung von Ereignissen initiiert, um beispielsweise auch auf diese Weise das Anstoßen von weiteren Policies und/oder Teilprozessen zu ermöglichen. Da der Aufruf resp. die Ausführung einer spezifizierten Aktion auch fehlschlagen kann, sollte die PDL die Möglichkeit der Spezifikation eines Blocks mit Anweisungen vorsehen, die bei Fehlschlagen von Managementaktionen ausgeführt werden. Damit wird die Robustheit der Managementlösung gegenüber auftretenden Fehlern erhöht (Anforderung MGT 10). Insgesamt ergibt sich die in Tabelle 5.5 schematisch dargestellte Notation für die Spezifikation von Aktionen.

Action	**default** {
	[*objectname.*] *methodname(Konstante,* **Event.***keyname, object.method(...), ...),*
	[*objectname.*] *methodname(...),*
	generateEvent *(keyname = value, keyname = value, ...),*
	...}
	[**onError** { */* identisch zum default-Block */* }]

Tabelle 5.5: Schematische Darstellung des *Action*–Feldes

[8] Damit sind explizit verschachtelte Methodenaufrufe erlaubt.

Constraint–Feld Für den Constraint–Ausdruck ist eine beliebige aussagenlogische Formel vorgesehen. Allerdings ist die Auswertung der Formel bei Vorliegen einer Normalform einfacher, da bspw. in der Disjunktiven Normalform (DNF) die Formel genau dann wahr ergibt, wenn eine ihrer Konjunktionen aus atomaren Formeln wahr ist. Da zudem jede aussagenlogische Formel in eine Normalform überführt werden kann, wird zur Leistungssteigerung die DNF bzw. KNF vorausgesetzt. Eine atomare Formel entspricht im vorliegenden Fall einem Vergleich zwischen zwei Werten. Diese Werte können beispielsweise Konstanten, Rückgabewerte von Methodenaufrufen als auch Werte, die mit dem Ereignis mitübertragen worden sind, sein. Tabelle 5.6 gibt einen schematischen Überblick über die Notation von Constraints.

Constraint	*(Konstante | methodCall | **Event.**keyname) **Op** (Konstante | ...)*
	| **CNF** { *Konstante **Op** methodCall* || *Konstante **Op** **Event.**keyname* || ... }
	| **DNF** { *Vergleich* **&&** *Vergleich* **&&** ... }

Tabelle 5.6: Schematische Darstellung des *Constraint*–Feldes

5.3.4 Spezifikation einer PDL für das prozessorientierte IT Management

In diesem Abschnitt wird auf Basis der im vorhergehenden Abschnitt analysierten und schematisch festgelegten Syntaxbausteine eine formale Spezifikation der Grammatik der PDL durchgeführt, die zur Entwicklung eines Policy–Parsers und –Enforcers durch Einsatz geeigneter Syntax–Werkzeuge verwendet werden kann. Die bereits identifizierten Schlüsselwörter spiegeln sich direkt (in leicht veränderter Form) in der Grammatik als Terminalsymbole wider. Die CORBA–basierte Ausführungsumgebung bestimmt die erlaubten Zeichenketten und Datentypen sowie die Zusammensetzung von im Aktionsteil festgelegten Methodenaufrufen.

Neben den im vorhergehenden Abschnitt 5.3.3 bereits identifizierten notwendigen Sprachbestandteilen, sieht die Grammatik die Umsetzung weiterer, bisher ungenannt gebliebener Konzepte vor. Es wird der durch DMTFs CIM Policy Model (auch: *Policy Core Information Model, PCIM*, vgl. Abschnitt 5.3.1) vorgeschlagene Verwaltungs– und Gruppierungsmechanismus von Policy–Bestandteilen aufgegriffen. PCIM sieht hierbei eine explizit außerhalb einer gegebenen Policy, separate und eigenständige Verwaltung der Aktionen, Ereignisse und Constraints vor. Eine Policy wird damit lediglich als ein Container angesehen, der auf bereits bestehende Deklarationen von Aktionen, Ereignissen und Constraints verweist. Damit wird eine Wiederverwendbarkeit von bereits durchgeführten Deklarationen unterstützt und auf diese Weise die Policy–Erstellung erleichtert. Desweiteren sieht PCIM einen durchgängigen, über alle Policy–Bestandteile hinweg verwirklichten Gruppierungsmechanismus vor. Das heißt, dass nicht nur Policies zu Policy–Mengen (*Policy Groups*), sondern auch Ereignisse, Aktionen und Constraints zu Mengen zusammengefasst werden können, die sich wiederum innerhalb einer Policy referenzieren lassen. Somit können nicht nur die Deklarationen von einzelnen Policy–Bestandteilen

 Als Buch erhältlich im Verlag Dr. Hut, München, www.dr.hut-verlag.de (ISBN 3-934767-97-4)

wiederverwendet werden, sondern auch (sinnvoll) gebildete Mengen. Dieses Konzept erhöht nochmals die Wiederverwendbarkeit von bereits durchgeführten Deklarationen und erleichtert damit die Policy–Erstellung. Diese vernünftigen PCIM–Konzepte können in der PDL wiedergefunden werden und werden durch '<*...Group*>' angezeigt.

Desweiteren enthält die Policy–Sprache Bestandteile, die notwendig sind, um die in dem noch kommenden Abschnitt 5.4.1 beschriebene Methodik zur Policy–Erstellung geeignet von der Managementanwendung zu unterstützen. Da sich Policies auf der Ebene der Teilprozesse ähneln, sieht die Methodik vor, dass diese als Schablonen für deren Management herangezogen werden (vgl. Abschnitt 5.4.2). Tatsächliche Policy–Deklarationen von z.B. Aktionen, Ereignissen, etc. können als Instantiierung dieser Schablonen verstanden werden, so dass diese evtl. für neu zu spezifizierende Policies wiederverwendet werden können. Um diese Art der Wiederverwendung durch ein Werkzeug unterstützen zu können, sieht die PDL vor, dass jeder Subject/Target/Event/Action/Constraint–Deklaration diejenigen Teilprozesse durch Nennung zugeordnet werden können, die dadurch gesteuert und/oder überwacht werden. Die Werkzeugunterstützung kann letztendlich dadurch verwirklicht werden, dass die Managementanwendung dem Policy–Ersteller bei Angabe des zu managenden Teilprozesses bereits deklarierte Policy–Bestandteile anzeigt. Der Policy–Ersteller kann dann einfach Teile durch Auswählen wiederverwenden. Dieses Konzept wird in der Grammatik der PDL durch das Nichtterminalsymbol '<*processAssignable*>' angezeigt.

Im Nachfolgenden wird die Grammatik der PDL in EBNF vorgestellt, welche die im vorhergehenden und diesen Abschnitt erwähnten Konzepte umsetzt. Die nachfolgende Grammatik wurde in [Danc 03] auf Basis des in [Radi 02] veröffentlichten Sprachentwurfs entwickelt. Die Grammatik sieht eine wohlgeklammerte Repräsentation der Sprachbausteine vor. Das heißt, dass die zu einem Schlüsselwort gehörende Deklaration immer durch geschweifte Klammern ('{' '}') eingeschlossen ist. Damit ist einerseits die Zuordnung von Deklarationen immer eindeutig und andererseits läßt sich dadurch die Grammatik sehr einfach durch *XML Schema* [XMLS-0, XMLS-1, XMLS-2] repräsentieren, das in dieser Arbeit zur Implementierung des Policy–Parsers verwendet wird.

EBNF der Policy Description Language

⟨*inputString*⟩ ::= ⟨*policySet*⟩

⟨*policySet*⟩ ::= ‘policySet{’ {⟨*policy*⟩ | ⟨*roleDefinition*⟩| ⟨*target*⟩ | ⟨*event*⟩ | ⟨*action*⟩ | ⟨*policyGroup*⟩} ‘}’

⟨*roleDefinition*⟩ ::= ‘roleDef{’ ⟨*referable*⟩ ⟨*roleName*⟩ ⟨*roleDescription*⟩ ‘}’

⟨*roleName*⟩ ::= ‘name{’ ⟨*xsd:token*⟩ ‘}’

⟨*roleDescription*⟩ ::= ‘description{’ ⟨*xsd:string*⟩ ‘}’

⟨*role* ⟩ ::= ‘role{’ ⟨*refs*⟩ ‘}’

⟨*policyGroup*⟩ ::= ‘policyGroup{’ ⟨*group*⟩ ⟨*comment*⟩ ‘}’

⟨*group*⟩ ::= ‘group{’ ⟨*refs*⟩ ‘}’

⟨*comment*⟩ ::= ‘comment{’ ⟨*xsd:string*⟩ ‘}’

⟨*referable*⟩ ::= ‘id=’ ⟨*xsd:ID*⟩ [⟨*libraryItem*⟩] [⟨*comment*⟩]

⟨*libraryItem*⟩ ::= ‘libraryItem=’ ⟨*xsd:boolean*⟩

⟨*processAssignable*⟩ ::= ⟨*referable*⟩ [⟨*relatedProcess*⟩]

⟨*relatedProcess*⟩ ::= ‘relatedProcesses{’ ⟨*process*⟩ {‘,’ ⟨*process*⟩} ‘}’

⟨*process*⟩ ::= ⟨*identifier*⟩ | ‘OTHER’ | ‘ALL’

⟨*policy*⟩ ::= ‘policy{’ ⟨*processAssignable*⟩ [‘domain{’ ⟨*domain*⟩ ‘}’] [⟨*subjectSet*⟩] [⟨*targetSet*⟩] ⟨*eventSet*⟩ [⟨*constraintSet*⟩] ⟨*actionSet*⟩ [⟨*descriptor*⟩] ⟨*attribs*⟩ ‘}’

⟨*descriptor*⟩ ::= ‘descriptor{’ [⟨*createdBy*⟩] [⟨*creationDate*⟩] [⟨*lastModified*⟩] [⟨*modifiedBy*⟩] [⟨*expires*⟩] ‘}’

⟨*attribs*⟩ ::= ‘attributes{’ ⟨*enabled*⟩ [⟨*priority*⟩] [⟨*version*⟩] ‘}’

⟨*createdBy*⟩ ::= ‘createdBy=’ ⟨*xsd:token*⟩

⟨*creationDate*⟩ ::= ‘dateOfCreation=’ ⟨*xsd:date*⟩

⟨*lastModified*⟩ ::= ‘lastModified=’ ⟨*xsd:dateTime*⟩

⟨*modifiedBy*⟩ ::= ‘modifiedBy=’ ⟨*xsd:token*⟩

⟨*expires*⟩ ::= ‘expires=’ ⟨*xsd:dateTime*⟩

⟨*enabled*⟩ ::= ‘isEnabled=’ ⟨*xsd:boolean*⟩

⟨*priority*⟩ ::= ‘priority=’ ⟨*digit*⟩ [⟨*digit*⟩]

⟨*version*⟩ ::= ‘version=’ ⟨*digit*⟩ ‘.’ ⟨*digit*⟩

⟨*subjectSet*⟩ ::= ‘subjectSet{’ (⟨*subject*⟩ | ⟨*subjectRef*⟩ | ⟨*subjectGroup*⟩) ‘}’

⟨*targetSet*⟩ ::= ‘targetSet{’ (⟨*target*⟩ | ⟨*targetRef*⟩ | ⟨*targetGroup*⟩) ‘}’

⟨*eventSet*⟩ ::= ‘eventSet{’ (⟨*event*⟩ | ⟨*eventRef*⟩ | ⟨*eventGroup*⟩) ‘}’

⟨*actionSet*⟩ ::= ‘actionSet{’ (⟨*action*⟩ | ⟨*actionRef*⟩ | ⟨*actionGroup*⟩) ‘}’

⟨*constraintSet*⟩ ::= ‘constraintSet{’ ⟨*cnf*⟩ | ⟨*dnf*⟩ | ⟨*constraint*⟩ | ⟨*constraintRef*⟩ ‘}’

⟨*subject*⟩ ::= ‘subject{’ [⟨*referable*⟩] ⟨*entityContainer*⟩ ‘}’

⟨*subjectRef*⟩ ::= ⟨*ref*⟩

⟨*subjectGroup*⟩ ::= ⟨*group*⟩

⟨*target*⟩ ::= ‘target{’ [⟨*referable*⟩] ⟨*entityContainer*⟩ ‘}’

⟨*targetRef*⟩ ::= ⟨*ref*⟩

⟨*targetGroup*⟩ ::= ⟨*group*⟩

⟨*entityContainer*⟩ ::= [⟨*domain*⟩] (⟨*entity*⟩ | ⟨*role*⟩)

⟨*domain* ⟩ ::= [‘/’] {⟨*identifier*⟩ ‘/’}

⟨*entity*⟩ ::= ⟨*identifier*⟩

⟨*event*⟩ ::= ‘event{’ [⟨*referable*⟩] ‘name=’ ⟨*identifier*⟩ ‘}’

⟨*eventRef*⟩ ::= ⟨*ref*⟩

⟨*eventGroup*⟩ ::= ⟨*group*⟩

⟨*action*⟩ ::= ‘action{’ ⟨*processAssignable*⟩ ⟨*defaultAction*⟩ [⟨*errorAction*⟩] ‘}’

⟨*actionRef*⟩ ::= ⟨*ref*⟩

⟨*actionGroup*⟩ ::= ⟨*group*⟩

⟨*defaultAction*⟩ ::= ‘default{’ ⟨*genericAction*⟩ ‘}’

⟨*errorAction*⟩ ::= ‘onError{’ ⟨*genericAction*⟩ ‘}’

⟨*genericAction*⟩ ::= ⟨*methodCall*⟩ | ⟨*generateEvent*⟩

⟨*methodCall*⟩ ::= ‘invoke{’ [⟨*objectName*⟩ ‘.’] ⟨*method*⟩ ‘}’

⟨*generateEvent*⟩ ::= ‘generateEvent{’ ⟨*eventName*⟩ ‘(’ ⟨*parameterSet*⟩ ‘)}’

⟨*method*⟩ ::= ⟨*identifier*⟩ ‘(’ [⟨*parameterSet*⟩] ‘)’

⟨*eventName*⟩ ::= ⟨*identifier*⟩

⟨*objectName*⟩ ::= ⟨*identifier*⟩

⟨*parameterSet*⟩ ::= ⟨*namedValue*⟩ {‘,’ ⟨*namedValue*⟩}

⟨*namedValue*⟩ ::= ‘namedValue{’ ‘name=’ ⟨*identifier*⟩ ⟨*value*⟩ ‘}’

⟨*value*⟩ ::= ⟨*type*⟩ ‘{’ ⟨*valueContent*⟩ ‘}’

⟨*type*⟩ ::= ‘float’ | ‘integer’ | ‘string’ | ‘boolean’ | ‘dateTime’

⟨*valueContent*⟩ ::= ⟨*literal*⟩ | ⟨*array*⟩ | ⟨*functionValue*⟩ | ⟨*attributeValue*⟩

⟨*literal*⟩ ::= ⟨*xsd:float*⟩ | ⟨*xsd:integer*⟩ | ⟨*xsd:string*⟩ | ⟨*xsd:boolean*⟩ | ⟨*xsd:dateTime*⟩

⟨*array*⟩ ::= ‘array{’ ⟨*length*⟩ ((⟨*literal*⟩ {‘,’⟨*literal*⟩ }) | ‘null’) ‘}’

⟨*functionValue*⟩ ::= ⟨*methodCall*⟩

⟨*attributeValue*⟩ ::= ‘`attributeValue{`’ ⟨*objectName*⟩ ‘`.`’ ⟨*attributeName*⟩ ‘`}`’

⟨*attributeName*⟩ ::= ⟨*identifier*⟩

⟨*cnf*⟩ ::= ‘`and{`’ ⟨*binaryLogOpOR*⟩ {‘`,`’ ⟨*binaryLogOpOR*⟩} ‘`}`’

⟨*dnf*⟩ ::= ‘`or{`’ ⟨*binaryLogOpAND*⟩ {‘`,`’ ⟨*binaryLogOpAND*⟩} ‘`}`’

⟨*constraint*⟩ ::= ((‘`equal`’ | ‘`smaller`’ | ‘`greater`’ | ‘`greaterEqual`’ | ‘`smallerEqual`’) ‘`{`’ ⟨*binaryPredicate*⟩ ‘`}`’) | ([‘`!`’] ⟨*value*⟩)

⟨*constraintRef*⟩ ::= ⟨*ref*⟩

⟨*binaryLogOpAND*⟩ ::= ⟨*constraint*⟩ {‘`&&`’ ⟨*constraint*⟩}

⟨*binaryLogOpOR*⟩ ::= ⟨*constraint*⟩ {‘`||`’ ⟨*constraint*⟩}

⟨*binaryPredicate*⟩ ::= ⟨*value*⟩ ‘`,`’ ⟨*value*⟩

⟨*identifier*⟩ ::= ⟨*alphachar*⟩ {⟨*alphachar*⟩ | ⟨*digit*⟩}

⟨*digit*⟩ ::= ‘`0`’ | .. | ‘`9`’

⟨*alphachar*⟩ ::= ‘`a`’ | ‘`b`’ | .. | ‘`z`’ | ‘`A`’ | ‘`B`’ | .. | ‘`Z`’ | ‘`_`’

⟨*refs*⟩ ::= ‘`{`’ ⟨*xsd:IDREFS*⟩ ‘`}`’

⟨*ref*⟩ ::= ‘`ref{`’ ⟨*xsd:IDREF*⟩ ‘`}`’

——— ***EBNF der Policy Description Language*** ———

5.4 Spezifikation von Policies für das prozessorientierte IT Management

Im Folgenden wird zunächst in Abschnitt 5.4.1 eine Methodik zur Spezifikation von Policies für das prozessorientierte IT Management festgelegt. Die Methodik leitet den Policy–Ersteller schrittweise bei der Policy–Deklaration an und dient damit insbesondere als Hilfestellung bei diesem Vorgang. Anschließend wird in Abschnitt 5.4.2 jeder in Kapitel 4 identifizierte Abrechnungsteilprozess auf prozessspezifische Aussagen bzgl. der Deklaration der einzelnen Policy–Felder untersucht. Ergebnis ist hierbei neben einer Aufstellung derjenigen Abrechnungsteilprozesse, die sich durch Policies (teilweise) automatisiert managen lassen, auch eine Übersicht der für jede Teilprozess–Policy in Frage kommenden Subjects, Targets und Events, welche als Policy–Schablonen verwendet werden können. Schließlich wird in Abschnitt 5.4.3 die Spezifikation von Meta–Policies diskutiert. In Abschnitt 5.4.4 werden die Ergebnisse dieses Abschnitts nochmals zusammengefasst.

5.4.1 Allgemeine Methodik

Die grundsätzliche Methodik zur Spezifikation von Policies und Meta–Policies sieht sowohl die Verwendung von Prozessmodellen, wie sie in Kapitel 4 für die Abrechnung entwickelten wurden, als auch von statischen Dienstmodellen, wie das in Kapitel 4 abgeleitete Abrechnungsdienstmodell, vor. Das Prozessmodell fokussiert auf die dynamischen Aspekte eines zu managenden Vorgangs, während sich das Dienstmodell auf die jeweiligen statischen Aspekte konzentriert. Beide Modelle werden als Basis verwendet, um die von einem *Policy–Ersteller* zu spezifizierenden Policy–Bestandteile, nämlich die evaluationsauslösenden Ereignisse, die auszuführenden Aktionen und das Subject und Target, zu identifizieren. Da, wie in Kapitel 4 festgelegt wurde, ein entwickeltes Prozessmodell sowohl Teilprozesse des „klassischen“ Managements (wie beispielsweise die Installation und Konfiguration von Komponenten) als auch des eigentlich zu managenden Vorgangs (wie z.B. den Abrechnungsvorgang) enthält, beschränken sich im vorliegenden Fall die durch Policies auszudrückenden Managementaktionen in dem expliziten Anstoßen einer identifizierten Aktivität bzw. eines Teilprozesses (siehe hierzu auch die Anmerkungen in Abschnitt 5.3.3). Somit gilt es herauszufinden, welche Aktivitäten zu welchem Zeitpunkt von welchem Akteur ausgeführt werden sollen, um dies dann durch eine geeignete Policy auszudrücken. Auch wenn die in Abbildung 5.5 visualisierten Schritte der Methodik allgemein in verschiedenen Managementfunktionsbereichen durchführbar sind, werden diese aus didaktischen Gründen anhand der in dieser Arbeit bereits entwickelten Prozess– und Dienstmodelle für die Abrechnung erklärt:

Schritt 1: *Formulierung der Managementaufgabe*

Die Spezifikation von Policies beginnt mit der Festlegung der Managementaufgabe, die mit Hilfe von Policies ausgedrückt werden soll. Das heißt, es muss eine grobe Eingrenzung des

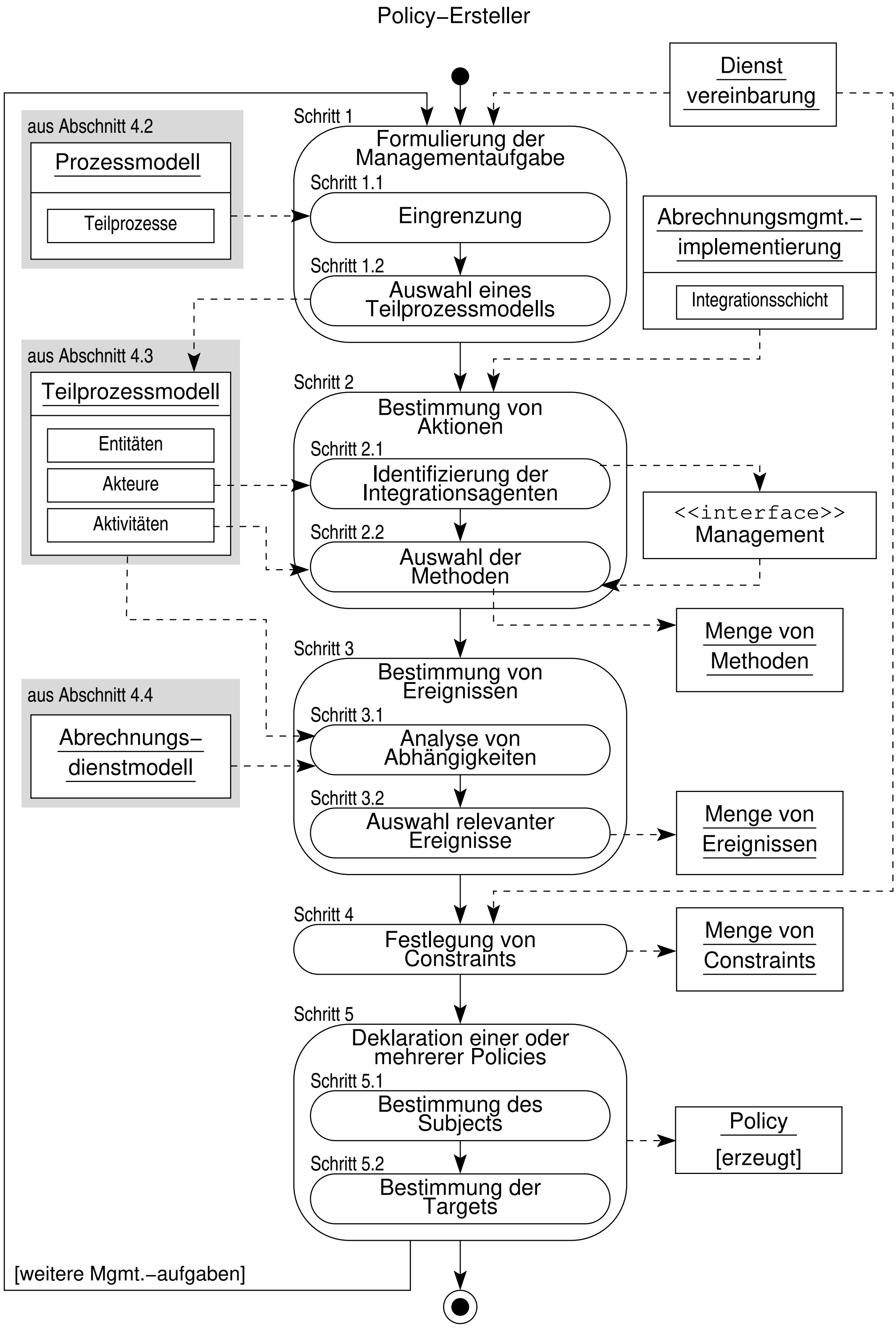

Abbildung 5.5: Methodik zur Spezifikation von Policies für das prozessorientierte IT Management

 Als Buch erhältlich im Verlag Dr. Hut, München, www.dr.hut-verlag.de (ISBN 3-934767-97-4)

Managementziels erreicht werden. Hierzu wird die zwischen Diensterbringer und Kunde geschlossene Dienstvereinbarung als eine Grundlage herangezogen. Im Detail werden die folgenden Schritte durchgeführt:

Schritt 1.1: *Eingrenzung anhand des Prozessmodells*
Wie schon in Abschnitt 5.3.3 ausgeführt wurde, ist das generelle Managementziel, das im vorliegenden Fall verfolgt wird, die Steuerung der am Abrechnungsvorgang beteiligten Teilprozesse und deren Aktivitäten. Zur Eingrenzung der Managementaufgabe eignet sich damit als Hilfestellung für den Policy–Ersteller das in Abbildung 4.2 auf Seite 76 dargestellte Prozessmodell, das einen Überblick über alle für die Abrechnung und das Abrechnungsmanagement relevanten Aufgaben in Form von Teilprozessen entlang des Dienstlebenszyklus enthält. Der Policy–Ersteller sollte in erster Näherung die für die, aus der Dienstvereinbarung abgeleiteten Managementaufgabe, relevante *Phase* des Dienstlebenszyklus bestimmen, um im nächsten Schritt den passenden *Teilprozess* dieser Phase zu wählen.

Schritt 1.2: *Auswahl des Teilprozessmodells*
Die einzelnen an der Abrechnung beteiligten Teilprozesse wurden bereits detailliert in Abschnitt 4.3 untersucht und können damit als *Vorlage* für die Spezifikation von Policies verwendet werden. Das zu diesem Teilprozess gehörende detaillierte *Teilprozessmodell* (siehe Abschnitte 4.3.2–4.3.18) dient in einem Teil der kommenden Schritte der Methodik als Eingabeobjekt.

Schritt 2: *Bestimmung von Aktionen*
Wie einführend zu Beginn dieses Abschnitts erklärt wurde, beschränken sich die durch Policies zu spezifizierenden Managementaktionen auf das Anstoßen von Aktivitäten, die in Kapitel 4 durch die Prozessanalyse identifiziert worden sind. Hierzu kann das aus dem vorhergehenden Schritt ausgewählte Teilprozessmodell herangezogen werden, das die für die auszudrückende Managementaufgabe relevanten Aktivitäten enthält. Folglich muss zunächst entschieden werden, welche der Aktivitäten anzustoßen sind, um anschließend zu entscheiden, auf welche Weise dies geschehen soll (durch direkten Aufruf oder Erzeugen eines Ereignisses). Hierzu wird folgendermaßen vorgegangen:

Schritt 2.1: *Identifizierung der beteiligten Integrationsagenten*
Jeder in den Prozessmodellen identifizierte Akteur wird durch einen eigenständigen Integrationsagenten der Integrationsschicht (siehe hierzu den Grobentwurf der Architektur in Abbildung 5.3 b) auf Seite 142) repräsentiert. Somit muss zunächst festgestellt werden, (a) welche Agenten der Integrationsschicht die am Teilprozess beteiligten Akteure tatsächlich repräsentieren, und (b) welchen Umfang die Managementschnittstelle des Integrationsagenten hat. Die von der Managementschnittstelle bereitgestellten Methoden bestimmen die Einflussmöglichkeiten auf die Prozessausführung, demnach welche im Teilprozessmodell repräsentierten Aktivitäten tatsächlich angestoßen werden können.

Schritt 2.2: *Auswahl der Methoden*

In Abhängigkeit von der Managementaufgabe werden die aufzurufenden Methoden ausgewählt, die der Erfüllung des Managementziels am nächsten kommen. Hier besteht grundsätzlich die Wahl zwischen direktem *Aufruf einer Methode* der Managementschnittstelle und dem *Generieren eines Ereignisses*, das entweder von einer anderen, aktiven Policy ausgewertet wird oder direkt die gewünschte Aktivität anstößt, im Falle, dass der Zielagent der Integrationsschicht Ereignisse direkt auswertet. Ebenfalls wird bereits zu diesem Zeitpunkt festgelegt, was zu tun ist, falls der Aufruf einer Methode und damit die Ausführung einer Managementaktion fehlschlägt. In jedem Fall wird das Fehlschlagen des Durchsetzens von Policies automatisch vom Policy–Enforcer mitprotokolliert. Je nach gebotener Funktionalität besteht zusätzlich die Möglichkeit, einen Rollback durchgeführter Aktionen einzuleiten oder einen Administrator zu benachrichtigen. Diese Aktionen werden gesondert im dafür vorgesehenen `onError`–Block notiert.

Bereits an dieser Stelle sei der Fall erwähnt, dass ein Akteur eine durch Personen eingenommene Rolle repräsentiert. In diesem Fall existiert ein gesonderter Integrationsagent, der für *alle* durch Personen eingenommenen Rollen als Stellvertreter agiert. Deren Managementschnittstelle sieht allerdings nur eine Benachrichtigung der Personen z.B. per Email vor, die dazu auffordert, einen bestimmten Teilprozess bzw. eine Aktivität durchzuführen (siehe hierzu auch Abschnitt 5.5).

Schritt 3: *Bestimmung von Ereignissen*

Als nächstes muss festgelegt werden, was der *Anlaß* zum Anstoßen einer Aktivität ist. Dieser Anlaß wird schließlich als Ereignis in der Policy ausgedrückt, das letztendlich die Evaluierung der Policy auslöst und damit im Falle einer positiven Constraint–Evaluierung zum Ausführen der spezifizierten Managementaktionen führt. Die für einen Teilprozess und damit für die zu deklarierende Policy relevanten Ereignisse werden folgendermaßen festgelegt:

Schritt 3.1: *Ereignisidentifizierung anhand von Abhängigkeiten*

Zunächst muss festgestellt werden, welche Ereignisse potentiell für die bereits ausgewählten Aktionen in Frage kommen, d.h. es gilt zu erkennen, welche Ereignisse im gegebenen System tatsächlich auftreten und damit auch auswertbar sind. Zur Identifizierung von Ereignissen eignet es sich in einem ersten Schritt, anhand des Prozessmodells und des Abrechnungsdienstmodells die *Abhängigkeiten* der am Teilprozess beteiligten Entitäten, Akteure und Aktivitäten zu anderen Objekten des Abrechnungsvorgangs zu untersuchen. Anhand des Prozessmodells können *Abhängigkeiten zwischen Aktivitäten* in Form von *Transition* erkannt werden. Wie bereits in Abschnitt 5.3.3 erklärt wurde, sind Transitionen immer mit einem *impliziten* oder *expliziten Ereignis* verbunden. Im Gegensatz zum impliziten Ereignis, das als Beendigung der vorangegangenen Aktivität verstanden wird, wird bei expliziten Ereignissen davon ausgegangen, dass diese auch tatsächlich in einem System auftreten, damit auswertbar sind und folglich als Ereignisse für die Policy–Spezifikation in Frage kommen, sofern diese mit

 Als Buch erhältlich im Verlag Dr. Hut, München, www.dr.hut-verlag.de (ISBN 3-934767-97-4)

den bereits festgelegten Aktivitäten zusammenhängen[9]. Für die impliziten Ereignisse müssen die Transitionen genauer betrachtet werden. Grundsätzlich lassen sich Transitionen in folgende Klassen aufteilen:

- *Zwischen Aktivitäten verschiedener Akteure desselben Teilprozesses*
  Diese Transitionen werden auch als *Interaktionen* zwischen den beteiligten Akteuren bezeichnet. Interaktionen sind grundsätzlich gut geeignet, um als Ursprung für das Auftreten eines Ereignisses zu dienen. Dies liegt darin begründet, dass, falls nicht bereits entsprechende Ereignisse von den beteiligten Akteuren generiert werden, Interaktionen prinzipiell „von außen" entdeckt werden können, indem die, zwischen den interagierenden Akteuren auftretenden Transitionen durch eine dritte Instanz (einem sog. *Interceptor*) abgehört bzw. abgefangen werden. Der Interceptor kann dann bei Erkennen einer Transition ein entsprechendes Ereignis generieren und läßt sich gut in die Integrationsschicht der Policy–Managementarchitektur einbetten.
- *Zwischen Aktivitäten desselben Akteurs und Teilprozesses*
  Die Transitionen treten innerhalb des Zuständigkeitsbereichs eines Akteurs auf. Sofern der Akteur bei den Aktivitätsübergängen nicht von sich aus Ereignisse generiert, können diese Art der Transitionen von außen nicht erkannt werden. Somit muss in diesem Fall die prozessrealisierende Komponente genauer auf diese Art von Ereignisse analysiert werden.
- *Zwischen Aktivitäten unterschiedlicher Teilprozesse*
  Diese Klasse kann auf die beiden erstgenannten Klassen zurückgeführt werden, d.h. ob die Transition zwischen unterschiedlichen Akteuren stattfindet und damit eine Interaktion ist oder nicht.

Aus dem Abrechnungsdienstmodell können Abhängigkeiten zwischen Akteuren und (Ein–/Ausgabe) Entitäten, die in Form von Assoziationen visualisiert sind, erkannt werden. Hierbei sollten diejenigen Abhängigkeiten in Betracht gezogen werden, die von Akteuren resp. Entitäten ausgehen, die an dem fokussierten Teilprozess beteiligt sind. Grundsätzlich können folgende Ausprägungen auftreten:

- *Zwischen Rollen, die von Personen eingenommen werden*
  In diesem Fall ist nicht davon auszugehen, dass ein für die Policy verwertbares Ereignis generiert wird, so dass dies im vorliegenden Schritt vernachlässigt werden kann.
- *Zwischen prozessrealisierenden Komponenten*
  Besteht eine Assoziation zu einer prozessrealisierenden Komponente, die nicht in dem fokussierten Teilprozess vorkommt, so ist das ein Hinweis darauf, dass eine Interaktion über Teilprozesse hinweg stattfindet. Das heißt, dass ein Transitionspfad zwischen den Teilprozessen existiert. Wird auf dem Transitionspfad ein Er-

[9] Ansonsten würde der mit einem expliziten Ereignis verbundene Übergang nie statt finden und somit wäre die Prozessspezifikation fehlerhaft.

eignis generiert, so kann dies als auslösender Event in der Policy–Spezifikation herangezogen werden.

- *Zwischen einer prozessrealisierenden Komponente und (Ein–/Ausgabe) Entitäten*
  In diesem Fall sollten die jeweiligen assoziierten (Ein–/Ausgabe) Entitäten auf Zustandsänderungen untersucht werden. Tritt eine Zustandsänderung auf (z.B. eine Datei wurde geändert/angelegt, etc.), so kann dies als evaluationsauslösendes Ereignis verwendet werden.

Mit der Analyse der Abhängigkeiten anhand der in dieser Arbeit in Kapitel 4 entwickelten Modelle sind in diesem Teilschritt eine Menge von Ereignissen identifiziert worden.

Schritt 3.2: *Auswahl relevanter Ereignisse*
Aus der Menge der Ereignisse werden diejenigen ausgewählt, welche die Fälle und Zeitpunkte, zu denen eine Evaluation der Policy gefordert ist, am genauesten treffen. Bei Ungenauigkeiten kann der Auslösemechanismus durch die Spezifikation von Constraints (Schritt 4 in der Methodik) beeinflußt werden.

Als sinnvoll können allerdings nur Ereignisse bezeichnet werden, die auf Basis von Interaktionen oder Zustandsänderungen von Ein–/Ausgabeentitäten ausgelöst werden. Dies liegt zum einen darin begründet, dass die Generierung dieser Art von Ereignisse sichergestellt ist (siehe hierzu auch die Ausführungen des noch kommenden Abschnitts 5.5). Zum anderen hängt eine Interaktion bzw. Zustandsänderung auch immer mit dem Beginn einer eigenständigen Teilprozessausführung *innerhalb* eines Akteurs zusammen. Das explizite Anstoßen und damit Kontrollieren dieser Teilprozessausführung ist aber auch genau das, was man aus Managementsicht mit einer Policy ausdrücken will.

Schritt 4: *Festlegung von Constraints*
Mit dem Constraint–Ausdruck können (weitere) Einschränkungen bzgl. der Gültigkeit und Anwendbarkeit der Policy deklariert werden. In der Regel sind diese Einschränkungen dienst– und/oder dienstvereinbarungsspezifisch, so dass auch in diesem Schritt die Dienstvereinbarung als Eingabeobjekt herangezogen werden sollte. Typischerweise werden für die Constraint–Auswertung Daten verwendet, die entweder mit den deklarierten Ereignissen mitgeliefert werden oder direkt aus der Dienstvereinbarung abgeleitet werden, wie beispielsweise die Zeitangaben für die Gültigkeit einer Policy (z.B. nur wochentags, etc.).

Schritt 5: *Deklaration einer oder mehrerer Policies*
Um die Policy–Spezifikation abzuschließen, müssen noch Subject und Target festgelegt werden:

Schritt 5.1: *Bestimmung des Subjects*
Als *Subject* wird üblicherweise der Identifikator des zwischen Kunden un Diensterbringer vereinbarten Dienstes angegeben, für welchen die Policy erstellt wurde[10]. Al-

[10] Im Prinzip wird jede Policy mit dem Ziel erstellt, die mit einem Kunden geschlossene Dienstvereinbarung zu

 Als Buch erhältlich im Verlag Dr. Hut, München, www.dr.hut-verlag.de (ISBN 3-934767-97-4)

ternativ kann auch der Identifikator eines Kunden angegeben werden, sofern die Policy für alle mit diesem Kunden vereinbarten Dienste gilt. Ebenso möglich, jedoch weitaus unüblicher, ist die Spezifikation von (provider–internen) Rollen und somit der Verzicht der expliziten Angabe eines Identifikators. Falls die Policy für mehrere Rollen resp. Entitäten gilt, kann alternativ auch die Domänennotation verwendet werden.

Schritt 5.2: *Bestimmung des Targets*
Grundsätzlich können mehrere Aktionen in einer Policy spezifiziert sein, die von unterschiedlichen Akteuren und damit Integrationsagenten ausgeführt werden. Derjenige Integrationsagent, welcher die meisten der spezifizierten Aktionen auszuführen hat, sollte als *Target* angegeben werden. Hierfür kann der CORBA–Name direkt angegeben werden oder die Domänen–/Rollennotation verwendet werden. Für die übrigen Aktionen sollte der CORBA–Name des ausführenden Agenten direkt bei der Aktionsdeklaration (siehe Tabelle 5.5 auf Seite 151) angegeben werden. Alternativ ist es auch denkbar, für jeden beteiligten Akteur, der als Target agiert, eine eigene Policy zu spezifizieren.

Mit Abschluss des Schritts 5 wurde eine bzw. mehrere Policies für eine gegebene Managementaufgabe spezifiziert. Für weitere Managementaufgaben sind die Schritte 1 bis 5 zu wiederholen.

Es muss betont werden, dass die angegebene Methodik *nicht* auf die Spezifikation von Abrechnungsmanagement–Policies beschränkt ist, sondern bei gegebenen Prozess– und Dienstmodellen anderer Managementfunktionsbereiche äquivalent anwendbar ist.

5.4.2 Überblick über Abrechnungsmanagementpolicies

Die im vorhergehenden Abschnitt vorgestellte allgemeine Methodik sieht prinzipiell vor, ein Teilprozessmodell als Vorlage für die Spezifikation von Policies heranzuziehen. Damit eignen sich insbesondere die in dieser Arbeit in Kapitel 4 entwickelten Teilprozessmodelle für die Deklaration von Abrechnungsmanagementpolicies. Die Spezifikation von konkreten Policies für das Abrechnungsmanagement ist allerdings nur mit dem zusätzlichen Vorhandensein einer Dienstvereinbarung, einer Dienstimplementierung und, davon abhängig, einer Dienstmanagementimplementierung möglich. Das heißt, dass die Teilprozessmodelle nicht als absolut strikte Vorlagen zu verstehen sind, die jede für die Policy–Spezifikation notwendige Information enthalten. Vielmehr sind die Teilprozessmodelle als Vorlagen mit gewissen Freiheitsgraden anzusehen, welche durch die Dienstvereinbarung sowie der Dienst(management)implementierung eingeschränkt werden. Dennoch können grundsätzlich auch ausschließlich auf Basis der Teilprozessmodelle allgemeine Aussagen über die dazugehörigen, zu spezifizierenden Policies geäußert werden. Nachfolgend wird jeder in Abschnitt 4.3 untersuchte Teilprozess durch Anwendung der im vorhergehenden Abschnitt vorgestellten Methodik auf allgemeingültige Aussagen bzgl. der Policy–Spezifikation untersucht, um damit dem Policy–Ersteller in Form von

erfüllen.

Policy–Schablonen eine weitere Hilfestellung zusätzlich zu der in Abschnitt 5.4.1 vorgestellten Methodik zu bieten.

In den kommenden beiden Abschnitten wird jeder identifizierte Teilprozess bzgl. der Spezifikation von Policies näher untersucht, um anschließend darauf basierend eine Kategorisierung der Abrechnungsmanagementpolicies zu entwickeln.

Analyse der Teilprozessmodelle zur Spezifikation von Abrechnungsmanagementpolicies

In diesem Abschnitt wird für jeden einzelnen Teilprozess zunächst geprüft, ob für diesen eine Policy–Spezifikation, die den Eingriff von Managementseite (teilweise) automatisiert, auf sinnvolle Art und Weise möglich ist. Hierfür werden in einem ersten Schritt die am Teilprozess beteiligten Akteure näher untersucht, um im nächsten Schritt die durch einen Akteur ausgeführten Aktivitäten zu begutachten. Allgemein kann festgestellt werden, dass Akteure, die üblicherweise von technischen Komponenten eingenommen werden, auf ein hohes Automatisierungspotential hinsichtlich des Managements hinweisen und somit die Spezifikation von Policies möglich und sinnvoll ist. Hingegen bedarf es bei Akteuren, die Rollen sind, die von Personen eingenommen werden, einer näheren Analyse der durch diese Rolle ausgeführten Aktivitäten. In diesem Fall muss letztendlich die Semantik der Aktivitäten untersucht werden, um festzustellen, ob eine Automatisierung des Managements und damit eine Policy–Spezifikation möglich ist. Unabdingbare Voraussetzung und damit K.O.–Entscheidungskriterium hierfür ist, dass die Ressourcen, die zur Aktivitätsausführung verwendet und benötigt werden, von Seiten des technischen Managements beeinflußt werden können.

Sofern eine Automatisierung von Managementseite möglich ist, werden in der nachfolgenden Aufstellung Vorschläge für die Spezifikation der Policy–Felder Subject, Target, Event und Action diskutiert. Für die Teilprozesse wird die folgende vierstufige Bewertungsskala verwendet: ↯↯ — kein sinnvolles Management mit Policies ausdrückbar, ↯ — nur Prozessinitiierung als Managementeingriff mittels Policies ausdrückbar, ✓ — teilweises Management des Teilprozesses mit Policies möglich, ✓✓ — vollständiges Management des Teilprozesses mittels Policies möglich.

- ***Kundenanalyse*** *(siehe Abschnitt 4.3.2, Seite 79)*
  - *Interagierende Akteure:* Diensterbringer, Kunde
    Bei den Akteuren handelt es sich um Rollen, die in Outsourcing–Szenarios von Personen eingenommen werden.
  - *Aktivitäten:* Die spezifizierten Aktivitäten lassen sich, bezogen auf Outsourcing–Szenarios und den damit verbundenen Individualdiensten, nicht automatisieren, da diese ausschließlich ohne Einsatz von technischen Hilfsmitteln durchgeführt werden.

 Als Buch erhältlich im Verlag Dr. Hut, München, www.dr.hut-verlag.de (ISBN 3-934767-97-4)

- *Policy-Spezifikation:* ↯
  Als Konsequenz ist eine Spezifikation von Policies zur Unterstützung des Teilprozesses nicht sinnvoll.

- ***Kostenprognose*** *(siehe Abschnitt 4.3.3, Seite 81)*
  - *Interagierende Akteure:* IT Finance Manager, Dienstdesigner
    Auch in diesem Teilprozess sind die Akteure Rollen, die von Personen eingenommen werden.
  - *Aktivitäten:* Die im Teilprozess spezifizierten Aktivitäten werden von den Akteuren mit Einsatz von technischen Hilfsmitteln durchgeführt, die sich nicht sinnvoll unter die Kontrolle des technischen Managements setzen lassen können, so dass keinerlei Ansatzpunkt für ein automatisiertes Management besteht.
  - *Policy-Spezifikation:* ↯
    Folglich ist eine Spezifikation von Policies für diesen Teilprozess nicht sinnvoll bzw. nicht möglich.

- ***Tarifierung*** *(siehe Abschnitt 4.3.4, Seite 83)*
  - *Interagierende Akteure:* Kunde, Vertrieb, IT Finance Manager
    Die Akteure sind auch in diesem Fall Rollen, die in Outsourcing–Szenarios von Personen eingenommen werden.
  - *Aktivitäten:* Die innerhalb des Teilprozesses identifizierten Aktivitäten finden ohne den Einsatz von technischen Hilfsmitteln statt, die potentiell und in sinnvoller Art und Weise Zugriff für das technische Management bieten würden, so dass das dazugehörige Management nicht automatisierbar ist.
  - *Policy-Spezifikation:* ↯
    Eine Policy–Spezifikation für das technische Management zur Unterstützung dieses Teilprozesses ist nicht sinnvoll. Zwar wäre es grundsätzlich denkbar, Policies als eine Art Unternehmensrichtlinie für die Tarifierung zu spezifizieren, allerdings ist diese nicht ohne weiteres automatisch durchsetzbar, da die beteiligten Akteure nicht durch technische Komponenten eingenommen werden.

- ***Implementierung der Messkomponenten*** *(siehe Abschnitt 4.3.5, Seite 88)*
  - *Akteur:* Entwickler
    Bei diesem Akteur handelt es sich um eine Rolle, die von einer Person eingenommen wird.
  - *Aktivitäten:* Grundsätzlich werden alle im Teilprozess identifizierten Aktivitäten mit Unterstützung der Entwicklungsumgebung (zur Erstellung von Programmcode) durchgeführt. Die Entwicklungsumgebung bietet hierbei prinzipiell einen Ansatzpunkt, um diese in das technische Management miteinzubinden. Allerdings lassen sich bis auf die Aktivität „Erstellung eines Messagentskeletts“ die Aktivitäten nicht ohne Interaktion mit dem Entwickler ausführen.

  - *Policy-Spezifikation:* ✓
    Dieser Teilprozess läßt sich durchaus mit Hilfe von Managementpolicies unterstützen, wenn auch keine Automatisierung des Managementvorgangs möglich ist. Sofern sich das Tarifmodell ändert, indem eine neue Abrechnungseinheit hinzugefügt wird ($\simeq$ Ereignis), für die es noch *keinen* Sensor für den gegebenen Dienst gibt ($\simeq$ Constraint), sollte automatisch innerhalb der Entwicklungsumgebung ($\simeq$ Target) ein Messagentskelett mit dem Namen der geänderten Abrechnungseinheit erzeugt werden ($\simeq$ Aktionen) und ein Entwickler ($\simeq$ Subject) benachrichtigt werden, dass ein geeigneter Sensor zu implementieren ist.

- ***Data Rollout*** *(siehe Abschnitt 4.3.6, Seite 96)*

  - *Interagierende Akteure:* Repository, Administrator, Kunde
    Bei diesen Akteuren handelt es sich um Rollen, die sowohl von Personen (Administrator, Kunde) als auch von HW/SW–Systemen (Repository) eingenommen werden können.
  - *Aktivitäten:* Grundsätzlich werden die im Teilprozess identifizierten Aktivitäten mit Unterstützung von technischen Hilfsmitteln (z.B. Datenübertragung per Email) durchgeführt, auf die von Seiten des technischen Managements zugegriffen werden kann. Je nach vereinbarter Art und Weise der Durchführung des Data Rollouts, kann dieser Teilprozess vollkommen automatisiert ablaufen (z.B. bei enger Kopplung der beteiligten Systeme und automatisch ablaufenden Konvertierungsroutinen) oder aber auch durch manuelles Eingreifen der beteiligten Rollen.
  - *Policy-Spezifikation:* ✓
    Dieser Teilprozess läßt sich je nach Automatisierungsgrad des Prozessablaufs mit Hilfe von Policies steuern und überwachen. Sofern der Data Rollout der Daten von Kundenseite ($\simeq$ Subject) in regelmäßigen Abständen stattfinden soll ($\simeq$ Ereignis), wird die Datenübertragung und die Datenkonvertierung angestoßen ($\simeq$ Aktion) und in den eigenen Datenbestand/Repository ($\simeq$ Target) eingefügt.

- ***Verteilung der Messkomponenten*** *(siehe Abschnitt 4.3.7, Seite 97)*

  - *Akteur:* Administrator
    Hierbei handelt es sich um eine Rolle, die üblicherweise von einer Person eingenommen wird.
  - *Aktivitäten:* Die im Teilprozess identifizierten Aktivitäten werden mit Unterstützung von technischen Hilfsmitteln durchgeführt (z.B. Sensor–Inkarnation durch Programmaufruf/Einschalten eines Probes, etc.), auf die i.d.R. von Seiten des technischen Managements zugegriffen werden kann. Je nach Automatisierungsgrad des Teilprozesses, der vom Sensortyp (Software–/Hardware–Probe) und Verteilungsstrategie (durch Software–Verteilung, Vorort–Installation, etc.) abhängig ist, ist eine Automatisierung des Managementvorgangs möglich.

 Als Buch erhältlich im Verlag Dr. Hut, München, www.dr.hut-verlag.de (ISBN 3-934767-97-4)

  - *Policy-Spezifikation:* ✓
    Dieser Teilprozess läßt sich mit Hilfe von Policies unterstützen und u.U. sogar vollständig automatisieren. Jeder Instantiierung ($\simeq$ Ereignis) einer zu überwachenden Komponente ($\simeq$ Constraint) ist ein Sensor ($\simeq$ Target) zuzuordnen und entsprechend des festgestellten Messorts zu verteilen ($\simeq$ Aktion).

- ***Konfiguration des Abrechnungssystems*** *(siehe Abschnitt 4.3.8, Seite 99)*
  - *Akteur:* Administrator
    Hierbei handelt es sich um eine Rolle, die von einer Person eingenommen wird.
  - *Aktivitäten:* Die im Teilprozess identifizierten Aktivitäten werden auf Ressourcen ausgeführt, die i.d.R. vom technischen Management einbezogen werden können. In Abhängigkeit von der Dienstvereinbarung werden die an der Abrechnung beteiligten Komponenten konfiguriert. Die jeweilig durchzuführenden Konfigurationseinstellungen sind in Abschnitt 4.3.8 aufgeführt, so dass auf eine erneute Aufstellung an dieser Stelle verzichtet wird.
  - *Policy-Spezifikation:* ✓✓
    Dieser Teilprozess läßt sich von Managementseite mit Hilfe von Policies unterstützen und u.U. sogar vollständig automatisieren. In Abhängigkeit von Änderungen an der Dienstvereinbarung ($\simeq$ Ereignis) wird die davon betroffene ($\simeq$ Constraint) Komponente ($\simeq$ Target) mit neuen Einstellungen konfiguriert ($\simeq$ Aktion).

- ***Dienstanalyse*** *(siehe Abschnitt 4.3.9, Seite 105)*
  - *Akteur:* Entwickler
    Hierbei handelt es sich um eine Rolle, die von einer Person eingenommen wird.
  - *Aktivitäten:* Die im Teilprozess identifizierten Aktivitäten können ausschließlich vom Akteur durchgeführt werden.
  - *Policy-Spezifikation:* $\lightning$
    Dieser Teilprozess läßt sich nicht mit Hilfe von Policies unterstützen, da zum einen, die Prozessaktivitäten einen niedrigen Automatisierungsgrad aufweisen, und zum anderen, die Aktivitäten nicht auf Basis von technischen Hilfsmitteln durchgeführt werden, die im Einflussgebiet des technischen Managements stehen. Es ist lediglich möglich, den Akteur durch regelmäßige, automatisch generierte Nachrichten an die Durchführung dieses Teilprozesses zu erinnern.

- ***Kostenermittlung*** *(siehe Abschnitt 4.3.10, Seite 106)*
  - *Interagierende Akteure:* IT Finance Manager, Entwickler
    Hierbei handelt es sich um Rollen, die von Personen eingenommen werden.
  - *Aktivitäten:* Die im Teilprozess identifizierten Aktivitäten werden ausschließlich von den beteiligten Akteuren durchgeführt.

 - *Policy-Spezifikation:* ↯
 Dieser Teilprozess eignet sich nur in sehr beschränkter Art und Weise dazu, diesen mit Hilfe von Policies zu unterstützen. Der Teilprozess weist einen geringen Automatisierungsgrad auf und die Aktivitäten werden auf Basis von Ressourcen durchgeführt, die sich dem Einfluss des technischen Managements entziehen. Allerdings ist es möglich, eine regelmäßige Durchführung dieses Teilprozesses durch automatisch generierte Nachrichten sicherzustellen, die an die verantwortlichen Akteure versendet werden.

- ***Nutzungserfassung*** *(siehe Abschnitt 4.3.11, Seite 108)*
  - *Interagierende Akteure:* Repository, Kollektor, Messagent, Sensor
 Diese Akteure werden von technischen Komponenten eingenommen.
  - *Aktivitäten:* Die im Teilprozess identifizierten Aktivitäten werden von den technischen Komponenten selbstständig ohne Interaktion mit weiteren Akteuren, wie z.B. einem Administrator, durchgeführt und weisen damit einen hohen Automatisierungsgrad auf.
  - *Policy-Spezifikation:* ✓✓
 Die Ausführung des Teilprozesses läßt sich mit Hilfe von Policies unterstützen. Es ist sogar ein vollständig automatisches Management des Teilprozesses möglich. Je nach *Integrationsgrad* der Integrationsschicht, können sich die Policies für diesen Teilprozess darauf beschränken, die jeweilig beteiligten Komponenten ($\simeq$ Target) zu den vereinbarten Betriebszeiten ($\simeq$ Ereignis, Constraint) zu starten ($\simeq$ Aktion). Die Art und Weise der *Durchführung* der im Teilprozess identifizierten Aktivitäten wird in solch einem Fall durch die Konfigurationspolicies gesteuert werden, so dass die Interaktionen zwischen den beteiligten Akteuren völlig autonom ablaufen würden. Alternativ ist es auch denkbar, für jeden *Interaktionsübergang* zwischen den Akteuren eine eingeständige Policy zu spezifizieren. In diesem Fall interagieren die beteiligten Komponenten nicht mehr autonom miteinander, sondern jede Interaktion wird durch eine entsprechende Policy explizit angestoßen und kontrolliert. Durch diese Verfahrensweise bieten sich zwar aus Sicht des Managements wesentlich feingranularere Einflussmöglichkeiten, jedoch steigt damit auch der Aufwand für das Management.

- ***Gebührenberechnung*** *(siehe Abschnitt 4.3.12, Seite 111)*
  - *Interagierende Akteure:* Repository, Charging Komponente, Kollektor
 Diese Akteure werden von technischen Komponenten eingenommen.
  - *Aktivitäten:* Die im Teilprozess identifizierten Aktivitäten werden von den technischen Komponenten selbstständig durchgeführt.
  - *Policy-Spezifikation:* ✓✓
 Die Ausführung des Teilprozesses läßt sich mit Hilfe von Policies unterstützen und es ist sogar ein vollständig automatisches Management des Teilprozesses möglich. Die Spezifikation von Policies für diesen Teilprozess ähnelt der des Nutzungserfassungsteilprozesses. Auch in diesem Fall ist die Spezifikation von Policies vom Integrations-

 Als Buch erhältlich im Verlag Dr. Hut, München, www.dr.hut-verlag.de (ISBN 3-934767-97-4)

grad der Integrationsschicht abhängig, d.h. dass die gleichen Anmerkungen wie beim Nutzungserfassungsteilprozess gelten.

- ***Rechnungsstellung*** *(siehe Abschnitt 4.3.13, Seite 113)*
  - *Interagierende Akteure:* Rechnungsempfänger, Repository, Billing System, Charging Komponente
    Mit Ausnahme des Rechnungsempfängers, welches einer von einer Person eingenommenen Rolle entspricht, werden die Akteure von technischen Komponenten eingenommen.
  - *Aktivitäten:* Die im Teilprozess identifizierten Aktivitäten werden von den technischen Komponenten selbstständig durchgeführt. Lediglich das Empfangen der ausgestellten Rechnung bedarf einer Interaktion mit einer Person, die aber i.d.R. keinen Einfluss auf die Durchführung der übrigen Aktivitäten hat[11].
  - *Policy-Spezifikation:* ✓✓
    Dieser Teilprozess läuft autonom ab und dessen Management läßt sich mit Hilfe von Policies automatisiert realisieren. Auch in diesem Fall ist die Policy–Spezifikation vom Integrationsgrad der Integrationsschicht abhängig, d.h. dass die gleichen Anmerkungen gelten wie beim Nutzungserfassungsteilprozess.
- ***Reporting*** *(siehe Abschnitt 4.3.14, Seite 114)*
  - *Interagierende Akteure:* Reportempfänger, Repository, Reporting Tool, Messkomponente
    Mit Ausnahme des Reportempfängers, welches üblicherweise einer von einer Person eingenommenen Rolle entspricht, werden die Akteure von technischen Komponenten eingenommen.
  - *Aktivitäten:* Die im Teilprozess identifizierten Aktivitäten werden von den technischen Komponenten selbstständig durchgeführt. Lediglich das Empfangen der Reports bedarf einer Interaktion mit einer Person, die aber i.d.R. keinen Einfluss auf die Durchführung der übrigen Aktivitäten hat.
  - *Policy-Spezifikation:* ✓✓
    Dieser Teilprozess läßt sich vollständig mit Hilfe von Policies automatisiert steuern. Ansonsten gelten die gleichen Anmerkungen wie beim Nutzungserfassungsteilprozess, d.h. dass die Spezifikation von Policies vom Integrationsgrad der Integrationsschicht abhängig ist.
- ***Rechnungsprüfung*** *(siehe Abschnitt 4.3.15, Seite 115)*
  - *Interagierende Akteure:* Rechnungsprüfer, Reporting Tool, Charging Komponente, Billing System

[11] Außer der Empfang der Rechnung muss durch den Rechnungsempfänger bestätigt werden.

Mit Ausnahme des Rechnungsprüfers, welches üblicherweise (aber nicht notwendigerweise) einer von einer Person eingenommenen Rolle entspricht, werden die Akteure von technischen Komponenten eingenommen.

- *Aktivitäten:* Die im Teilprozess identifizierten Aktivitäten werden, mit Ausnahme des Versendens der für die Überprüfung notwendigen Daten, vom Rechnungsprüfer durchgeführt.
- *Policy-Spezifikation:* ✓/✓✓
  Die Policy–Spezifikation hängt vom Automatisierungsgrad des Teilprozesses ab. Sofern der Rechnungsprüfer ($\simeq$ Target) einer technischen Komponente entspricht (wie bspw. bei Conformance–Tests), läßt sich der Prozess vollständig automatisiert durch Policies steuern und überwachen. Werden die Aktivitäten allerdings von einer Person durchgeführt, so besteht kaum eine Möglichkeit, den Teilprozess aus Sicht des technischen Managements zu beeinflussen. In diesem Fall kann lediglich das Anstoßen dieses Teilprozesses durch das regelmäßige ($\simeq$ Ereignis) Generieren einer Nachricht an den Rechnungsprüfer unterstützt werden ($\simeq$ Aktion). Sofern der Rechnungsprüfer technische Komponenten zum Durchführen der Überprüfung einsetzt, wie z.B. das erneute Errechnen eines in Rechnung gestellten Betrags, kann dies evtl. teilweise durch Policies gesteuert werden.

• ***Zahlungsüberwachung*** *(siehe Abschnitt 4.3.16, Seite 116)*

- *Interagierende Akteure:* Zahlungsüberwachungskomponente, Billing System, Buchungssystem
  Die Akteure werden ausnahmslos von technischen Komponenten eingenommen.
- *Aktivitäten:* Die im Teilprozess identifizierten Aktivitäten werden autonom von den beteiligten Akteuren durchgeführt.
- *Policy-Spezifikation:* ✓✓
  Die Spezifikation von Policies zum automatisierten Managements des Vorgangs ist möglich. Für die Policy–Deklaration gelten die gleichen Aussagen wie für die Nutzungserfassung: soll der Vorgang vollständig autonom ohne den Eingriff durch das Management ablaufen, so werden die notwendigen Einstellungen durch Konfigurationspolicies vorgenommen. Ansonsten können Policies spezifiziert werden, die bei Überschreiten des Zahlungsintervalls ($\simeq$ Ereignis) und bei Eingehen von Zahlungsbeträgen ($\simeq$ Ereignis) den Teilprozess der Überwachungskomponente ($\simeq$ Target) anstoßen ($\simeq$ Aktion).

• ***Deinstallation*** *(siehe Abschnitt 4.3.17, Seite 117)*

- *Akteur:* Administrator
  Der Akteur entspricht einer Rolle, die von einer Person eingenommen wird.
- *Aktivitäten:* Die Aktivitäten werden eingeständig durch den einzigen Akteur durchgeführt.

 Als Buch erhältlich im Verlag Dr. Hut, München, www.dr.hut-verlag.de (ISBN 3-934767-97-4)

– *Policy-Spezifikation:* ✓✓
  Es ist eine Policy–Spezifikation möglich, wobei es sich hierbei z.T. bereits um Meta–Policies handelt: die für den Kunden und den bereitgestellten Dienst spezifizierten Policies werden deaktiviert und u.U. gelöscht. Zum anderen müssen Managementaktionen, welche die Konfiguration des Abrechnungsprozesses und den damit verbundenen prozessrealisierenden Komponenten betreffen, rückgängig gemacht werden. Dieser Sachverhalt wird mit Hilfe von Deinstallationspolicies ausgedrückt.

Zusammenfassender Überblick

In Tabelle 5.7 werden die Teilprozesse auf Basis der im vorhergehendem Abschnitt durchgeführten Analyse und getroffenen Aussagen kategorisiert. Die hierbei ermittelten vier Kategorien teilen die Teilprozesse bezüglich des durch Policies erreichbaren *Managementgrades* ein. Der Managementgrad bestimmt den Anteil eines Teilprozesses, der durch Policies automatisiert gesteuert und überwacht werden kann. Allgemein kann festgestellt werden, dass je höher der *Automatisierungsgrad* eines Teilprozesses bezogen auf dessen Ablauf ist, desto höher ist auch dessen Managementgrad. In der nachfolgenden Aufstellung werden die vier identifizierten Kategorien erklärt:

Vollständiges Management des Teilprozesses Für alle in dieser Kategorie aufgeführten Teilprozesse ist eine Spezifikation von Policies in einer Art und Weise möglich, dass die in der Policy deklarierten Managementaktionen den vollständigen Teilprozess eigenständig steuern und überwachen. In diese Kategorie fallen demnach alle Teilprozesse, die einen hohen Automatisierungsgrad aufweisen. Die Teilprozesse, die das Abrechnungssystem realisieren, werden naturgemäß alle dieser Kategorie zugeordnet. Für diese Teilprozesse existieren in Abhängigkeit vom Integrationsgrad der Integrationsschicht grundsätzlich zwei Möglichkeiten, wie von Managementseite aus die Ausführung des Teilprozesses kontrolliert werden kann:

- *Prozesssteuerung hauptsächlich durch Konfigurationsmanagementpolicies*
  Dies bedeutet, dass die Art und Weise des Ablaufs des Teilprozesses durch entsprechende Konfigurationspolicies in der Installations– & Testphase beeinflusst wird und nur sehr allgemein gehaltene Managementpolicies für die Betriebsphase spezifiziert sind, wie z.B. das explizite Starten und Anhalten der Prozessausführung. Konsequenterweise laufen dann die Interaktionen zwischen den prozessrealisierenden Komponenten eigenständig und ohne Kenntnis und Einflussnahme des Managements ab.
- *Prozesssteuerung hauptsächlich durch Betriebsmanagementpolicies*
  Hierbei sehen die Konfigurationspolicies nur absolut notwendige Grundeinstellungen vor. Die Teilprozesse werden durch geeignete Managementpolicies in der Betriebsphase des Dienstlebenszyklus gesteuert und überwacht. Folglich laufen die Interaktionen zwischen prozessrealisierenden Komponenten nicht mehr eigenständig ab, sondern werden durch die in Policies deklarierten Managementaktionen explizit angestoßen

und damit gesteuert und überwacht. Dies ermöglicht i.d.R. eine weitaus feingranularere Steuerung des Abrechnungsvorgangs in der Betriebsphase, allerdings steigt auch der Managementaufwand sowohl bei der Policy–Spezifikation als auch bei der Realisierung der Integrationsschicht.

Teilweises Management des Teilprozesses Für alle in dieser Kategorie aufgeführten Teilprozesse ist eine Spezifikation von Policies zwar möglich, allerdings können nur für einen Teil der Prozessausführung Managementaktionen in Form von Policies ausgedrückt werden. Konsequenterweise wird nur ein Teil des Teilprozesses mittels Policies eigenständig gesteuert und überwacht. Für den übriggebliebenen, nicht durch Policies gesteuerten Rest des Teilprozesses ist eine Interaktion mit den am Teilprozess beteiligten Akteuren durch den Manager notwendig. Sofern diese Art der Interaktionen eines Teilprozesses automatisiert werden und damit ein explizites Eingreifen von Managementseite nicht (mehr) notwendig ist, kann ein derartiger Teilprozess der 1. Kategorie zugeordnet werden.

nur Prozessinitiierung In dieser Kategorie befinden sich ausschließlich Teilprozesse, deren Ausführung lediglich mittels Policies „gewünscht" werden kann. Das bedeutet, es besteht aus Managementsicht keinerlei Möglichkeit der automatisierten Überprüfung/Kontrolle der Ausführung des Teilprozesses. Hierbei handelt es sich um Policies, welche die Akteure per einfacher Nachrichten (bspw. per Email) zur Prozessausführung auffordern.

keine sinnvolle Policy–Spezifikation möglich Diese Kategorie beinhaltet Teilprozesse, für welche eine Policy–Spezifikation nicht sinnvoll erscheint. Das heißt, dass weder der Teilprozess in einer automatisierten Form vorliegt und somit eine Automatisierung des Managements wie bei den ersten beiden Kategorien nicht erreichbar ist, noch eine einfache Prozessinitiierung wie bei der dritten Kategorie sinnvoll ist, da die Zeitpunkte zur Ausführung der Teilprozesse nicht angegeben werden können.

Policy–Management des Teilprozesses			*Policies*
vollständig	*teilweise*	*nur Prozessinitiierung*	*nicht sinnvoll*
Konfiguration Nutzungserfassung Gebührenberechnung Rechnungsstellung Reporting Zahlungsüberwachung Deinstallation	MK*–Implementierung Data Rollout MK*–Verteilung Rechnungsprüfung	Dienstanalyse Kostenermittlung	Kundenanalyse Kostenprognose Tarifierung
*MK: Messkomponente			

Tabelle 5.7: Teilprozesseinteilung für die Policy–Spezifikation

Anhand der Tabelle 5.7 kann ersehen werden, dass das Management der Teilprozesse in den beiden links notierten Kategorien mittels Policies automatisch bzw. semi–automatisch ablau-

 Als Buch erhältlich im Verlag Dr. Hut, München, www.dr.hut-verlag.de (ISBN 3-934767-97-4)

fen kann und dies damit zu einer spürbaren Aufgabenerleichterung auf Managementseite beiträgt. Bei den Teilprozessen in den beiden rechten Kategorien handelt es sich ausschließlich um Teilprozesse, deren Akteure durch Personen eingenommene Rollen sind und somit eine Automatisierung des Teilprozess–Managements nicht ohne weiteres mit den hierfür vorgesehenen Mitteln des technischen Managements möglich ist. Dies ist allerdings nicht als Manko des vorgestellten Ansatzes zu verstehen, da diese Art der Teilprozesse im Fokus eines vollständig anderen Managements (nämlich des Personal–/Workflow–/Businessmanagement) stehen, das außerhalb des in dieser Arbeit betrachteten technischen Managements steht. Allerdings ermöglicht die in dieser Arbeit vorgestellte Prozessanalyse (siehe Kapitel 4) eine klare Einteilung der Teilprozesse in die eingeführten vier Kategorien und zeigt damit den Zuständigkeitsbereich des technischen Managements auf.

Teilprozess \ Entität	IT Finance Manager	Entwickler	Administrator	Rechnungsprüfer	Kunde	Rechnungsempfänger	Reportempfänger	Tarifmodell	Rechnung	Report	Dienst	Messkomponente	Kollektor	Charging Komponente	Billing System	Reporting Tool	Repository	Entwicklungsumgebung	Dienstimplementierung
Implementierung		○						▽										✖	
Data Rollout					○												✖		
Verteilung			✖								○	✖					▽		▽
Konfiguration								▽	▽	▽	○	✖	✖	✖	✖	✖			
Nutzungserfassung											○	✖	✖						
Gebührberechnung					○								▽	✖					
Rechnungsstellung						○								▽	✖				
Rechnungsprüfung				✖	○						○				▽	▽			
Reporting							○									✖			
Deinstallation			○					▽	▽	▽	○	✖	✖	✖	✖	✖	✖		▽
Dienstanalyse		✖									○								▽
Kostenermittlung	✖										○								▽
	Rollen							Dienst–vereinbarung				Dienst– und Mgmt.–implementierung							

○ Subject ✖ Target ▽ Zustandsänderung als Ereignis

Abbildung 5.6: Übersichtstabelle für die Spezifikation von Abrechnungspolicies

In Abbildung 5.6 ist für diejenigen Teilprozesse, die durch Policies automatisierbar zu managen sind, eine Übersicht über die jeweiligen Subjects, Targets und auslösenden Ereignisse dargestellt, die in der Analyse im vorhergehenden Abschnitt identifiziert wurden. Sofern es sich bei dem Target um eine Rolle handelt, bedeutet dies, dass die Aktion immer eine Benachrichtigung der Rolleneinnehmer ist. Sind mehrere Targets angegeben, so müssen die Managementaktionen entweder in unterschiedlichen Policies ausgedrückt werden oder die Aktionsdeklaration enthält die explizite Nennung des nicht als Default–Target angegebenen Objekts.

Es muss betont werden, dass die in Abbildung 5.6 dargestellte Übersicht als Orientierungshilfe dienen soll und nicht als verbindliche, unumstößliche Spezifikation zu verstehen ist. Wie bereits zu Beginn des Abschnitts 5.4.2 festgestellt wurde, ist die Spezifikation von *konkreten* Policies nur bei Vorhandensein einer Dienstvereinbarung und einer Dienst(management)implementierung möglich. Somit können Policies in Abhängigkeit der dienstspezifischen Implementierungen und Vereinbarungen in Subject und Target durchaus variieren.

5.4.3 Diskussion der Spezifikation von Meta–Policies

Meta–Policies werden zum Management von „normalen“ Policies verwendet und sollen demnach den Policy–Ersteller und Administrator bei ihren in Zusammenhang mit Policies stehenden Aufgaben unterstützen. Das mit Meta–Policies verfolgte Ziel ist, die für das Management von Policies notwendigen Aufgaben möglichst zu automatisieren, um einerseits damit eine Arbeitserleichterung zu schaffen und anderseits, um Fehlspezifikationen und Inkonsistenz zu vermeiden. Folglich werden Meta–Policies analog zu den bisher betrachteten Abrechnungsmanagement–Policies, die den Abrechnungslebenszyklus steuern und überwachen, dazu eingesetzt, den *Policy–Lebenszyklus* in einem gegebenen Policy–Managementsystem geeignet zu managen.

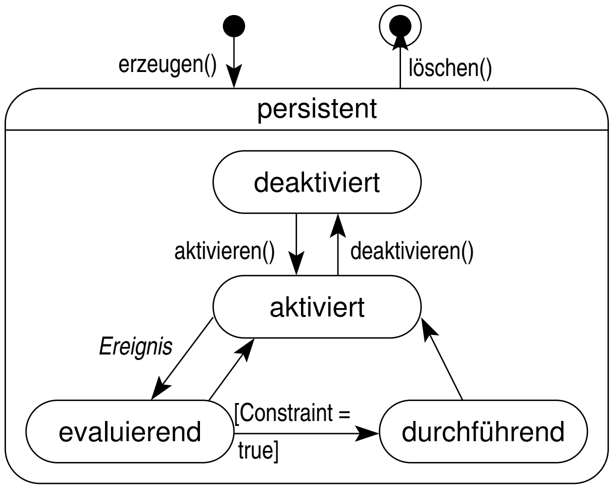

Abbildung 5.7: Zustandsdiagramm einer Policy

In Abbildung 5.7 ist der Lebenszyklus einer Policy als Zustandsdiagramm dargestellt. Dieser beginnt mit dem Erzeugen und damit persistenten Speichern einer Policy, nachdem der Policy–Ersteller diese spezifiziert hat. Um die Policy „scharf“ zu stellen, muss diese explizit aktiviert werden. Nach der Aktivierung arbeitet die Policy bis zu ihrer Deaktivierung vollkommen autonom, d.h. ohne den Eingriff des Policy–Erstellers oder eines Administrators. Bei Eintreten relevanter Ereignisse wird die Anwendbarkeit der Policy, u.a. durch Evaluierung der spezifizierten Constraints, überprüft. Fällt die Evaluierung positiv aus, so werden die spezifizierten Aktionen durchgeführt. Nach Beendigung der Ausführung von Managementaktionen fällt die Policy wieder in den aktivierten Zustand zurück und wartet bis zu ihrer Deaktivierung auf das Eintreffen weiterer spezifizierter Ereignisse. Im deaktivierten Zustand kann die Policy wieder gelöscht werden.

 Als Buch erhältlich im Verlag Dr. Hut, München, www.dr.hut-verlag.de (ISBN 3-934767-97-4)

Zentrale Frage ist nun, was bei Meta–Policies als Subject, Target, Aktion, evaluationsauslösendes Ereignis und Constraint zu spezifizieren ist. Hierfür werden im Nachfolgenden die Policy–Felder einzeln auf ihre für Meta–Policies relevante Bedeutung analysiert.

Subject Eine Meta–Policy bezieht sich immer auf eine oder mehrere Policies, für welche die im Aktionsteil deklarierten Managementaktivitäten anzuwenden sind. Somit wird im Subject–Feld einer Meta–Policy der Identifikator einer Policy oder einer Policy–Gruppe angegeben.

Target Das Policy–Target enthält immer das tatsächliche Ziel der innerhalb der Policy deklarierten Managementaktionen. Im vorliegenden Fall ist dies immer die Policy–Verwaltungsinstanz innerhalb des Policy–Managementsystems, welche Methoden zur Steuerung des Policy–Lifecycles bereitstellt. Diese Policy–Verwaltungsinstanz ist, wie bereits innerhalb dieses Kapitels erwähnt, als CORBA–Server realisiert und kann damit durch die Angabe des registrierten CORBA–Namens im Target–Feld deklariert werden.

Aktion Wie bereits zu Beginn dieses Abschnitts erläutert wurde, gilt es mit Hilfe von Meta–Policies die bzgl. von Policies durch den Policy–Ersteller resp. den Administrator durchzuführenden steuernden Eingriffe geeignet zu unterstützen, wenn möglich gar zu automatisieren. Demnach sind die Aktionen Erzeugen, Löschen, Aktivieren und Deaktivieren bezüglich einer Policy geeignet durch die entsprechenden Methoden der Schnittstelle der Policy–Verwaltungsinstanz zu spezifizieren.

Ereignis Das evaluations– und letztendlich auch aktionsauslösende Ereignis ist der wohl interessanteste Teil der Meta–Policy–Spezifikation, da damit festgelegt wird, zu welchen Zeitpunkten und in welchem Kontext Managementeingriffe bzgl. der herkömmlichen Policies notwendig sind. Idealerweise wäre nach der Spezifikation von Policies kein zusätzlicher Managementeingriff mehr erforderlich und damit die Deklaration von Meta–Policies überflüssig. Dies ist genau dann der Fall, wenn innerhalb des Gesamtsystems, bestehend aus der Abrechnungsmanagementimplementierung, der Abrechnungsprozessimplementierung und der Dienstimplementierung, *keine* Änderungen mehr auftreten, die nicht bereits durch die herkömmlichen Policies abgefangen werden[12] (siehe hierzu Abschnitt 5.4.2). Dieser (einfachste) Fall ist aber nie gegeben, da zumindest die Gültigkeit der Dienstvereinbarung zeitlich begrenzt ist und der mit der Beendigung der Dienstleister–Dienstnehmer–Beziehung zusammenhängende Abbau des Dienstes auch immer mit dem Abbau des dazugehörigen Managements verbunden ist. Im vorliegenden Fall wären dann die für den Kunden aktiven Policies zu deaktivieren und ggf. zu löschen. Verallgemeinernd muss im Grunde untersucht werden, welche *Änderungen* innerhalb des Gesamtsystems welche Konsequenzen für existierende und noch nicht existierende Policies haben. Für diese Analyse kann wieder das *Abrechnungsdienstmodell* aus Abschnitt 4.4 herangezogen werden. Insbesondere die in Abbildung 4.23 auf Seite 123 dargestellte *Vereinbarungssicht* und in Abbildung 4.24 auf Seite 126 visualisierte *Realisierungssicht* des Abrechnungsdienstmodells bieten Hinweise auf die Auswirkungen von Änderungen.

[12]Grundsätzlich kann das Erzeugen von bspw. Usage Records und Rechnungen als Änderung innerhalb des Gesamtsystems aufgefasst werden. Die Reaktion auf derartige Änderungen wird allerdings mittels Policies ausgedrückt.

Insgesamt können die folgenden Fälle bzgl. der möglichen Änderungen unterschieden werden:

Dienstnehmer–getrieben In Abschnitt 2.2.2 wurde bereits erläutert, dass eine wesentliche Eigenschaft in Outsourcing–Szenarios das Zulassen von *Änderungen an der Dienstvereinbarung* ist. Wird hierzu die Vereinbarungs– und Realisierungssicht des Abrechnungsdienstmodells herangezogen, so ergeben sich die folgenden, für die Abrechnung und das Abrechnungsmanagement relevanten Änderungen an der Dienstvereinbarung:

- *Änderung des Tarifmodells*
  Wie aus der Realisierungssicht des Abrechnungsdienstmodells in Abbildung 4.24 (siehe hierzu die verbindende Assoziation) ersehen werden kann, beeinflusst das innerhalb der Dienstvereinbarung enthaltene Tarifmodell die zur Umsetzung des Abrechnungsvorgangs eingesetzten Messkomponenten. Die in Tabelle 4.1 auf Seite 92 dargestellte Aufstellung zeigt, dass eine Abrechnungseinheit den zu instrumentierenden Dienstbestandteil bestimmt, während die Preisfunktion den Messort bestimmt. Nachfolgend werden die Auswirkungen von Änderungen am Tarifmodell für das Abrechnungsmanagement diskutiert:

  – *Ändern einer Preisfunktion*
    Eine Abrechnungseinheit ist immer mit einer Preisfunktion assoziiert, welche die Gebühr für eine Abrechnungseinheit in Kontext von weiteren Parametern wie erfahrene Dienstgüte, etc. bestimmt. Wie bereits erwähnt wurde, bestimmt die Preisfunktion insbesondere den Messort. Ändert sich die Preisfunktion, muss demnach überprüft werden, ob die existierenden *Verteilungspolicies* deaktiviert werden müssen und neue Verteilungspolicies zu spezifizieren sind. Dies ist nicht ohne Interaktion mit einem Policy–Ersteller möglich. Zudem muss das neue Tarifmodell in der entsprechenden Repository aktualisiert werden.
  – *Hinzufügen einer Abrechnungseinheit/Preisfunktion*
    Falls noch kein geeigneter Sensor und Messagent für die Abrechnungseinheit existiert, so müssen jeweils neue implementiert werden. Dieser Sachverhalt wird allerdings bereits mit einer Implementierungspolicy ausgedrückt und muss nicht mehr betrachtet werden. Jedoch müssen für neue Messkomponenten geeignet neue *Konfigurationspolicies* erstellt werden. Dies läßt sich allerdings nicht ohne Interaktion mit dem Policy–Ersteller durchführen und kann damit nicht vollautomatisch durchgeführt werden.
  – *Entfernen einer Abrechnungseinheit/Preisfunktion*
    In diesem Fall müssen die dafür vorgesehenen Messkomponenten abgebaut werden, welches bereits durch Deinstallationspolicies ausgedrückt wird. Zusätzlich müssen die zur Messkomponente gehörenden *Konfigurationspolicies* und *Nutzungserfassungspolicies* deaktiviert und u.U. gelöscht werden. Dies läßt sich mittels einer Meta–Policy vollautomatisch durchführen.

 Als Buch erhältlich im Verlag Dr. Hut, München, www.dr.hut-verlag.de (ISBN 3-934767-97-4)

- *Änderung der vereinbarten Dienstfunktionalität*
  Die Dienstvereinbarung sieht Festlegungen sowohl bezüglich der Nutzungsfunktionalität als auch bezüglich der Managementfunktionalität vor. Werden an der *Nutzungsfunktionalität* Änderungen vorgenommen, die sich auf die Dienstimplementierung auswirken, so muss überprüft werden, ob hierfür neue Messkomponenten implementiert werden müssen. Dieser Fall wird allerdings bereits durch Implementierungspolicies abgedeckt, so dass hierfür *keine* Spezifikation von Meta–Policies notwendig ist.

  In der Dienstvereinbarung wird zusätzlich explizit die Managementfunktionalität festgehalten und somit auch die Funktionalität der Abrechnung und des Abrechnungsmanagements. Bezüglich der Abrechnungsmanagementfunktionalität werden insbesondere Vereinbarungen über die Rechnungsstellung, Reportausstellung und der Zahlungsweise getroffen. Dies betrifft *direkt* aussschließlich die Teilprozesse der Betriebsphase, da diese die eigentlich vereinbarte Funktionalität bereitstellen[13]. Werden diesbezüglich Änderungen durchgeführt, so existieren, je nach Integrationsgrad der Integrationsschicht (siehe hierzu auch die abschließenden Anmerkungen des Abschnitts 5.4.2 auf Seite 171), die folgenden zwei Möglichkeiten, um wieder einen konsistenten Zustand zu erlangen:

  - *Änderung der Konfiguration*
    Sofern die tatsächlichen Akteure der Teilprozesse in der Betriebsphase hauptsächlich autonom und mit nur wenigen bis keinen Interaktionen mit dem Management ablaufen, so bedeutet dies, dass die entsprechenden Managementziele (hauptsächlich) durch Konfigurationspolicies ausgedrückt werden. Somit müssen die entsprechenden Konfigurationspolicies der betroffenen Komponenten angepasst werden. Dies bedarf einer Deaktivierung bestehender Konfigurationspolicies, welches durch Meta–Policies vollautomatisch durchgeführt werden kann, und eine Erstellung und Aktivierung neuer Konfigurationspolicies, welches nicht ohne Interaktion mit einem Policy–Ersteller durchgeführt werden kann.
  - *Änderung der direkten Einflussnahme während des Betriebs*
    Werden die Teilprozesse in der Betriebsphase direkt durch Policies gesteuert, so müssen die betreffenden, bereits bestehenden Policies deaktiviert werden und durch neue ersetzt werden. Hierbei ist ein wesentlich feingranularer Eingriff von Managementseite aus möglich, der den laufenden Betrieb wesentlich weniger belastet.
  - *Austausch einer prozessrealisierenden HW/SW–Komponente*
    Falls die Anforderungen an die tatsächliche prozessrealisierende Komponente, die aus der neu festgelegten Managementfunktionalität resultieren, nicht mehr erfüllt werden können, muss diese ausgetauscht werden. Dies entspricht einer Deinstallation und einer erneuten Installation einer Komponente. Die Deinstallation kann

[13] Die übrigen Teilprozesse der anderen Phasen werden natürlich zum Erreichen der Betriebsphase benötigt und sind somit in Teilen indirekt betroffen.

durch explizite Aktivierung der entsprechenden Deinstallationspolicy erreicht werden. Die Installation einer neuen Komponente ist allerdings nicht ohne einen Administrator durchzuführen. Damit verbunden ist auch die Spezifikation von neuen Konfigurations–/ und Betriebspolicies, welches ebenfalls nicht ohne Interaktion mit einem Policy–Ersteller bewerkstelligt werden kann.

Dienstleister–getrieben Der Dienstleister ist daran interessiert, den Betrieb und das Management der bereitgestellten Dienste hinsichtlich Effizienz und Effektivität zu optimieren. Konsequenterweise werden bei Durchsetzung von Optimierungsmaßnahmen auch Teile der Dienst– und Dienstmanagementimplementierung restrukturiert [BRS 02]. Insgesamt ergeben sich bei den Änderungen die nachfolgenden Auswirkungen:

- *Änderung der Dienstmanagementimplementierung*
  Wie in der Realisierungssicht des Dienstmodells in Abbildung 4.24 visualisiert ist, ist der für die Abrechnung relevante Teil der Dienstmanagementimplementierung aufgeteilt in die Abrechnungsprozess– und in die Abrechnungsmanagementimplementierung. Die Abrechnungsmanagementimplementierung ist im vorliegenden Fall durch die Spezifikation und maschinelle Interpretation von Policies realisiert, so dass diesbezügliche Änderungen in eine Respezifikation von Policies münden. Dies kann nur durch den Policy–Ersteller durchgeführt werden und eine Meta–Policy–Spezifikation ist in diesem Fall nicht möglich. Bezogen auf die Abrechnungsprozessimplementierung kann folgende Änderung durchgeführt werden:
  Austausch von prozessrealisierenden HW/SW–Komponenten
  Um die Güte des Abrechnungsmanagements zu erhöhen, ist eine häufig angewendete Optimierungsmaßnahme der Austausch einer prozessrealisierende HW/SW–Komponente. Sofern sich die Anforderungen bzgl. der gewünschten Managementfunktionalität[14], wie sie in der Dienstvereinbarung zwischen Dienstnehmer und Dienstleister vereinbart wird, nicht ändern, muss *keinerlei* Respezifikation von Policies durchgeführt werden, da diese nicht direkt auf den zu managenden Entitäten agieren, sondern diese über die Agenten der Integrationsschicht gesteuert und überwacht werden.

- *Änderung der Dienstimplementierung*
  Wird die Dienstimplementierung auf Dienstleisterseite aufgrund von Optimierungsmaßnahmen geändert, so muss einzig überprüft werden, ob die Messkomponenten zur Nutzungserfassung weiter verwendet werden können. Sofern dies nicht der Fall ist, müssen neue Messkomponenten implementiert werden. Dieser Fall wird allerdings bereits durch Implementierungspolicies abgedeckt, so dass hierfür keine Spezifikation von Meta–Policies notwendig ist. Die existierenden Konfigurations– und Betriebspolicies können ohne weitere Änderung bestehen bleiben.

[14] Man beachte hierbei die ausdrückliche Trennung der *Managementfunktionalität* von der *Güte* der Managementfunktionalität, wie es auch vom MNM Dienstmodell (vgl. Abschnitt 2.1.1) vorgesehen ist.

 Als Buch erhältlich im Verlag Dr. Hut, München, www.dr.hut-verlag.de (ISBN 3-934767-97-4)

Somit hat die durchgeführte Analyse gezeigt, dass nur Änderungen an der Dienstvereinbarung bzgl. des Tarifmodells und der Managementfunktionalität potentiell Änderungen an bestehenden Policies zur Folge haben können. Der Grund hierfür liegt darin, dass die Dienstvereinbarung zu Beginn als Basis herangezogen wird, um die initialen Policies zu spezifizieren. Treten nun Änderungen bzgl. dieser Basis auf, so müssen auch die entsprechenden Policies überprüft werden, ob diese ebenso angepasst werden müssen. Dieser Vorgang läßt sich allerdings komfortabel mittels Meta–Policies unterstützen und teilweise auch automatisieren, sofern es sich um die Deaktivierung und Löschung von existierenden Policies handelt, die keine Interaktion mit dem Policy–Ersteller erfordern.

5.4.4 Policies und Meta–Policies: Zusammenfassender Überblick

In Tabelle 5.8 ist ein vergleichender, schematischer Überblick sowohl über die Sprachbausteine als auch über die vom Policy–Ersteller zu spezifizierenden Bestandteile von Policies und Meta–Policies gegeben, die v.a. auf Basis der Analyse dieses Abschnitts 5.4 gewonnen werden konnten. Policies und Meta–Policies unterscheiden sich, aufgrund ihrer unterschiedlichen Zielsetzung, vor allem in Subject, Target und Event.

Policy	Meta–Policy
subject {*Dienst-ID, Kunden-ID oder (andere) Rolle*}	**subject** {*Policy–ID*}
target {*Integrationsagent für Akteur*}	**target** {*Policy–Verwaltungsagent*}
event {*Zustandsänderung von Austauschentitäten oder Initiierung einer Interaktion*}	**event** {*Zustandsänderung an Tarifmodell oder Managementfunktionalität*}
action{*(target|object.)Methodenaufruf*}	**action**{*(subject.)erzeugen/löschen/aktivieren/deaktivieren*}
constraint{*Boolescher Ausdruck*}	**constraint**{*Boolescher Ausdruck*}

Tabelle 5.8: Vergleich der Spezifikation von Policies und Meta–Policies

Policies sehen als Subject–Spezifikation meist den Dienst– und/oder Kunden–Identifikator vor. In seltenen Fällen wird auch nur eine allgemeine, u.U. provider–interne Rolle angegeben. Das Policy–Target ist immer ein sich innerhalb der Integrationsschicht befindlicher Integrationsagent für einen Akteur. Das aktions– bzw. evaluationsauslösende Ereignis ist entweder ein Interaktionsübergang zwischen zwei Akteuren oder eine Zustandsänderung von Ein–/Ausgabeentitäten. Die deklarierten Aktionen beziehen sich, sofern nichts anderes angegeben ist, immer auf die CORBA–Schnittstelle des Targets.

Das Subject von Meta–Policies bezieht sich hingegen immer auf eine Policy, so dass deren Identifikator spezifiziert wird. Das Meta–Policy–Target ist immer die jeweilige Policy–Verwaltungsinstanz, die den Policy–Lebenszyklus steuert und überwacht. Beim evaluationsauslösenden Ereignis handelt es sich immer um eine Änderung der vereinbarten Managementfunktionalität bzw. des vereinbarten Tarifmodells. Die Aktionen beschränken sich auf das Erzeugen, Löschen, Aktivieren und Deaktivieren von Policies.

Insgesamt ist es gelungen, für die Spezifikation von Policies für das prozessorientierte IT Management eine allgemeine Methodik für den Policy–Ersteller anzugeben (vgl. Abschnitt 5.4.1) sowie, anhand des Prozessmodells und der Methodik, allgemeine Aussagen für das Management jedes einzelnen identifizierten Abrechnungsteilprozesses abzuleiten (vgl. Abschnitt 5.4.2). Damit ist es insbesondere auch gelungen, für jeden Teilprozess einen Überblick der in Frage kommenden Subjects, Targets und Events zu nennen (vgl. auch Abbildung 5.6). Gleichermaßen konnten in Abschnitt 5.4.3 derartige Aussagen für die Deklaration von Meta–Policies gemacht werden.

In dem nun folgenden Abschnitt 5.5 dieses Kapitels werden nun auf Basis des ersten Grobentwurfs der policy–basierten Managementarchitektur in Abschnitt 5.3.2 die Schnittstellen des Abrechnungsmanagementsystems entworfen, damit ein policy–basiertes Management, wie in diesem Abschnitt beschrieben, ermöglicht wird.

 Als Buch erhältlich im Verlag Dr. Hut, München, www.dr.hut-verlag.de (ISBN 3-934767-97-4)

5.5 Entwurf eines policy–basierten, prozessorientierten Managementsystems

Dieser Abschnitt beschäftigt sich mit dem objektorientierten Entwurf eines policy–basierten, prozessorientierten Managementsystems, das den in Abschnitt 5.3.2 bereits vorgestellten Grobentwurf der Managementarchitektur umsetzt. Hierzu wird zunächst in Abschnitt 5.5.1 eine Anforderungsanalyse bezüglich der durch Schnittstellen und Klassen zu erfüllenden Funktionalität durchgeführt. Darauf aufbauend wird in Abschnitt 5.5.2 ein Überblick über die Package–Struktur des Managementsystems geliefert, um anschließend in den Abschnitten 5.5.3 bis 5.5.6 die einzelnen Packages detaillierter zu beschreiben.

Es gilt festzustellen, dass sich die entworfenen Klassen und Schnittstellen für die Implementierung einer allgemein prozessorientierten, auf policy–basierten Konzepten beruhenden Managementanwendung eignen und somit *nicht* auf den Funktionsbereich des Abrechnungsmanagements beschränkt sind.

5.5.1 Anforderungen an die Funktionalität

In diesem Abschnitt werden Anforderungen an die Funktionalität des policy–basierten, prozessorientierten Managementsystems formuliert, indem die Interaktionen zwischen den beteiligten Objekten an den entsprechenden Schnittstellen analysiert werden. Im Folgenden werden allerdings nur die technischen Schnittstellen der Beteiligten in Betracht gezogen und somit Anforderungen an etwaige graphische Benutzerschnittstellen außer Acht gelassen. In Abbildung 5.8 sind die im System interagierenden Objekte sowie die aus Managementsicht relevanten Interaktionsschnittstellen dargestellt. Grob lassen sich die Schnittstellen einteilen in die Nutzungsschnittstelle der policy–basierten Managementanwendung, mit welcher der Policy–Ersteller und Administrator interagiert, in die Managementschnittstelle der Integrationsagenten, mit denen die Managementanwendung interagiert, und schließlich in die Nut-

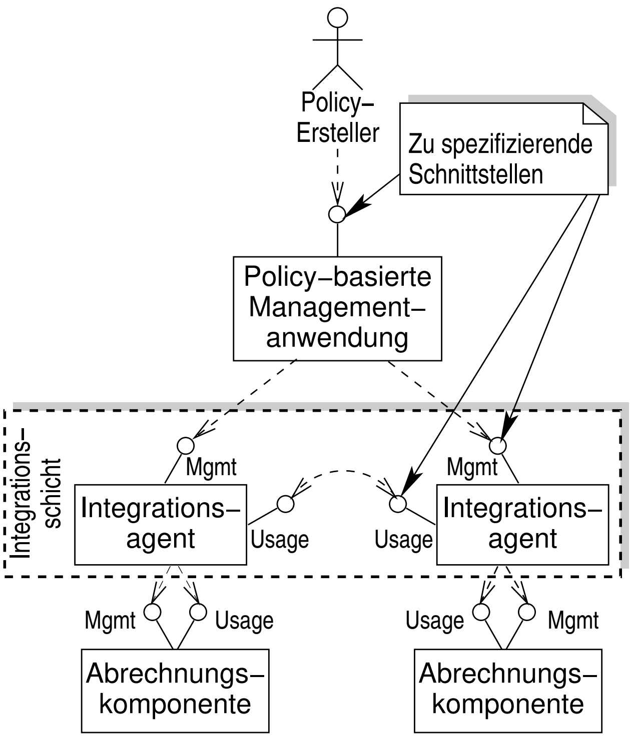

Abbildung 5.8: Überblick über die zu spezifizierenden Schnittstellen des Managementsystems

zungsschnittstellen der Integrationsagenten, mit Hilfe derer die Agenten innerhalb der Integrationsschicht miteinander interagieren und damit kooperieren. Nachfolgend werden die einzeln genannten Schnittstellen auf ihren obligatorischen Funktionalitätsumfang hin untersucht:

- *Zwischen Policy–Ersteller und Managementanwendung*
  Aus Sicht des Policy–Erstellers muss die Schnittstelle zur Managementanwendung mindestens Methoden zum Management des Policy–Lebenszyklus aufweisen, demnach also zum *Erstellen*, *Löschen*, *Aktivieren* und *Deaktivieren* einer oder mehrerer Policies. Um das Erstellen von Policies zu erleichtern, ist eine *Unterstützung* der in Abschnitt 5.4.1 vorgestellten *Methodik* durch die Managementanwendung wünschenswert. Die *Korrektheit* der erstellten Policies sollte ebenfalls getestet werden können, z.B. durch einfache Sanity–Checks und Syntaxprüfungen. Desweiteren, um eine De-/Aktivierung von bestehenden Policies zu ermöglichen, ist ebenso eine *effiziente Suche* nach Policies mit einer Variation von unterschiedlichen Suchargumenten gefordert.

- *Zwischen Managementanwendung und Integrationsschicht*
  Die policy–basierte Managementanwendung interagiert mit der Integrationsschicht über die jeweiligen *Managementschnittstellen* der Integrationsagenten. Hierbei muss diese Managementschnittstelle aufgeteilt werden in Methoden, die den *Lebenszyklus des Integrationsagenten steuern* und *überwachen*, und in Methoden, welche den *Lebenszyklus* der, einem Integrationsagenten zugewiesenen „tatsächlichen“, *vorgangsrealisierenden Komponenten steuern* und *überwachen*. Die Methoden zum Management des Agentenlebenszyklus lassen sich ohne weiteres für alle Integrationsagenten vereinheitlicht darstellen: diese Schnittstelle umfasst das Erzeugen, Löschen, Starten und Anhalten des Agenten. Die Methoden zum Management der vorgangsrealisierenden Komponenten müssen allerdings z.T. in Abhängigkeit von der jeweiligen, durch den Akteur zu erfüllenden Funktionalität definiert werden.

- *Zwischen Integrationsagenten der Integrationsschicht*
  Die Kooperation der vorgangsrealisierenden Komponenten wird über die Kopplung der Nutzungsschnittstellen der jeweiligen Integrationsagenten erreicht. Um diese Kopplung zu vereinfachen, ist auch in diesem Fall eine Vereinheitlichung der Schnittstellen wünschenswert. Allerdings lassen sich auch in diesem Fall keine allgemeinen Anforderungen ableiten, da diese ebenfalls (nutzungs–)funktionalitätsabhängig sind. Hierbei kann man auf bereits standardisierte Schnittstellen zurückgreifen, sofern diese vorhanden sind bzw. wieder die Prozessanalyse aus Kapitel 4 heranziehen.

5.5.2 Überblick: Die Package–Struktur

In Abbildung 5.9 ist ein schematischer Überblick über die entworfene Package–Struktur in der UML Klassendiagramm Notation dargestellt. Die Abbildung visualisiert die wichtigsten Klassen, Schnittstellen und Assoziationen, die in den nachfolgenden Abschnitten verfeinert und

 Als Buch erhältlich im Verlag Dr. Hut, München, www.dr.hut-verlag.de (ISBN 3-934767-97-4)

detaillierter erläutert werden. Beim Entwurf wurden bekannte Entwurfsmuster (engl.: *Design Patterns*) [GHJV 95] aus dem Software–Engineering–Bereich angewendet. Inbesondere kamen das Creational Pattern *Abstract Factory*, das Structural Pattern *Proxy* sowie die Behavioral Patterns *Mediator* und *State* zum Einsatz.

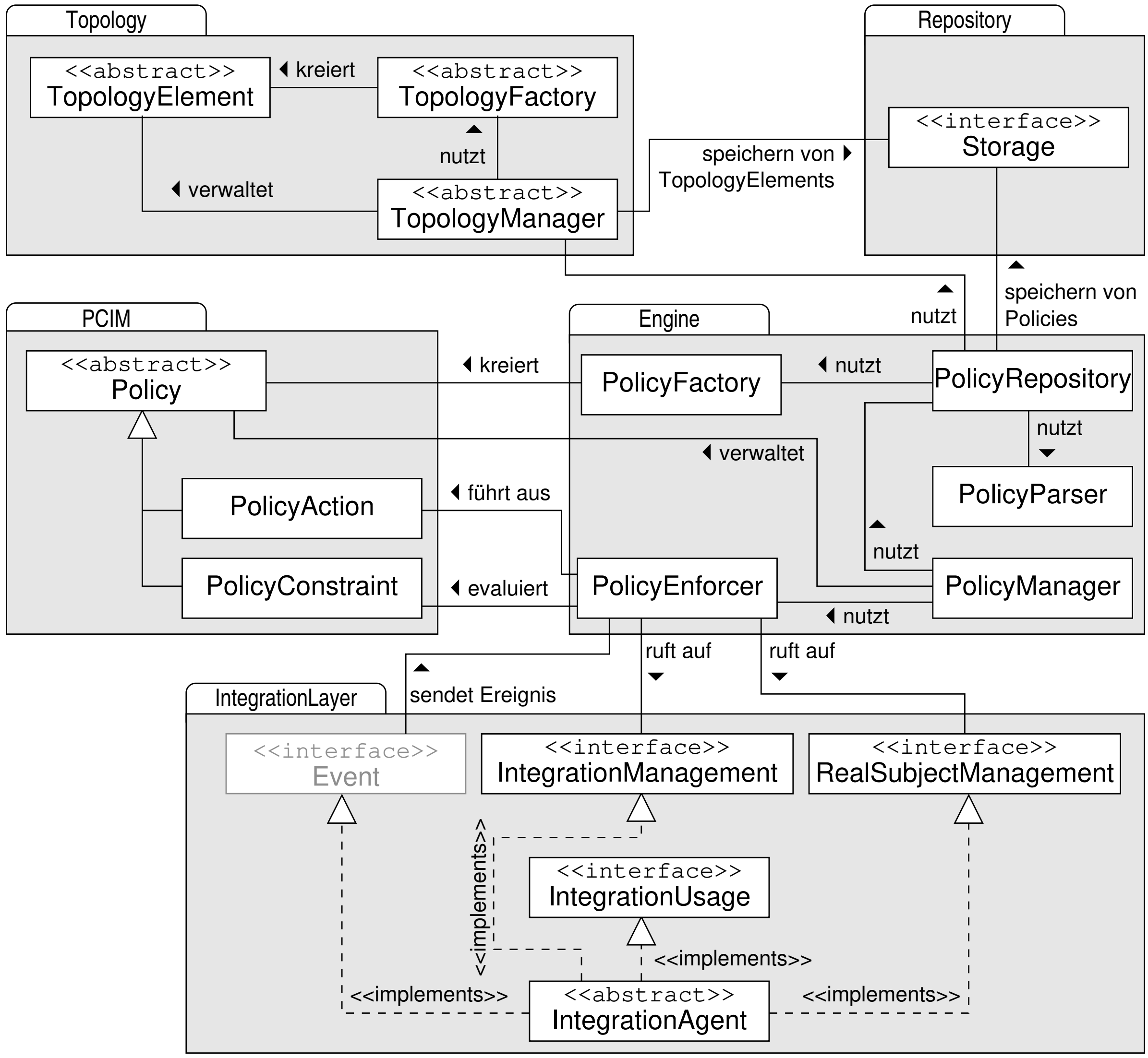

Abbildung 5.9: Schematischer Teilausschnitt der Package–Hierarchie

Insgesamt ist das System in fünf Packages aufgeteilt:

- `Topology`–*Package:* Dieses Package ermöglicht grundsätzlich das Erstellen und Verwalten von beliebigen Topologieinformationen. Mit Hinblick auf die von der PDL zu unterstützenden Topologiestrukturen (siehe hierzu die Sprachanalyse aus Abschnitt 5.3.3) sind insbesondere die Verwaltung von Rollen, Domänen und Gruppen vorgesehen.

- `Repository`–*Package:* Hiermit wird es ermöglicht, Objekte persistent zu speichern sowie wiederherzustellen. Es wird v.a. für das Speichern von Policies und Topologiedaten verwendet. Das Package ist als konzeptionelle Kapselung für existierende Schnittstellen, wie z.B. JNDI für Verzeichnisdienste, zu sehen und wird deswegen im Weiteren nicht detaillierter dargestellt.
- `PCIM`–*Package:* Das Package enthält die Umsetzung und Implementierung des bereits in Abschnitt 5.3.1 kurz erläuterten CIM Policy Models [DSP 0108] der DMTF resp. der IETF. Wie bereits erwähnt, handelt es sich beim CIM Policy Model um eine Standardisierung der grundsätzlichen Struktur und des Aufbaus von Managementpolicies. Damit werden die wichtigsten Policy–Bestandteile und deren Beziehungen zueinander festgelegt. Allerdings enthält das CIM Policy Model keinerlei Spezifikationen bzgl. der Interpretierung und Durchsetzung von Policies.
- `Engine`–*Package:* Das Herzstück des Systems bildet das Engine–Package, das Klassen zur Verwaltung und Durchsetzung von Policies enthält. In diesem Package sind auch Schnittstellen für die Interaktion mit dem Policy–Ersteller und Administrator definiert.
- `IntegrationLayer`–*Package:* Dieses Package sieht Schnittstellen vor, welche von den Integrationsagenten der Integrationsschicht zu implementieren sind, damit diese in vernünftiger Art und Weise in das policy–basierte, prozessorientierte Management miteinbezogen werden können. Insbesondere sind Schnittstellen für das Agentenmanagement und für das Management der zugewiesenen, vorgangsrealisierenden Komponenten vorgesehen.

In den nachfolgenden Abschnitten werden die wichtigsten Klassen und Schnittstellen der einzelnen Packages erläutert. Hierbei wird für jedes Package ein eigenes, verfeinertes Klassendiagramm entworfen. Es muss betont werden, dass nachfolgend lediglich die wichtigsten Klassen und Schnittstellen vorgestellt werden, die für die Umsetzung der in den vorangegangenen Abschnitten dieses Kapitels entwickelten Konzepte notwendig sind. Hierbei fokussiert die Beschreibung der noch kommenden, verfeinerten Diagramme auf die Semantik der einzelnen Elemente, wie z.B. der Methoden, und weniger auf Implementierungsdetails, wie z.B. die gewählten Datentypen, auch wenn diese teilweise bereits in den Diagrammen notiert sind. Eine prototypische Implementierung des in diesem Abschnitt vorgestellten Entwurfs wird im darauffolgenden Kapitel 6 präsentiert.

5.5.3 Das `Topology`–Package

Abbildung 5.10 gibt einen Überblick über die wichtigsten Klassen des Topology–Packages, die es ermöglichen, beliebige Topologiedaten anzulegen und zu verwalten. Eine Topologie wird hierbei ganz allgemein als eine Verwaltungsstruktur verstanden, mit der Assoziationen zwischen Objekten ausgedrückt werden können. Im vorliegenden Fall sind Topologien in Form von Rollen, Domänen und Gruppen von Interesse.

 Als Buch erhältlich im Verlag Dr. Hut, München, www.dr.hut-verlag.de (ISBN 3-934767-97-4)

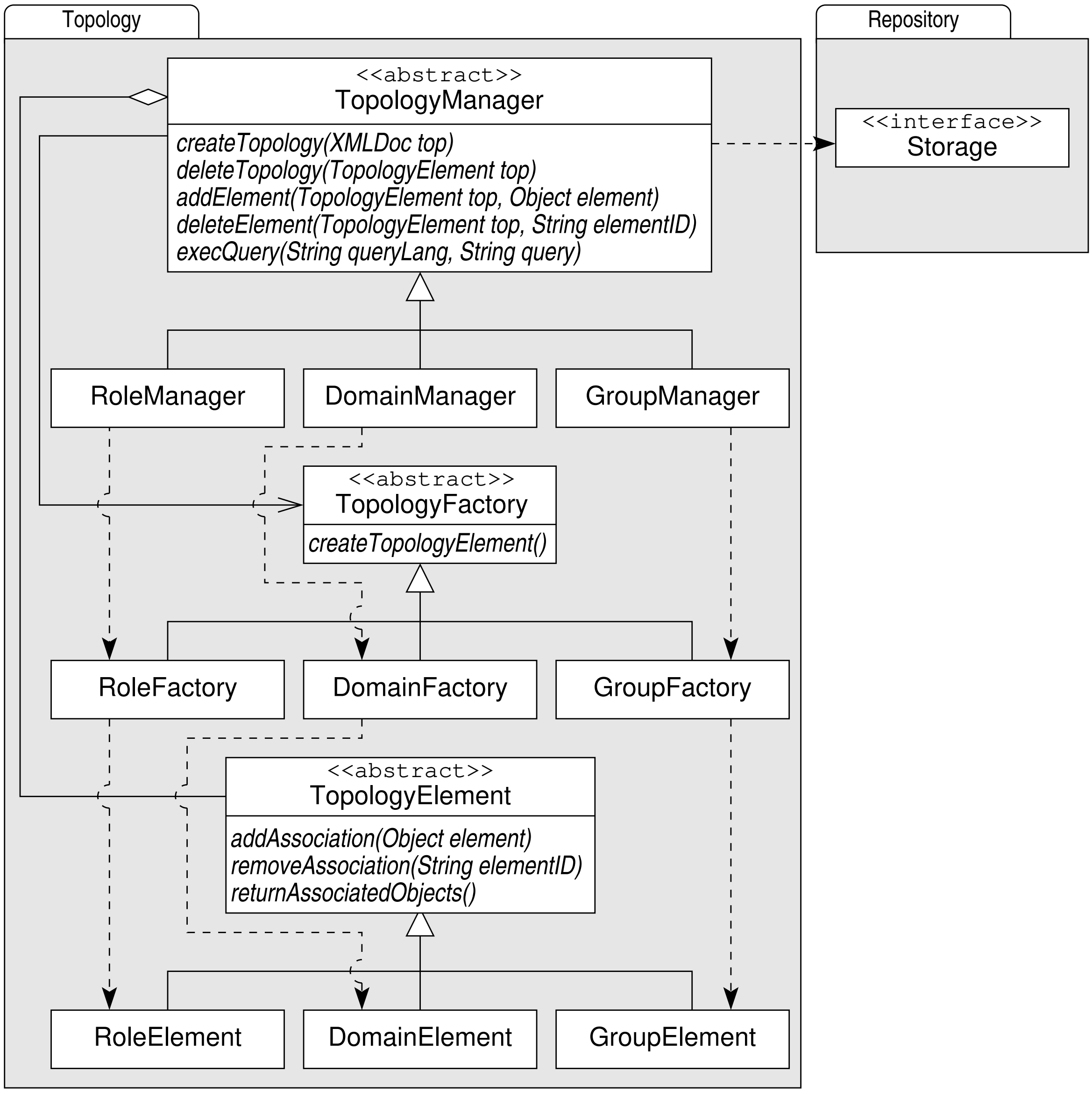

Abbildung 5.10: Das `Topology`–Package

Beim Entwurf dieses Packages wurde das nach [GHJV 95] als *Abstract Factory* bezeichnete Design Pattern angewendet, das sich in den folgenden zwei Designaspekten niederschlägt:

- Die Erstellung von konkreten Objekten wird durch eine *Factory* von deren tatsächlicher Verwaltung getrennt. Ein sich daraus ergebender Vorteil ist, dass die Verwaltung der Objekte von der (tatsächlichen) Objektimplementierung getrennt wird und die Objekte ausschließlich über die zur Verfügung gestellte Schnittstelle manipuliert werden können. Da eine Factory für konkrete Objekte üblicherweise genau einmal instantiiert wird, können Versionsänderungen durch Austausch der Factory leicht durchgesetzt werden. Im Package wird dieses Entwurfsmuster durch Einführen der drei Klassen *TopologyManager*, *TopologyFactory* und *TopologyElement* umgesetzt.

- Das *Abstract Factory* Design Pattern sieht desweiteren vor, dass für ähnliche, zu verwaltende Objekte dieselbe Schnittstelle zum Erzeugen dieser Objekte verwendet wird. Dies wird dadurch erreicht, dass die Factory und die von der Factory zu instantiierende Klasse als abstrakt deklariert werden und die jeweiligen konkreten Klassen in einer Vererbungsbeziehung zu diesen abstrakten Klassen stehen. Damit ist gesichert, dass die Schnittstelle für das Erzeugen von ähnlichen Objekten gleich ist. Im Package wird dieser Aspekt durch die Einführung von konkreten Manager–, Factory– und Element–Klassen für Rollen, Domänen und Gruppen umgesetzt.

Nachfolgend werden die Schnittstellen der abstrakten Klassen *TopologyElement*, *TopologyFactory* und *TopologyManager* erklärt.

Die abstrakte Klasse `TopologyElement`

Diese Klasse ist die kleinste Verwaltungseinheit zum Ausdruck von Topologieinformationen. Wie bereits zu Beginn des Abschnitts 5.5.3 erklärt wurde, wird eine Topologie durch Assoziationen zwischen Objekten ausgedrückt. Damit sieht die Schnittstelle folgende Methoden vor:

- `addAssociation(`*`Object element`*`)` — Eine Assoziation zum übergebenen Objekt wird erstellt. Bei dem übergebenen Objekt kann es sich wieder um ein TopologyElement handeln, so dass damit Hierarchien und allgemeine Graphstrukturen aufgebaut werden können. Beim Erzeugen der Assoziation wird ein eindeutiger Schlüssel generiert.

- `removeAssociation(`*`String elementID`*`)` — Eine Assoziation mit gegebenem Schlüssel wird gelöscht.

- `returnAssociatedObjects()` — Alle assoziierten Objekte werden zurückgegeben. Durch rekursiven Aufruf dieser Methode auf den zurückgelieferten TopologyElement–Objekten ist eine Objektsuche realisierbar.

Die abstrakte Klasse `TopologyFactory`

Diese Klasse stellt lediglich die folgende Methode zur Verfügung:

- `createTopologyElement()` — Es wird ein Objekt der Klasse TopologyElement erzeugt und zurückgeliefert.

 Als Buch erhältlich im Verlag Dr. Hut, München, www.dr.hut-verlag.de (ISBN 3-934767-97-4)

Die abstrakte Klasse `TopologyManager`

Die Schnittstelle dieser Klasse weist Methoden zur Verwaltung von *TopologyElements* auf. Zur persistenten Speicherung von Topologiedaten wird das Storage–Interface des Repository–Packages verwendet. Folgende Methoden sind spezifiziert:

- `createTopology(`*`XMLDoc top`*`)` — Auf Basis eines wohlstrukturierten Dokuments (in diesem Fall als XML–Dokument), das Assoziationen zwischen Topologie–Elementen deklariert, wird die entsprechende Topologie erzeugt. Eine Assoziation zwischen zwei TopologyElement–Objekten wird jeweils in beiden Objekten als solche registriert. Das Wurzelelement der Topologie wird zurückgegeben[15]. Je nach Art und Weise der Realisierung dieser Klasse, kann beim Erzeugen der Topologie beispielsweise eine Indexstruktur erzeugt werden, welche die Suche nach Objekten optimiert.
- `deleteTopology(`*`TopologyElement top`*`)` — Die zu einem TopologyElement–Objekt gehörende Topologie wird gelöscht. Hierbei werden rekursiv alle Assoziationen und TopologyElement–Objekte gelöscht.
- `addElement(`*`TopologyElement top, Object element`*`)` — Das übergebene Objekt *element* wird an der angegebenen Stelle *top* in die Topologie eingefügt. Handelt es sich bei *element* um ein Objekt der Klasse TopologyElement, so wird *element* auch eine Assoziation zum *top* hinzugefügt.
- `deleteElement(`*`TopologyElement top, String elementID`*`)` — Ein Element an der angegebenen Stelle wird aus der Topologie entfernt. Handelt es sich bei diesem Element um ein Objekt der Klasse TopologyElement, so wird diesem Objekt ebenfalls die entsprechende Assoziation zum *top* entfernt.
- `execQuery(`*`String queryLang, String query`*`)` — Es wird die Suche nach ein oder mehreren Objekten ausgeführt. Da die Topologie–Informationen grundsätzlich auf unterschiedliche Art und Weise verwaltet werden kann, sind mehrere Anfragensprachen durchaus denkbar. Um sich grundsätzlich für Erweiterungen und Optimierungen offen zu halten, wird neben der Suchanfrage selber auch die Anfragesprache beim Aufruf mitangegeben.

5.5.4 Das `PCIM`–Package

In Abbildung 5.11 ist das PCIM–Package in UML Klassendiagramm Notation dargestellt. Es handelt sich hierbei im Wesentlichen um die Umsetzung des CIM Policy Model [DSP 0108] der DMTF resp. der IETF (siehe hierzu auch Abschnitt 5.3.1). Wie aus der Abbildung

[15] Es existiert immer ein Wurzelelement, auch wenn dieses nicht explizit als solches in der (ursprünglichen) Topologie vorgesehen ist. In diesem Fall wird eine als künstlich gekennzeichnete Wurzel erzeugt, welche auf die weiteren TopologyElement–Objekte verweist.

ersehen werden kann, entspricht das Klassendiagramm einer objektorientierten Darstellung der PDL–Sprachbausteine aus Abschnitt 5.3.4. In spezifikationskonformer Erweiterung zum PCIM–Standard, wurden die Bestandteile des *PolicyConstraint* spezifiziert. Da die Klassen–Strukturierung direkt aus der bereits in Abschnitt 5.3.3 beschriebenen PDL ableitbar ist, wird auf eine wiederholten Erläuterung der Semantik der einzelnen Klassen und damit Sprachbausteine an dieser Stelle verzichtet und auf den genannten Abschnitt verwiesen.

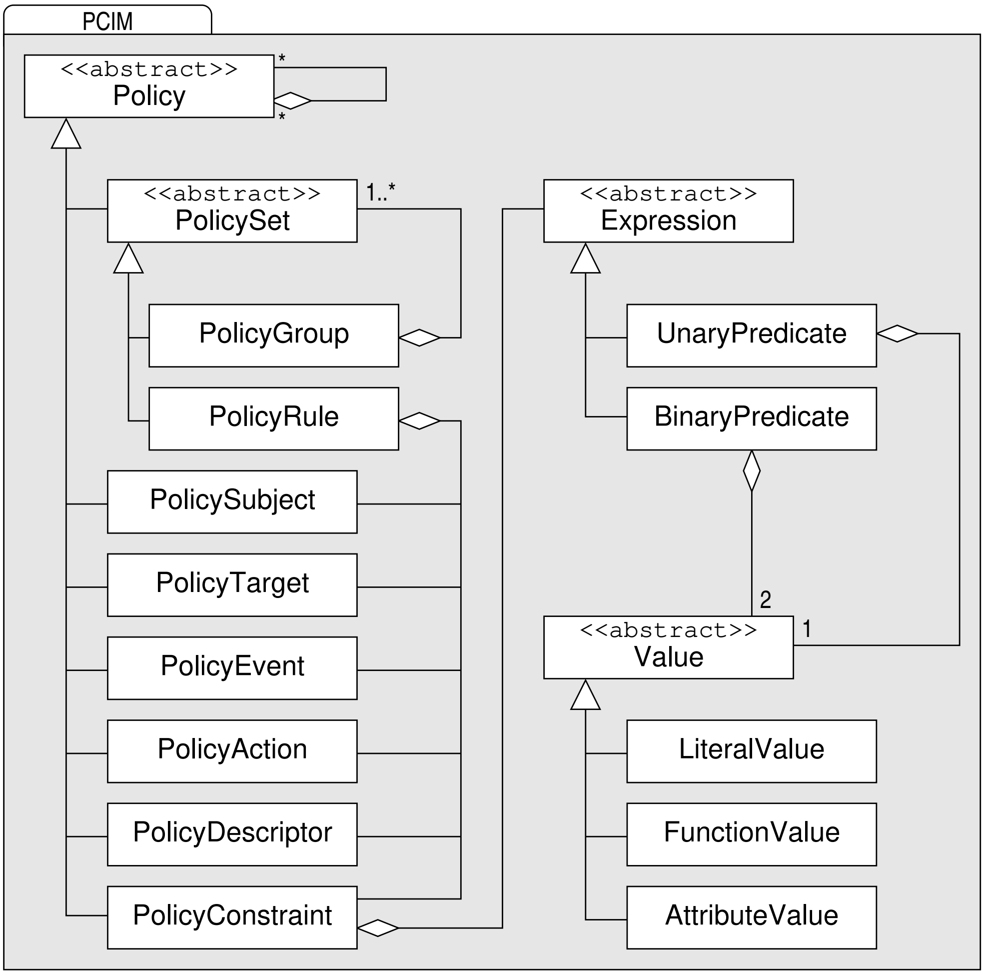

Abbildung 5.11: Das `PCIM`–Package

5.5.5 Das `Engine`–Package

Das *Engine*–Package weist als Herzstück der policy–basierten Abrechnungsmanagementanwendung Klassen zur Verwaltung, Manipulation und Durchsetzung von Policies auf. Ähnlich zum *Topology*–Package (siehe Abschnitt 5.5.3) wurde das *Abstract Factory* Design Pattern beim Entwurf des Packages angewendet, so dass die relevante Package–Funktionalität zur Policy–

Verwaltung in die beiden Klassen *PolicyFactory* und *PolicyManager* aufgeteilt wurde. Insbesondere sieht das Package mit der *PolicyManager*–Klasse eine Schnittstelle für die Interaktion mit einem Policy–Ersteller und Administrator vor. In Abbildung 5.12 sind die relevanten Klassen des Packages dargestellt, die in den nachfolgenden Abschnitten erläutert werden.

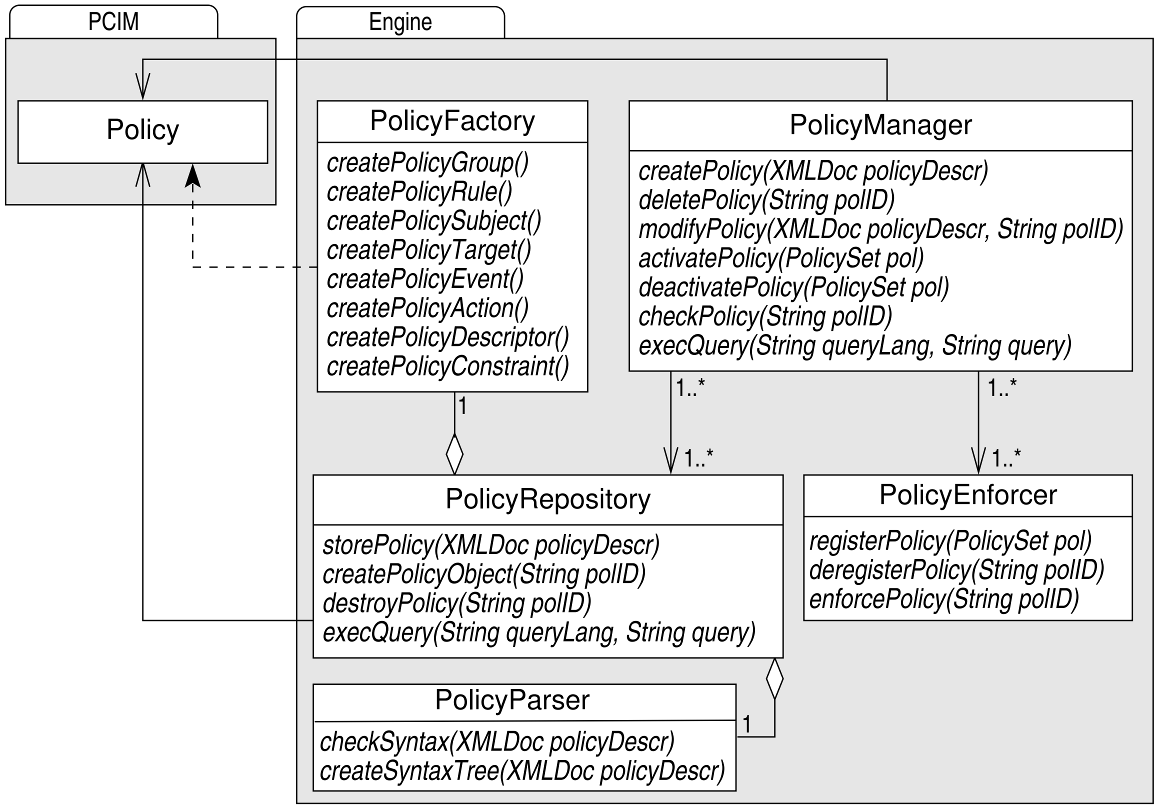

Abbildung 5.12: Das `Engine`–Package

Die Klasse `PolicyFactory`

Die Klasse PolicyFactory beinhaltet Methoden zum Erzeugen von Objektklassen des PCIM–Packages. Da alle Methoden der Schnittstelle die gleiche Semantik haben, nämlich das Instantiieren der zur Methode gehörigen Klasse des PCIM–Packages, werden sie an dieser Stelle nicht einzeln genannt und erläutert.

Die Klasse `PolicyParser`

Die Schnittstelle dieser Klasse weist Methoden zum Parsen von in PDL vorliegenden Dokumenten auf. Je nachdem, in welcher Form die Grammatik spezifiziert ist (als XML–Schema, DTD, BNF, etc.) können unterschiedliche Parser verwendet werden. Bei der prototypischen

Implementierung, die in Kapitel 6 vorgestellt wird, wird ein XML–Parser verwendet. Bei dem `XMLDoc` bezeichneten Datentyp handelt es sich um einen `String`, so dass der Parser ohne Änderung der Schnittstelle ausgetauscht werden kann. Im Einzelnen werden folgende Methoden angeboten:

- `checkSyntax(`*`XMLDoc policyDescr`*`)` — Ein in PDL vorliegendes Dokument wird explizit auf korrekte Syntax überprüft.
- `createSyntaxTree(`*`XMLDoc policyDescr`*`)` — Es wird ein in PDL vorliegendes Dokument wird geparst und ein Syntaxbaum aufgebaut.

Die Klasse `PolicyRepository`

Die Schnittstelle dieser Klasse weist Methoden zum persistenten Speichern, Kreieren und Löschen von Policies auf. Die Klassen *PolicyParser* und *PolicyFactory* sind jeweils Teil der *PolicyRepository*–Klasse. Im einzelnen werden folgende Methoden angeboten:

- `storePolicy(`*`XMLDoc policyDescr`*`)` — Ein in PDL vorliegendes Dokument wird persistent gespeichert. Hierbei wird ein eindeutiger Identifikator erzeugt.
- `createPolicyObject(`*`String polID`*`)` — Für ein bereits abgespeichertes, in PDL vorliegendes Dokument wird durch Verwendung der *PolicyParser*–Schnittstelle ein Syntaxbaum erzeugt und darauf aufbauend werden die in diesem Dokument deklarierten Policy–Bestandteile wie Subjekte, Aktionen, etc. durch Verwendung der *PolicyFactory*–Schnittstelle instantiiert.
- `destroyPolicy(`*`String polID`*`)` — Eine persistent gespeicherte Policy wird gelöscht.
- `execQuery(`*`String queryLang, String query`*`)` — Ähnlich zum *Topology-Manager* aus Abschnitt 5.5.3 wird diese Methode verwendet, um mit einem gegebenen Query–String nach persistent gespeicherten Policies zu suchen. Um sich grundsätzlich für Erweiterungen und Optimierungen offen zu halten, wird auch die Anfragesprache beim Aufruf mitangegeben.

Die Klasse `PolicyManager`

Die Schnittstelle dieser Klasse weist Methoden für die direkte Interaktion mit dem Policy–Ersteller und Administrator zur Verwaltung von *Policies* auf. Hierbei wird von der *PolicyManager*–Klasse die Funktionalität sowohl der *PolicyRepository*– als auch *PolicyEnforcer*–Klasse in Anspruch genommen. Folgende Methoden sind spezifiziert:

- `createPolicy(`*`XMLDoc policyDescr`*`)` — Es wird ein Dokument in der durch die PDL vorgeschriebenen Syntax aus Abschnitt 5.3.4 übergeben (im vorliegenden Fall der prototypischen Implementierung als XML–Dokument). Dieses Dokument kann die Deklaration von einer oder mehreren Policies als auch von einzelnen Policy–Bestandteilen, wie z.B. Aktionen, Subjects und Targets enthalten. Die notwendigen Objekte werden unter zur Hilfenahme der *PolicyRepository* mit entsprechend eindeutigen Identifikatoren erzeugt. Zudem werden die erzeugten Objekte in geeignete Verwaltungsstrukturen eingefügt.

- `deletePolicy(`*`String polID`*`)` — Ein bereits existierendes Objekt innerhalb der Policy–Verwaltungsinstanz wird gelöscht. Hierbei kann es sich um eine Policy oder um einzelne, per ID referenzierbare Policy–Bestandteile handeln. Bei Durchführung wird mittels der eigenen Verwaltungsstrukturen auf Querabhängigkeiten geachtet, insbesondere wenn beispielsweise ein Policy–Bestandteil, wie eine Menge von Aktionen, von mehreren Policies referenziert wird.

- `modifyPolicy(`*`XMLDoc policyDescr, String polID`*`)` — Diese Methode entspricht semantisch der `createPolicy()`, nur dass hierbei keine ID generiert wird, sondern diese explizit übergeben wird. Damit wird ein bereits existierendes Objekt bzw. bereits existierende Objekte durch einfaches Überschreiben aktualisiert.

- `activatePolicy(`*`PolicySet pol`*`)`— Ein existierender *PolicySet* wird aktiviert. Hierbei wird die Policy beim zugewiesenen *PolicyEnforcer* registriert, welcher sich um die tatsächliche Durchsetzung der deklarierten Aktionen kümmert. Da ein PolicySet immer nur eine Menge von *PolicyRules* sein kann, sind alle Daten, wie Subject, Target, Action, etc. die zum erfolgreichen Durchsetzen der Policy benötigt werden, vorhanden.

- `deactivatePolicy(`*`PolicySet pol`*`)` — Ein existierender und vorher aktivierter *PolicySet* wird deaktiviert. Hierzu wird die Policy beim zugewiesenen *PolicyEnforcer* wieder deregistriert.

- `checkPolicy(`*`String polID`*`)` — Es wird ein existierendes Policy–Objekt auf Sinnhaftigkeit überprüft. Dies beinhaltet beispielsweise Überprüfungen, ob spezifizierte Aktionen auf der angegeben Target–Menge tatsächlich ausführbar sind, ob die Subjekt–Auflösung erfolgreich ist, etc.

- `execQuery(`*`String queryLang, String query`*`)` — Ähnlich zum *Topology-Manager* aus Abschnitt 5.5.3 wird diese Methode verwendet, um mit einem gegebenen Query–String nach Policies bzw. einzelnen, referenzierbaren Policy–Bestandteilen zu suchen. Um sich grundsätzlich für Erweiterungen und Optimierungen offen zu halten, wird auch die Anfragesprache beim Aufruf mitangegeben.

Die Klasse `PolicyEnforcer`

Die Funktionalität der Klasse *PolicyEnforcer* besteht darin, aktivierte Policies entsprechend ihrer Deklaration durchzusetzen. Die Klasse *PolicyEnforcer* weist explizit keine nach außen hin sichtbare Methoden zur Verwaltung von registrierten Policies auf. Methoden zur Verwaltung von Policies sieht alleinig der dafür zuständige *PolicyManager* vor. Nachfolgend werden die Methoden der Schnittstelle erläutert:

- `registerPolicy(`*`PolicySet pol`*`)` — Eine (aktivierte) Policy wird zur Durchsetzung registriert. Hierbei werden die evaluationsauslösenden Ereignisse in geeignete Verwaltungsdatenstrukturen eingefügt. Bei Auftreten eines Ereignisses werden die diesem Ereignis zugewiesenen Policies resp. deren Constraints evaluiert und im positiven Fall die entsprechenden Aktionen ausgeführt.
- `deregisterPolicy(`*`String polID`*`)` — Eine bereits registrierte Policy wird wieder deregistriert. Hierzu werden alle zu dieser Policy gehörenden Daten aus den Verwaltungsstrukturen entfernt.
- `enforcePolicy(`*`String polID`*`)` — Eine registrierte Policy wird explizit durchgesetzt. Das heißt, dass weder vorher ein evaluationsauslösendes Ereignis eintreten muss, noch die dazugehörigen Constraints evaluiert werden.

5.5.6 Das `IntegrationLayer`–Package

Das *IntegrationLayer*–Package sieht Schnittstellen für die Integrationsagenten der Integrationsschicht vor. In Abbildung 5.13 sind die relevanten Schnittstellen des Packages in der UML Klassendiagramm Notation dargestellt. Beim Entwurf des Packages wurden die Design Pattern *Proxy* und *Mediator* angewendet, die sich v.a. durch die Existenz der Schnittstelle *RealSubjectManagement* und den Referenzen *Subject* und *Colleague* der abstrakten Klasse *IntegrationAgent* äußern. Alle Agenten der Integrationsschicht stehen in einer Vererbungsbeziehung zur abstrakten Klasse *IntegrationAgent*. Neben den noch in den nachfolgenden Abschnitten erläuterten Elementen des Packages implementiert ein *IntegrationAgent* ein *Event*–Interface, das nicht näher beschrieben wird, da es sich hierbei um die Abstraktion eines von der gegebenen Entwicklungs– und Ablaufumgebung zur Verfügung gestellten Ereignismechanismus handelt (im vorliegenden Fall: CORBA Event Service).

Die Schnittstelle `IntegrationUsage`

Die Schnittstelle *IntegrationUsage* ist die vereinheitlichte Nutzungsschnittstelle eines Integrationsagenten, mit der die Kopplung von verschiedenartigen Agenten der Integrationsschicht wesentlich vereinfacht wird. Die generische Schnittstelle sieht hierbei die folgende Methode vor:

- `execQuery(`*`String queryLang, String query`*`)` — Mit dieser Methode kann eine Anfrage an den Integrationsagenten abgesetzt werden. In der Regel erfüllt ein Integrationsagent neben der reinen Proxy–Funktionalität bzgl. der ihm zugewiesenen Subjekte auch die Funktionalität eines Mediators/Gateways zwischen verschiedenen Integrationsagenten (den sog. *Colleagues*) und damit zwischen den diesen Agenten zugewiesenen

 Als Buch erhältlich im Verlag Dr. Hut, München, www.dr.hut-verlag.de (ISBN 3-934767-97-4)

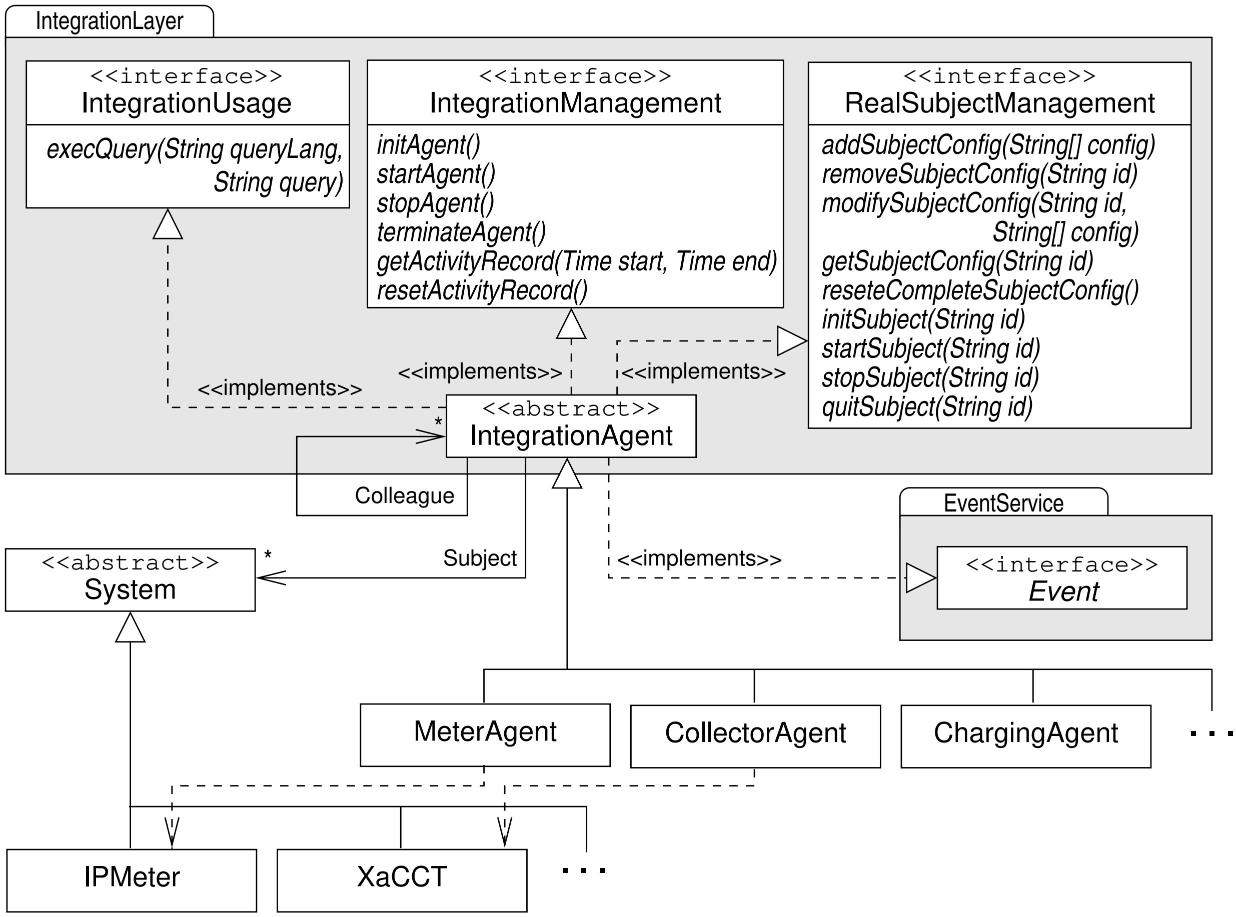

Abbildung 5.13: Das `IntegrationLayer`–Package

Subjekten (siehe hierzu auch die Ausführungen in Abschnitt 5.3.2 auf Seite 142). Mit der Funktionalität des Mediators ist auch eine gewisse Verarbeitung von Daten verbunden. Diese Methode stößt nun eine derartige Verarbeitung an bzw. fragt die Ergebnisse einer bereits durchgeführten Verarbeitung ab. Diese Methode wird v.a. zur Kopplung von unterschiedlichen Agenten der Integrationsschicht verwendet, d.h. dass diese Methode v.a. von anderen Agenten aufgerufen wird.

Die Schnittstelle `IntegrationManagement`

Die *IntegrationManagement*–Schnittstelle sieht Methoden zum Management des Agenten–Lebenszyklus vor, die im nachfolgenden detailliert erklärt werden:

- `initAgent()` — Der Integrationsagent wird mit (übergebenen) Startwerten initialisiert. Die obligatorischen Initialisierungswerte beziehen sich hierbei lediglich auf den Lebenszyklus des Integrationsagenten und *nicht* auf die der ihm zugewiesenen Abrechnungskomponenten. Üblicherweise werden mit der Initialisierung auch die Referenzen auf weitere Integrationsagenten der Integrationsschicht übergeben (im *Mediator*–Jargon auch *Colleague* genannt), mit denen der Agent zusammenarbeiten soll. Da die tatsächlichen Initialisierungswerte von der Funktionalität des jeweiligen Agenten abhängig sind, sind in diesem Fall keine Standardparameter angegeben.

- `startAgent()` — Die Ausführung des Integrationsagenten wird gestartet.
- `stopAgent()` — Die Ausführung des Integrationsagenten wird beendet.
- `terminateAgent()` — Der Integrationsagent wird deinitialisiert und vollständig entfernt.
- `getActivityRecord(`*`Time start, Time end`*`)` — Jeder Integrationsagent zeichnet besondere Ereignisse während der Ausführung auf. Um die durchgeführten Aktivitäten für ein späteres Auditing nachvollziehen und damit nachprüfen zu können, sollten im Fall des *IntegrationAgents* mindestens die aufgerufenen Methoden des *RealSubject-Management*–Interfaces jeweils mit Zeitstempeln aufgezeichnet werden. Der Aufruf der Methode liefert damit das Aktivitätsprotokoll des übergebenen Zeitfensters [*start, end*] zurück.
- `resetActivityRecord()` — Das bisher geführte Aktivitätsprotokoll wird gelöscht und ein neues angelegt. Das neu angelegte Aktivitätsprotokoll weist hierbei als ersten Eintrag das Zurücksetzen der Protokollführung auf.

Die Schnittstelle `RealSubjectManagement`

Die *RealSubjectManagment*–Schnittstelle ist als Teil der Umsetzung des *Proxy*–Entwurfsmusters zu verstehen. Die abstrakte Klasse *System* repräsentiert konzeptionell eine oder mehrere tatsächliche Komponenten eines vorgangsrealisierenden Systems (wie z.B. eines Abrechnungssystems), von denen mittels der *IntegrationAgent*–Klasse abstrahiert werden soll. Diese zu abstrahierenden Komponenten werden im Proxy–Entwurfsmuster als *Subject* bezeichnet und sind nicht mit dem Policy–Subject zu verwechseln. Das Interface sieht nun generische Methoden zum Management des Lebenszyklus eines dem Integrationsagenten zugewiesenen Subjects vor. Wie im nächsten Abschnitt noch detaillierter erklärt wird, kann es z.T. sinnvoll sein, dass der tatsächliche Integrationsagent, wie z.B. ein realisierter *MeterAgent*, etc., eine weitere Verfeinerung dieser Schnittstelle implementiert und somit zusätzliche Methoden zur Beeinflussung des Subject–Lebenszyklus besitzt. Desweiteren kann ein Integrationsagent grundsätzlich bzgl. mehrerer Subjekte gleichzeitig als Stellvertreter agieren, so dass geeignete Verwaltungsstrukturen vorhanden sein müssen. Konkret sind folgende Methoden vorgesehen:

- `addSubjectConfig(`*`String[] config`*`)` — Mit dieser Methode kann ein neues Subjekt, das durch den Integrationsagenten repräsentiert und verwaltet werden soll, samt

 Als Buch erhältlich im Verlag Dr. Hut, München, www.dr.hut-verlag.de (ISBN 3-934767-97-4)

der für dieses Subjekt benötigten Konfigurationsdaten hinzugefügt werden. Die übergebenden Konfigurationsparameter enthalten dabei alle notwendigen Daten, die zu einer korrekten und erfolgreichen Initialisierung und u.U. auch erfolgreichem Betrieb des Subjekts notwendig sind. Für jeden neu hinzugefügten Konfigurationseintrag wird ein eindeutiger Identifikator erzeugt.

- `removeSubjectConfig(`*`String id`*`)` — Diese Methode löscht den Konfigurationseintrag eines nicht mehr benötigten Subjekts.
- `modifySubjectConfig(`*`String id, String[] config`*`)` — Hiermit wird eine bestehende Konfiguration eines dem Integrationsagenten zugewiesenen Subjekts geändert. Die übergebenen Konfigurationsparameter überschreiben den bereits vorhandenen Eintrag.
- `getSubjectConfig(`*`String id`*`)` — Es wird der zu einem Subjekt gehörende Konfigurationseintrag zurückgeliefert.
- `resetCompleteSubjectConfig()` — Alle Konfigurationseinträge werden gelöscht und auf Standardwerte zurückgesetzt, sofern welche vorhanden sind.
- `initSubject(`*`String id`*`)` — Ein Subjekt wird mit den im Konfigurationseintrag vorhandenen Werten initialisiert. In Abhängigkeit vom Subjekt muss dieses u.U. vor der Initialisierung erzeugt werden.
- `startSubject(`*`String id`*`)` — Die Ausführung des betreffenden Subjekts wird angestoßen. Der Aufruf dieser Methode ist nur nach der vorherigen Ausführung der Initialisierung erfolgreich.
- `stopSubject(`*`String id`*`)` — Die Ausführung des betreffenden Subjekts wird beendet. Der Aufruf dieser Methode ist nur sinnvoll und erfolgreich, wenn das Subjekt vorher gestartet wurde.
- `quitSubject(`*`String id`*`)` — Die zu Beginn durchgeführte Initialisierung des Subjekts wird rückgängig gemacht und das Subjekt u.U. vollständig entfernt. Diese Methode ist nur sinnvoll, wenn der Agent sich nicht in Ausführung befindet.

5.5.7 Verfeinerung der Schnittstelle eines Integrationsagenten

Diskussion von Designaspekten

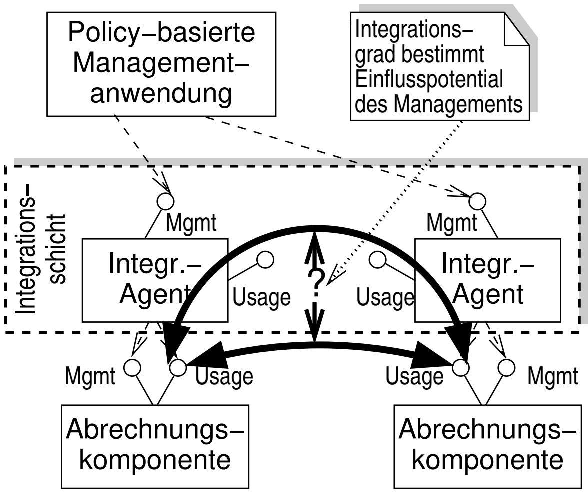

Abbildung 5.14: Aspekte der Integrationsschicht

Da die entworfene policy–basierte Managementarchitekur festlegt, dass spezifizierte Policies nicht direkt auf den zu managenden Ressourcen, die in diesem Fall die abrechnungsprozess–realisierenden HW/SW–Komponenten sind, agieren, sondern auf den Agenten der Integrationsschicht, ist konsequenterweise die Einflussmöglichkeit des Managements vom *Grad der Integration*, der durch die tatsächliche Implementierung der Integrationsagenten bestimmt wird, abhängig. Das heißt, die Realisierung der Integrationsschicht und damit die Art und Weise der Realisierung der Integrationsagenten bestimmt den Grad des Einflusses des Managements. Hierzu sind in Abbildung 5.14 schematisch die beiden Extreme bzgl. der Einbindung von vorgangsrealisierenden Komponenten, wie z.B. Abrechnungskomponenten, durch die Integrationsschicht dargestellt, die nachfolgend erläutert werden:

Management ausschließlich durch Konfiguration Einerseits besteht die Möglichkeit, dass die Komponenten über ihr jeweiliges Managementinterface zwar konfiguriert werden, aber der Vorgang an sich von den jeweiligen Komponenten autonom und ohne Einfluss von Seiten des Managements ausgeführt wird. Damit umfasst die Konfiguration nicht nur die Steuerung der eigentlichen Funktionalität einer Komponente, sondern auch die Kooperation mit anderen, am Vorgang beteiligten Komponenten. Dies heißt insbesondere, dass die zur Erbringung der Funktionalität notwendige direkte Kommunikation zwischen den Komponenten über die jeweiligen Nutzungsschnittstellen vollkommen unabhängig vom dazugehörigen Management abläuft. Damit wird das Einflusspotential des Managements auf den Ablauf des Vorgangs vom Umfang der durch die Management– bzw. Konfigurationsschnittstelle bereitgestellten Funktionalität einer Komponente beschränkt. In diesem Fall agiert der Integrationsagent ausschließlich in der Rolle eines Proxys.

Management durch direkte Beeinflussung der Interaktionen Andererseits besteht die Möglichkeit, dass die vorgangsrealisierenden Komponenten nicht nur initial konfiguriert werden, sondern dass in die jeweiligen Interaktionen zwischen den Komponenten aktiv von Seiten des Managements eingegriffen wird. Höchstmögliche Einflussnahme bietet sich genau dann, wenn jede *direkte* Kommunikation zwischen den vorgangsrealisierenden Komponenten unterbunden wird. Dies kann dadurch erreicht werden, dass jegliche Kommunikation zwischen den tatsächlich vorgangsrealisierenden Komponenten *indirekt* über die jeweiligen Integrationsagen-

 Als Buch erhältlich im Verlag Dr. Hut, München, www.dr.hut-verlag.de (ISBN 3-934767-97-4)

ten realisiert wird. Damit agiert der jeweilig für eine Komponente zuständige Integrationsagent in der Rolle des eigentlichen Kommunikationspartners. Beispielsweise würde der Integrationsagent einer Messkomponente dieser gegenüber als Kollektor agieren. Die Kommunikation zur tatsächlichen Kollektorsoftware würde allerdings ebenfalls nicht direkt zwischen Meter–Integrationsagent und Kollektor laufen, sondern immer über den dafür zuständigen Integrationsagenten. Hierzu werden die Agenten über die jeweiligen Nutzungsschnittstellen *Integration-Usage* miteinander gekoppelt. Damit agieren die Agenten in diesem Fall v.a. in der Rolle eines Mediators. Es liegt auf der Hand, dass auf diese Weise sich aus Sicht des Managements das Einflusspotential auf den Ablauf des Vorgangs wesentlich erhöht, da das Management nicht mehr ausschließlich auf die Konfigurationsschnittstelle der tatsächlichen Komponente beschränkt ist.

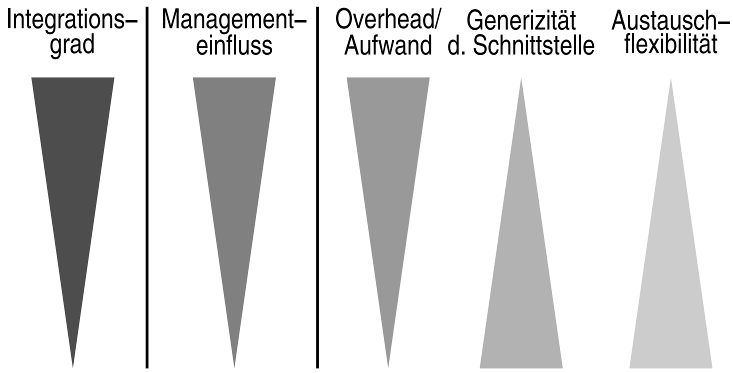

Abbildung 5.15: Zusammenhang zwischen Integrationsgrad und Managementeinfluss

Damit kann insgesamt festgestellt werden, dass je mehr die tatsächlich zu managenden Entitäten *direkt* ohne Einbeziehen der Integrationsschicht miteinander kommunizieren, desto weniger Einflussmöglichkeiten bieten sich dem Management, da der Ablauf des Vorgangs in diesen Fällen unabhängig und ohne Kenntnis des Managements stattfindet. Damit ist der Grad des Managementeinflusses *indirekt* vom Grad der direkten Kommunikation zwischen den am Prozess beteiligten Komponenten abhängig. Dies ist insbesondere beim Design der subjekteigenen, akteurspezifischen Schnittstellen und damit der Verfeinerung des *RealSubjectManagement*–Interfaces zu beachten. Hierbei steht die Erhöhung des Managementeinflusses im Zielkonflikt einerseits mit der Minimierung des entstehenden (Kommunikations–)Overheads und andererseits mit der Erhaltung der Generizität der akteurspezifischen Schnittstellen. Sollen die Möglichkeiten des Managements maximiert werden, so muss jede *direkte* Kommunikation zwischen den abrechnungsrealisierenden Komponenten unterbunden werden. Je indirekter die Kommunikation gestaltet wird, desto aufwendiger wird allerdings die Realisierung des Integrationsagenten und desto höher wird der Overhead. Zudem hat die Erhöhung der Einflussmöglichkeiten durch das Management selbstverständlich Auswirkungen auf den Entwurf der dazugehörigen, verfeinerten Managementschnittstelle *RealSubjectManagement* eines Integrationsagenten. Je feingranularer die subjektspezifische Managementschnittstelle ist, desto weniger generisch ist diese ausgelegt, da produktspezifische Besonderheiten der abrechnungsrealisierenden Komponente durchschlagen. Dies erschwert insbesondere den Austausch von Komponenten „gleichen Typs“, die von unterschiedlichen Herstellern stammen. Folglich ist beim Entwurf der verfeinerten, subjektspezifischen Schnittstellen auf einen guten Kompromiss zwischen Managementeinfluss und Generizität/Overhead zu achten. In Abbildung 5.15 ist eine zusammenfassende Gegenüberstellung der in diesem Absatz genannten Aspekte dargestellt.

Es muss betont werden, dass sich die Einflussmöglichkeiten durch den vorgeschlagenen Ansatz im Vergleich zu den bisherigen Ansätzen in keinem Fall verschlechtern. Lediglich beim Entwurf der Schnittstellen der tatsächlichen Integrationsagenten und damit bei der Verfeinerung der in Abschnitt 5.5.6 vorgestellten generischen Schnittstellen ist auf die erläuterten Zusammenhänge zu achten. Im Folgenden wird das Vorgehen zur Spezifikation einer konkreten Schnittstelle eines Integrationsagenten erklärt und exemplarisch vorgeführt.

Verfeinerung der subjekt–spezifischen, generischen Schnittstellen eines Integrationsagenten

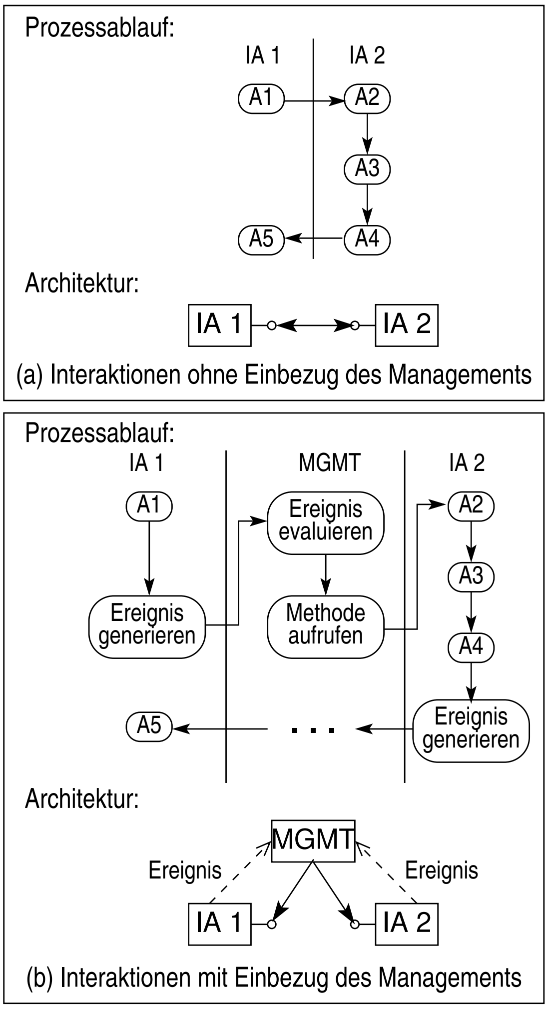

Abbildung 5.16: Einbezug des Managements in die Interaktionen der Integrationsschicht

Die Managementarchitektur sieht vor, dass jeder im Prozessmodell aus Kapitel 4 identifizierte Akteur durch einen Integrationsagenten repräsentiert wird, der gegenüber der Managementanwendung als *Proxy* für eine tatsächliche vorgangsrealisierende Komponente agiert und als *Mediator/Gateway* zwischen Agenten der Integrationsschicht (und damit zwischen den repräsentierten, tatsächlichen Komponenten) agiert. Eine Ausnahme bilden Akteure, die durch Personen eingenommene Rollen darstellen. Diese werden, wie bereits in Abschnitt 5.4.1 kurz erwähnt, durch einen gesonderten Agenten repräsentiert, der alleinig eine Schnittstelle zur Benachrichtigung der Personen, die in dem gegebenen Kontext die jeweilige Rolle einnehmen, vorweist. Damit ist eine minimale Unterstützung einer Workflowsteuerung vorgesehen. Idealerweise gelingt es durch diesen Rollen–Agenten eine Kopplung zu bestehenden Workflowsystemen zu erreichen[16].

Für alle anderen Akteure, die üblicherweise durch technische HW/SW–Komponenten, wie z.B. Messkomponenten, Kollektorsoftware, Billing Systeme, etc., eingenommen werden, wäre es denkbar, eine für diesen Akteur spezifische, aber dennoch, bezogen auf die konkrete Komponentenausprägung eines Herstellers, generische Minimalschnittstelle zu spezifizieren. Diese Minimalschnittstelle würde einerseits eine Festle-

[16]Dieser Sachverhalt wurde allerdings im Rahmen dieser Arbeit nicht weiter verfolgt.

gung der Semantik der Konfigurationsschnittstelle vorsehen (in vorliegenden Fall demnach die Spezifikation der Felder des *config[]*–Stringarrays der *addSubjectConfig()*–Methode des *RealSubjectManagement*–Interfaces) als auch die Definition einer akteur–eigenen und –spezifischen Managementschnittstelle, die zudem die Interaktionen mit anderen Agenten aktiv zu steuern vermag.

Wie bereits in Abschnitt 5.3.3 auf Seite 145 erläutert wurde, ist jede Interaktion zwischen Akteuren auch gleichzeitig ein Aktivitätsübergang, der implizit bzw. explizit mit einem Ereignis zusammenhängt. Somit kann die generische, akteur–eigene „Managementschnittstelle“ auf die Spezifikation eines Satzes von Ereignissen reduziert werden, welche die Ausführung einer Folge von Aktivitäten durch andere Akteure initiiert. Insgesamt werden damit demnach alle Interaktionsübergänge explizit gemacht durch (a) die Generierung von Ereignissen und (b) das aktive Anstoßen einer Folge von Aktivitäten durch das Management (vgl. Abbildung 5.16). Wenn die Ausführung der jeweilig relevanten Aktivitätsfolgen durch entsprechende *Methoden* von den Integrationsagenten angeboten wird, kann jede Prozessausführung bis auf die Granularität von Interaktionsübergängen unter die Kontrolle des policy–basierten Managements gebracht werden. Voraussetzung hierfür ist allerdings, dass alle relevanten Ereignisse, die einen Interaktionsübergang initiieren können, generiert werden und die Ausführung aller (abgeschlossenen) Aktivitätsfolgen durch Methoden an einer Schnittstelle durch die Agenten angeboten werden. Dies muss nun durch die Verfeinerung bzw. Präzisierung der bestehenden Schnittstellen und einer geeigneten Implementierung der Integrationsagenten sichergestellt werden.

Wie im vorherigen Abschnitt diskutiert wurde, muss bei der Verfeinerung/Präzisierung der Schnittstelle eines Integrationsagenten auf das richtige Maß zwischen Generizität und Integrationsgrad geachtet werden. Im vorliegenden Fall müssen die Konfigurationsmethoden der *RealSubjectManagement*–Schnittstelle präzisiert sowie die konkreten, von einem akteur–spezifischen Agenten zu generierenden Ereignisse spezifiziert werden. Als Hilfsmittel und damit zur Identifizierung des richtigen Maßes können hierzu das Prozessmodell und insbesondere die Teilprozessmodelle aus Kapitel 4 folgendermaßen für diese Aufgabe herangezogen werden:

- *Konfigurationsmanagement:* Hierfür kann das Teilprozessmodell „Konfiguration von Abrechnungskomponenten“ aus Abschnitt 4.3.8 als Grundlage herangezogen werden, das für jede zu konfigurierende Komponente eine Beschreibung generischer Konfigurationsparameter enthält. Hierbei muss die Entscheidung getroffen werden, was tatsächlich als Konfigurationsparameter angebbar ist und welche Konfigurationen „intern“ vom Integrationsagenten vorgenommen werden sollen, die sich nicht durch Setzen von Parametern beeinflussen lassen und somit statisch und hartkodiert vorliegen. Es liegt auf der Hand, dass je weniger Einstellparameter für die Konfiguration angeboten werden, desto leichter gestaltet sich die Konfiguration für den Aufrufer. Allerdings leidet darunter die Generizität des dadurch entworfenen und implementierten Integrationsagenten.

- *Interaktionsmanagement/Ereignisse:* Hierfür können insbesondere die Teilprozessmodelle (vgl. Abschnitt 4.3.11–4.3.17) aus der Betriebsphase als Grundlage verwendet werden. Jeder Interaktionsübergang zwischen Integrationsagenten verpflichtet im vorliegenden Fall den Interaktionsursprungs–Agenten zur Generierung eines Ereignisses. Der Interaktionsziel–Agent hingegen muss eine Methode bereitstellen, welche die Ausführung der dann folgenden Aktivitäten anstößt. Somit müssen lediglich die Interaktionen der Teilprozessmodelle anhand der bildlichen Darstellung identifiziert werden, um diese für die Spezifikation von Methoden resp. Ereignissen in der beschriebenen Weise zu nutzen.

Das skizzierte Vorgehen zur Präzisierung der Managementschnittstelle wird beispielhaft für den Akteur „Kollektor“ durchgeführt. Für die übrigen Akteure ist das Vorgehen analog.

Konfigurationsmanagement Wie bereits erwähnt, betrifft die Präzisierung des Konfigurationsmanagements immer die Präzisierung der Methode `addSubjectConfig(`*`String[] config`*`)` der *RealSubjectManagement*–Schnittstelle (vgl. Abschnitt 5.5.6, Seite 194). Im vorliegenden Fall wird die Beschreibung der zu konfigurierenden Parameter eines Kollektors (vgl. Abschnitt 4.3.8, Seite 101) für diesen Vorgang herangezogen. Im Folgenden werden die einzeln genannten Konfigurationsparameter daraufhin untersucht, ob diese als Einstellungsparameter realisiert werden sollen oder statisch innerhalb des Agenten verwirklicht werden:
Die *IDs der Messkomponenten*, das *Einsammelintervall*, das *Einsammelmodell* sowie die *IDs der weiterverarbeitenden Anwendungen* sollten nicht durch die Konfigurationsmethode des Integrationsagenten einstellbar sein, da dies Interaktionen mit anderen Agenten betrifft und dies durch das Management explizit gesteuert wird. Somit bleiben die folgenden Parameter für das Konfigurationsarray *`String[] config`* übrig:

- *Felder eines zu generierenden Usage Records:* Dieser Parameter kann als Komma–separierte Liste von Feldernamen realisiert werden, welche aus den erhaltenden *Meter Records* der Messagenten für die Erstellung der *Usage Records* benötigt werden.
- *Interpretation der Meter Records:* Wie in Abschnitt 4.3.11 bereits beschrieben wurde, wird die Interpretation durch Anwendung von Aggregationsfunktionen realisiert. An dieser Stelle wäre es sinnvoll, dass der Kollektor einen Satz von internen Aggregationsfunktionen besitzt und als Konfiguration lediglich der Name der Funktion sowie die Aufrufparameter, bspw. anhand der Reihenfolge der unter dem ersten Punkt angegebenen Feldernamen, übergeben wird.
- *Dienst–ID:* Dieser Identifikator wird für die Zuordnung der erstellten Usage Records zu eindeutigen Accounting User/Customer IDs benötigt und sollte direkt angegeben werden können.
- *Aufbewahrungszeit:* Diese Information sollte jeweils in Tagen, Wochen, Monaten oder Jahren angebbar sein.

Ereignisse Der Kollektor interagiert als Interaktions–Ursprung mit der Charging–Komponente und dem Reporting Tool. Daraus ergibt sich als interaktionsauslösendes Ereignis:

- `UsageRecordGenerated`: Dieses Ereignis sollte zumindest die Kollektor–ID, die Dienst–ID, den Generierungszeitpunkt sowie die Aufbewahrungszeit mitübertragen.

 Als Buch erhältlich im Verlag Dr. Hut, München, www.dr.hut-verlag.de (ISBN 3-934767-97-4)

Methoden Untersucht man jeweils die drei Teilprozesse „Nutzungserfassung“ (siehe Abschnitt 4.3.11), „Gebührenberechung“ (siehe Abschnitt 4.3.12) und „Reporting“ (siehe Abschnitt 4.3.14), an denen der Kollektor beteiligt ist, so ergeben sich die folgenden beiden Methoden:

- `generateUsageRecord(`*`String meterDomain`*`)`: Damit wird der Vorgang zur Erstellung eines Usage Records angestoßen. Hierzu fragt der Kollektor die ihm zugeordneten Messagenten ab und erzeugt auf Basis der empfangenen Meter Records ein Usage Record.
- `getUsageRecords(`*`Time start, Time end`*`)`: Mit dieser Methode werden die im übergebenen Zeitraum erstellten Usage Records zurückgeliefert.

Ebenso ist es denkbar, diese beiden Methoden durch die generische *execQuery()*–Methode der *IntegrationUsage*–Schnittstelle zu realisieren.

Analog zu dieser exemplarisch durchgeführten Spezifikation läßt sich für die übrigen Akteure eine subjekt–spezifische, aber dennoch generische Verfeinerung und Präzisierung von Managementmethoden und –ereignissen vollziehen.

5.6 Beispiel–Policies für das prozessorientierte Abrechnungsmanagement

In diesem Abschnitt werden einige Beispiel–Policies und –Meta-Policies vorgestellt. In den Abschnitten 5.4.2 und 5.4.3 wurden bereits die Teilprozesse durch Anwendung der in Abschnitt 5.4.1 vorgestellten Methodik auf allgemeingültige Aussagen bzgl. der Spezifikation von Policies und Meta–Policies untersucht und damit bereits die Menge der möglichen resp. notwendigen *subjects, targets, action* und *constraints* identifiziert. Die nachfolgend präsentierten Policies sind somit eine Präzisierung dieser zwar bzgl. des zu managenden Teilprozesses spezifischen, aber bzgl. einer konkreten Umsetzung allgemein gehaltenen Feststellungen. Da die im Nachfolgenden vorgestellten Policies z.T. komplex aussehen, soll an dieser Stelle bereits der Hinweis erfolgen, dass die prototypische Implementierung die Deklaration von Policies im XML–Format vorsieht. Dies heißt insbesondere auch, dass der Policy–Ersteller durch einen entsprechenden XML–Editor lediglich die von ihm auszufüllenden Felder zu deklarieren hat und somit die explizite Angabe von Schlüsselwörtern (wie *event*, *action*, etc.) sowie Klammern entfällt. Diese werden durch entsprechende XML–Tags realisiert, die in Editoren durch Einsatz entsprechender PDL–XML–Templates auch durch farbliche Kennzeichnung hervorgehoben werden. Durch eine Werkzeugunterstützung der in Abschnitt 5.4.1 vorgestellten Methodik wird die Policy–Erstellung zusätzlich erleichtert.

Im nachfolgenden Listing 5.1 wird die Implementierung eines Messagenten durch eine Policy ausgedrückt. Wie bereits in Abschnitt 5.4.2 festgestellt wurde, läßt sich dieser Teilprozess nur z.T. automatisieren, da eine Interaktion mit einem nicht–technischen Akteur (in diesem Fall mit dem Entwickler) notwendig ist. Ziel dieser Policy ist es, bei Änderungen am Tarifmodell, die sich durch das Hinzufügen von Abrechnungseinheiten äußern, die Implementierung eines hierfür notwendigen Messagenten und Sensors anzustoßen.

Listing 5.1: Policy für die Implementierung neuer Messagenten

```
policy{
  relatedProcess{meter−implementation} ∗ Policy steuert den Implementierungsteilprozess ∗
  domain{customer−1} ∗ Standarddomaene (als Prefix verwendet) ∗
  subjectSet{subject{role{Entwickler}}}
  targetSet{target{Entwicklungsumgebung}}
  eventSet{event{name= newTarifModel}} ∗ Evaluierung immer dann, wenn Aenderung an Tarifmodell ∗
  actionSet{
    ∗ Default− Aufrufzielobjekt ist Target ∗
    action{default{invoke{
    ∗ Erzeuge Standard−Messagent−Skelett mit Namen = neue Abrechnungseinheit ∗
          createMeterSkeleton (string{event.acctUnitName})}}}
    action{default{invoke{
    ∗ Benachrichtige Entwickler ∗
          subject. notify (
          string{"Tarifmodellaenderung: Sensor/Messagent implementieren"})}}}
  }
  ∗ Aktionen nur ausfuehren, falls nicht bereits Messagent existiert ∗
  constraint {! boolean{agentExists(string{newTarifModel.acctUnit})}}
}
```

Die auf der gegenüberliegenden Seite in Listing 5.2 formulierte Policy bezieht sich auf die Umkonfiguration eines bestehenden Kollektors, sofern sich das Format des zu erzeugenden Usage Records ändert.

 Als Buch erhältlich im Verlag Dr. Hut, München, www.dr.hut-verlag.de (ISBN 3-934767-97-4)

Listing 5.2: Policy für die (Re–)Konfiguration eines Kollektors

```
policy{
  relatedProcess{ configuration }
  subjectSet{subject{role{Administrator}}}
  targetSet{target{customer-1/IESA/mgmt/collector}}
  eventSet{event{name= newUsageRecordFormat}} * Falls Aenderung an Format des UsageRecords *
  actionSet{
    action{default{invoke{
    * Umkonfiguration des Kollektors auf Erstellung des neuen Formats *
          modifySubjectConfig (string{ getCollectorID ()},
               string{array{length= event.formatLength
               event.newFormat * enthaelt als String-Array die neuen Felder *
               aggregateOctets
               customer-1/IESA
               1 year}})}}}
  }
  * Umkonfiguration nur, wenn Kollektor bereits existiert *
  * Ansonsten separate Initialkonfigurationspolicy notwendig *
  constraint{boolean{ collectorExists (customer-1/mgmt/collector)}}
}
```

Die nachfolgend formulierte Policy (vgl. Listing 5.3) wird zum Management der Betriebsphase und damit zur aktiven Steuerung der Interaktionen zwischen den Integrationsagenten verwendet. Im vorliegenden Fall wird die Interaktion und damit die Kopplung zwischen Messagent und Kollektor überwacht und gesteuert. Sofern ein neuer Meter Record von einem Messagenten erzeugt wurde, wird der hierfür zugewiesene Kollektor mit der sofortigen Erzeugung eines Usage Records angewiesen.

Listing 5.3: Policy zur Steuerung der Interaktion zwischen Messkomponente und Kollektor

```
policy{
  relatedProcess{usage-accounting}
  domain{customer-1/}
  subjectSet{subject{IESA}}
  targetSet{target{mgmt/collector}}
  eventSet{event{name= newMeterRecordGenerated}} * Ein Meter Record wurde erzeugt *
  actionSet{
    * Erzeuge Usage Record auf Basis der dem Kollektor zugewiesenen Messagenten *
    action{default{invoke{
          generateUsageRecord(string{event.meterID})}}}
  }
  * Aktionen nur dann ausfuehren, wenn Ereignis von einem Messagenten erzeugt wurde, *
  * der dem Kollektor zugewiesen ist *
  constraint{equal{string{ repository . resolveCollector (
        string{event.meterDomain})},string{target}}}
```

In Listing 5.4 ist eine Meta–Policy formuliert, die sich um die Deaktivierung von Policies kümmert, die einem Kollektor zugewiesen sind. Die einen Kollektor betreffenden Policies werden durch Suche über Policy–Targets gefunden.

Listing 5.4: Meta–Policy zur Deinstallation von Kollektor–Policies

```
policy{
  subjectSet{subject{/mgmt/policies / collector /}}
  targetSet{target{/mgmt/policyManagerAgent}}
  eventSet{event{name= collectorDeinstalled }}
  actionSet{
    * Deaktiviere  Policies *
    action{default{invoke{
             deactivatePolicy (invoke{
               * Verschachtelter  Aufruf  zur  Suche von  Policies *
               execQuery(*Finde alle  Policies  deren  Target  der  Kollektor  ist *)})}}
            }}
}
```

5.7 Zusammenfassung

In diesem Kapitel wurde der neu entwickelte Lösungsansatz vorgestellt, der durch Anwendung policy–basierter Konzepte eine prozessorientierte Sichtweise auf die Managementaufgabe realisiert. Da der grundsätzliche Ansatz nicht auf das dienstorientierte Abrechnungsmanagement beschränkt ist, sondern in gleicher Art und Weise auf andere Managementfunktionsbereiche anwendbar ist, wurde bei den im Rahmen dieses Kapitels erarbeiteten Entwürfen zur Realisierung des Ansatzes auf einen möglichst generische Umsetzung geachtet. Insgesamt ist die entwickelte Policy–Sprache, die spezifizierte Methodik und die der objektorientierte Entwurf des Managementsystems allgemein für das Management von IT Prozessen einsetzbar und nicht aussschließlich auf das dienstorientierte Abrechnungsmanagement beschränkt.

Zunächst wurde basierend auf der grundlegenden Lösungsidee ein erster Grobentwurf der Managementarchitektur entwickelt. Diese sieht im Wesentlichen eine Integrationsschicht aus Integrationsagenten zwischen der policy–basierten Managementanwendung und den tatsächlich zu managenden, vorgangsrealisierenden HW/SW–Komponenten vor. Die Integrationsschicht ermöglicht auf diese Weise neben einer prozessorientierten Sicht auf den zu managenden IT Prozess auch einen einheitlichen Zugriff auf Nutzungs– und Managementschnittstellen und verschattet so die Heterogenität der tatsächlichen Komponenten.

Basierend auf diesem ersten Architekturentwurf wurde die für das Einsatzgebiet notwendige Ausdrucksmächtigkeit einer Policy–Sprache untersucht, um damit, unter Berücksichtigung bereits existierender Policy–Sprachkonzepte, Sprachbausteine einer PDL für das prozessorientierte IT Management zu identifizieren und zu analysieren. Ergebnis war der Entwurf einer vollständigen Grammatik einer PDL, die Voraussetzung für die Entwicklung eines geeigneten Werkzeugs zum automatisierten Durchsetzen von Policies ist. Bei der PDL–Grammatik wurde ebenfalls darauf geachtet, dass eine hohe Wiederverwendbarkeit von Teilen der Policy–Spezifikation gesichert ist.

Im Anschluss wurde eine allgemeine Methodik zur Spezifikation von Policies und Meta–Policies für das prozessorientierte IT Management entwickelt, die den Policy–Ersteller anleitet und damit als Hilfestellung dient. Bezüglich des Abrechnungsmanagements ist es darüber hinaus gelungen, abrechungsteilprozess–spezifische Aussagen über das Automatisierungspotential des Managements sowie die Menge der in Frage kommenden Subjects, Targets, Events, Actions und Constraints anzugeben. Generell kann gesagt werden, dass die in Kapitel 4 entwickelten Teilprozessmodelle als Schablonen für die Policy–Spezifikation verwendet werden können. Da die entwickelte Policy–Sprache die Wiederverwendung von Policy–Teilen unterstützt, kann ein Managementwerkzeug entwickelt werden, dass den Policy–Ersteller wesentlich bei dessen Aufgabe unterstützt.

Zum Abschluss dieses Kapitels wurde ein objektorientierter Entwurf des policy–basierten Managementsystems, bestehend aus der policy–basierten Managementanwendung und der Integrationsschicht, vorgestellt. Im Wesentlichen wurde die Package–Struktur sowie die wichtigsten Klassen und Schnittstellen erläutert. Beim Entwurf wurde auf die Anwendung von wohlbekannten Entwurfsmustern (Design Patterns) aus dem Software–Engineering–Bereich geachtet. Dieser objektorientierte Entwurf dient nun im nächsten Kapitel als Grundlage für die Entwicklung einer prototypischen Implementierung.

 Als Buch erhältlich im Verlag Dr. Hut, München, www.dr.hut-verlag.de (ISBN 3-934767-97-4)

Kapitel 6

Prototypische Implementierung

Als Tragfähigkeitsnachweis des in den vorangegangenen Kapiteln dieser Arbeit entwickelten Lösungsansatzes wird in diesem Kapitel eine prototypische Implementierung vorgestellt. Hierfür wird zunächst die verwendete, auf Java/CORBA–basierende Laufzeit– und Entwicklungsumgebung eingeführt. Darauf basierend werden neben der Umsetzung der policy–basierten Managementanwendung auch einige implementierte Agenten der Integrationsschicht beschrieben. Zum Schluss erfolgt eine Gesamtbewertung des in dieser Arbeit entwickelten Lösungsansatzes durch Anwendung des in Kapitel 2 erstellten Anforderungskatalogs.

6.1 Einführung

In diesem Abschnitt wird die gewählte, auf Java/CORBA sowie mobilen Agenten basierende Laufzeit– und Entwicklungsumgebung beschrieben. Ein Schwerpunkt der Erläuterungen liegt hierbei auf der Darstellung der sich dadurch für das Management ergebenden Vorteile.

6.1.1 Wahl der Middleware: CORBA und Management

Bei der von der *Object Management Group (OMG)*[1] standardisierten *Object Management Architecture (OMA)* [OMG 92-11-1], in der die *Common Object Request Broker Architecture (CORBA)* [OMG 01-02-33] ein Teil der Spezifikation ist, handelt es sich nicht um eine rein auf das Management von vernetzten Systemen ausgerichtete Architektur. Vielmehr wird mit OMA eine allgemeine Architektur für die Spezifikation von beliebigen verteilten Systemen und Anwendungen festgelegt. Eine Managementanwendung wird hierbei als eine mögliche Ausprägung einer verteilten Anwendung angesehen. Damit haben Nutz– und Managementdaten nicht nur dieselbe Kommunikationsarchitektur, sondern werden auch auf einheitliche Weise modelliert und implementiert. Folglich wird die Definition des Managementteils einer Anwendung

[1] `http://www.omg.org/`

resp. eines Dienstes parallel zum Design der eigentlichen Nutzungsfunktionalität durchgeführt. Somit entfällt die Notwendigkeit nach Spezialwissen bei der Modellierung von Managementinformation, wie es üblicherweise bei anderen Managementarchitekturen (z.B. Internet Management) der Fall ist.

Neben der mit der *Interface Description Language (IDL)* [OMG 01-02-39] gebotenen Möglichkeit, Managementinformation programmiersprachenunabhängig und objektorientiert zu definieren, bietet CORBA durch die Vielzahl an spezifizierten und bereits implementierten CORBAservices [OMG 98-12-9] eine breite Palette an anwendungsunabhängigen Basisdiensten, deren Funktionalität insbesondere für das Management eine hohe Relevanz hat. Damit stellt CORBA einen sehr flexiblen Ansatz dar, um als Managementarchitektur eingesetzt zu werden und insbesondere dadurch die Implementierung von Managementlösungen zu erleichtern. Ebenso hat sich gezeigt, dass CORBA als integrierende Infrastruktur im Rahmen eines Umbrella Managements [Kell 98] für kooperierende Managementsysteme sehr geeignet ist. Der Overhead, der durch CORBA entsteht, darf allerdings nicht verschwiegen werden, so dass sich der Einsatz des CORBA–Ansatzes im Management v.a. in einer heterogenen, komplexen Umgebung eignet, in der der Overhead vernachlässigbar ist. Dass sich CORBA tatsächlich für den Einsatz im Managementumfeld eignet, ist v.a. daran ersichtlich, dass kommerzielle Managementplattformen, die auf das System– und Enterprise–Management ausgerichtet sind, wie beispielsweise das Tivoli Management Framework[2], CORBA als Kommunikationsinfrastruktur zwischen Manager und Agenten einsetzen.

Damit wird CORBA als die geeignete Managementarchitektur für die prototypische Implementierung der Managementanwendung angesehen. Am Lehrstuhl für Kommunikationssysteme und Systemprogrammierung der LMU München (Prof. Dr. H.-G. Hegering) wurde in zahlreichen Arbeiten die sog. *Mobile Agent System Architecture (MASA)* [GHR 99] spezifiziert und implementiert, die eine auf Java/CORBA und flexiblen Agenten basierende Architektur für das Management darstellt und bereits in zahlreichen Forschungsarbeiten erfolgreich als Entwicklungs– und Ablaufumgebungen für prototypische Implementierungen im Managementumfeld gedient hat. Im nachfolgenden Abschnitt wird MASA näher beschrieben.

6.1.2 Die Mobile Agent System Architecture (MASA)

Mit der *Mobile Agent System Architecture (MASA)* [GHR 99, KRRV01] wurde eine Plattform für Mobile Agenten entwickelt, deren Kern der sogenannten *Mobile Agent System Interoperability Facilities (MASIF)*–Spezifikation [MASIF] der *Object Management Group (OMG)* entspricht und die eine Weiterentwicklung des vor allem auf das Netz- und Systemmanagement ausgerichteten Konzepts der *flexiblen Agenten* [Moun 97] ist. Im Folgenden wird ein kurzer Überblick über die Bestandteile, Konzepte und die Funktionalität von MASA gegeben, das als Entwicklungs– und Ablaufumgebung für die prototypische Implementierung des in Kapitel 5 vorgestellten Ansatzes ist.

[2] `http://www.tivoli.com/`

 Als Buch erhältlich im Verlag Dr. Hut, München, www.dr.hut-verlag.de (ISBN 3-934767-97-4)

Die Architektur von MASA

Abbildung 6.1 zeigt eine *MASA–Region* bestehend aus einigen *MASA–Agentensystemen*. Die Agentensysteme stellen die Ablaufumgebung für *MASA–Agenten* zur Verfügung, wobei i.d.R. jeweils genau ein Agentensystem auf jedem Endsystem installiert ist, auch wenn parallel mehrere Agentensysteme nebeneinander ausgeführt werden können.

Die Agenten sind zunächst für sich gesehen autonome Softwarekomponenten, die z.B. einen Dienst implementieren oder als Proxy–Agent zu einem nicht-MASA-konformen Dienst agieren. Ein Agent entspricht einem Prozess in einem Betriebssystem. Agenten sind die Verwaltungseinheit eines Agentensystems, das somit, in beschränkter Art und Weise, dem Agenten in der Managerrolle gegenübertritt. Zusätzlich bietet MASA *mobilen* Agenten die Möglichkeit, während des laufenden Betriebs auf ein anderes Agentensystem zu *migrieren*. Damit kann der Forderung nach einer hohen Verfügbarkeit und Performanz Rechnung getragen werden: ist ein Endsystem überlastet oder ist es in Begriff, heruntergefahren zu werden, so besteht für die mobilen Agenten die Möglichkeit, auf ein anderes System zu wechseln.

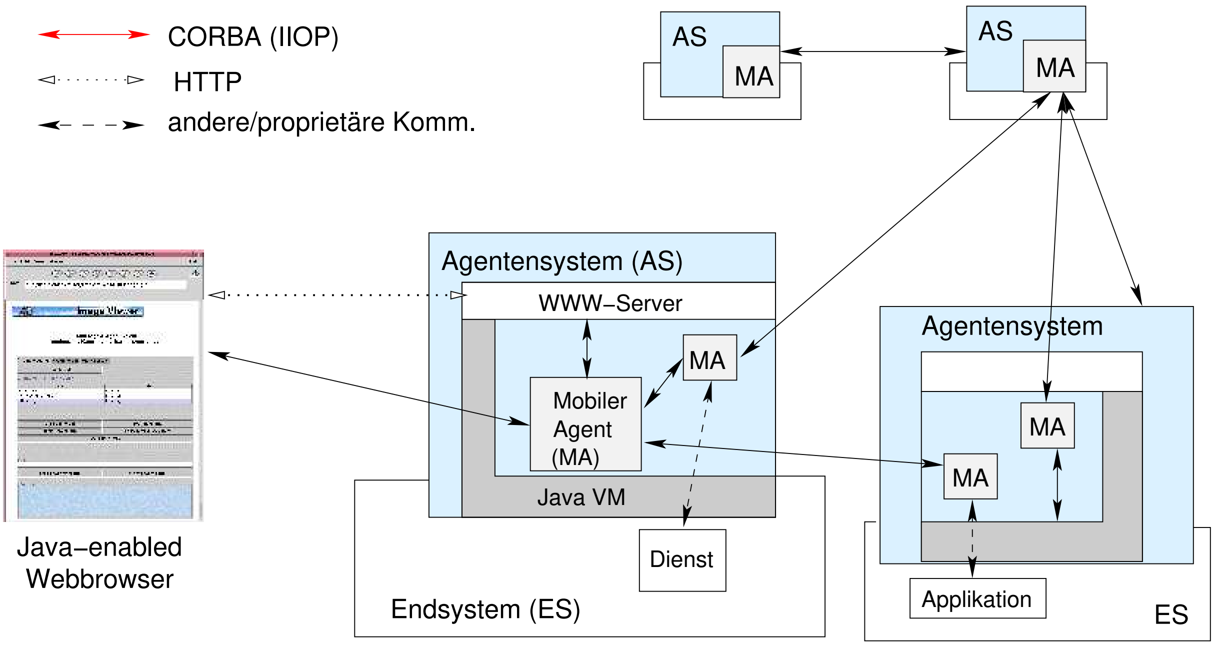

Abbildung 6.1: Überblick über die MASA–Architektur (aus [GHR 99])

Um die Interaktion mit einem Benutzer zu ermöglichen, ist in jedem MASA–Agenten eine eigene WWW–basierte Oberfläche integriert, die auf diese Weise von einem beliebigen Web–Browser aus aufgerufen werden kann. Die Oberfläche ist meist durch ein in einer HTML Seite eingebettetes Java Applet realisiert und bietet auf diese Weise eine komfortable Bedienungsumgebung.

Das Agentensystem selbst bietet ebenfalls eine derartige Web–Oberfläche an, die im Wesentlichen den in MASIF spezifizierten Funktionsumfang eines Agentensystems für einen Benutzer zugänglich macht.

Die MASA–Architektur besteht aus mehreren Schichten, die in Abbildung 6.2 dargestellt sind. Das vernetzte System selbst, dargestellt durch Betriebs– und Transportsystem, ist gegeben. Auf der nächst höheren Ebene muß zunächst eine einheitliche Umgebung für die verteilte Anwendung MASA geschaffen werden. Dies wird durch Java als Ausführungsplattform und CORBA als Transportsystem und Lieferant essentieller Basisdienste erreicht.

Java bietet den entscheidenden Vorteil der Hardwareunabhängigkeit der Ausführungsumgebung durch das Konzept der *Interpretierung* von Java–Programmen durch die *JAVA Virtual Machine (VM)*. Damit wird insbesondere der Einsatz des Agentensystems in heterogener Umgebung unterstützt.

CORBA als standardisierte Middleware bietet eine große Anzahl spezifizierter Basisdienste, die die Basisfunktionalität des Agentensystems erweitert.

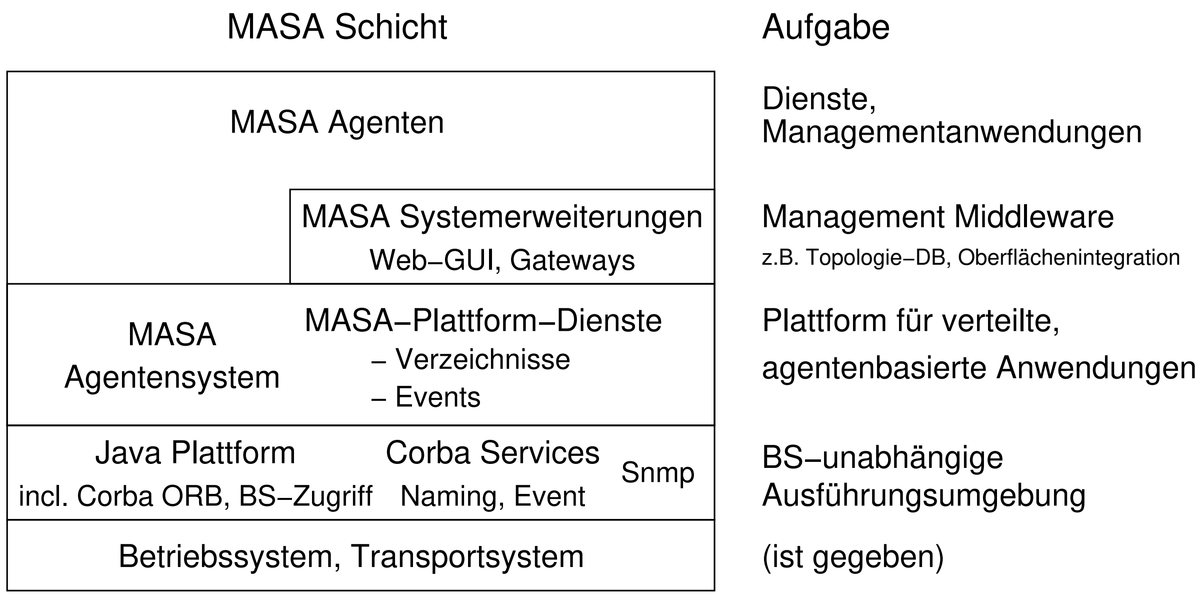

Abbildung 6.2: Die Schichtenarchitektur von MASA (nach [GHR 99])

Neben den in einem Agentensystem festintegrierten Plattform–Basisdiensten, werden weitere durch Agenten realisierte MASA–Dienste angeboten, die die Funktionalität von MASA völlig anwendungsunabhängig erweitern. Die Plattform–Basisdienste wären mit dem Betriebssystemkern zu vergleichen, während die durch Agenten realisierten MASA–Dienste analog zu Systemprozessen zu betrachten sind. Anwendungsbezogene Dienste, die als Agenten verwirklicht werden, können die oben beschriebene Infrastruktur nutzen.

An dieser Stelle lag das Interesse bisher vor allem in der Entwicklung von *Management Agenten*, die eine Steuerung und Überwachung anderer MASA–Agenten und den dadurch realisier-

 Als Buch erhältlich im Verlag Dr. Hut, München, www.dr.hut-verlag.de (ISBN 3-934767-97-4)

ten Dienst ermöglichen.

Bisher ist es mit diesem Konzept gelungen, neben einer Managementplattform, eine umfangreiche, durch zahlreiche Agenten verteilt realisierte Managementanwendung zu verwirklichen. Viele Aspekte und Konzepte des Managements verteilter Systeme, wie in [HAN 99a] dargestellt, wurden bereits innerhalb der MASA–Umgebung umgesetzt.

Um die bereits entwickelten Stand–alone–Managementanwendungen und nicht-MASA-konformen Dienste miteinbeziehen zu können, sind Proxy Agenten als Gateway entwickelt worden (SNMP, DPI, RMI, etc.).

6.1.3 Diskussion der Verwendung von MASA

Mit MASA als Ausführungs– und v.a. Entwicklungsplattform bieten sich viele Vorteile, die im Nachfolgenden kurz aufgezählt werden:

- *Java als Programmiersprache*
  Java bietet die bekannten Vorteile objektorientierter Programmiersprachen (Generizität, Polymorphie, Datenkapselung, etc.). Zudem können Standard CASE–Tools zur Entwicklung von Managementanwendungen eingesetzt werden.

- *CORBA als Middleware*
  Wie bereits in Abschnitt 6.1.1 erwähnt, bietet CORBA neben der Verschattung der Kommunikationsinfrastruktur zahlreiche bereits implementierte Basisdienste wie den Naming Service und den Event Service. Insbesondere wird damit ein domänenübergreifendes Management wesentlich erleichtert, so dass Anforderung MGT 6 erfüllt wird. In MASA wird als ORB derzeit Orbacus Version 3.1 [ORBacus] eingesetzt, der zusätzlich durch Aufsetzen auf SSL eine vollständige Verschlüsselung der Kommunikation zwischen den Agenten bietet.

- *Zahlreiche, bereits realisierte Basisdienste*
  Neben den mitgelieferten CORBA Basisdiensten wurden zahlreiche weitere (CORBA) Dienste in Form von MASA–Agenten realisiert: Topology Service [Roel 99], Notification Service [Maul 99], MAFFinder [Gigl 00], etc. Neben der Umsetzung dieser standardisierten Dienste, wurden zahlreiche weitere Managementdienste und –anwendungen implementiert, die in unterschiedlichen Managementszenarien eingesetzt werden können.

- *Funktionalität dynamisch zur Laufzeit erweiterbar*
  Zur Laufzeit lassen sich beliebig viele MASA–Agenten, die eine bestimmte Funktionalität erfüllen, starten, anhalten und migrieren. Zusätzlich bietet sich die Möglichkeit durch Nutzung von CORBA DSI und Java Classloader die Funktionalität einzelner Agenten zur Laufzeit zu erweitern, welches die Anforderung MGT 7 erfüllt.

- *Dezentrale Implementierung der Managementanwendung*
  Vordergründiges Einsatzziel von MASA–Agenten ist nicht, wie man aus dem Namen

fälschlicherweise vermuten könnte, die Implementierung von Managementagenten. Statt dessen sollen MASA–Agenten für die dezentrale, verteilte Realisierung von Managern und Managementanwendungen verwendet werden. Damit kann das Management robust gegenüber Fehlern als auch ausfallsicher gestaltet werden. Dies erfüllt die Anforderungen MGT 9 und MGT 10.

- *Sicherheit zentraler Bestandteil von MASA*
  In [Roel 99a] und [Reis 01] wurde die Sicherheit von Agentensystemen untersucht sowie ein umfassendes Konzept entwickelt und in MASA implementiert. Das Sicherheitskonzept sieht sowohl eine Autorisierung als auch Authentifizierung von allen an MASA beteiligten Entitäten vor. Sicherheit ist v.a. in Hinblick auf das Abrechnungsmanagement ein relevantes Kriterium, da sehr sensible Daten zwischen mehreren Komponenten ausgetauscht werden müssen. Aus diesem Grund wird in der IETF Working Group AAA versucht, das Accounting immer in Bezug zu Sicherheitsaspekten zu sehen. Bei MASA als Ausführungsplattform ist dies ein Built–In–Feature.

Damit bietet MASA eine ideale Umgebung für ein Quick Prototyping im Managementumfeld.

6.2 Beschreibung der prototypischen Implementierung

In diesem Abschnitt werden nach einem Überblick über die Implementierungsarchitektur zunächst die Bestandteile der policy–basierten Managementanwendung beschrieben. Anschließend werden einige, realisierte Integrationsagenten vorgestellt. Die policy–basierte Managementanwendung ist hierbei allgemein für das prozessorientierte IT Management anwendbar und somit nicht auf das Abrechnungsmanagement beschränkt.

6.2.1 Überblick

In Abbildung 6.3 ist ein Überblick über die realisierte Architektur der Implementierung sowie das Zusammenspiel der einzelnen Komponenten dargestellt, welche nachfolgend erläutert werden.

Die *Mobile Agent System Architecture (MASA)* wird demnach als Entwicklungs– und Ablaufumgebung für die zu entwickelnde Managementanwendung gewählt. Zusammenfassend kann MASA grundsätzlich in Agenten, welche die eigentliche dienst– und managementspezifische Funktionalität implementieren, und in das Agentensystem eingeteilt werden, das die Ablaufumgebung für die Agenten sowie einige Basisdienste, wie einen Namens– und Ereignisdienst, zur Verfügung stellt. Auf Basis der in Abschnitt 5.3.2 gefällten Designentscheidungen bzgl. des

 Als Buch erhältlich im Verlag Dr. Hut, München, www.dr.hut-verlag.de (ISBN 3-934767-97-4)

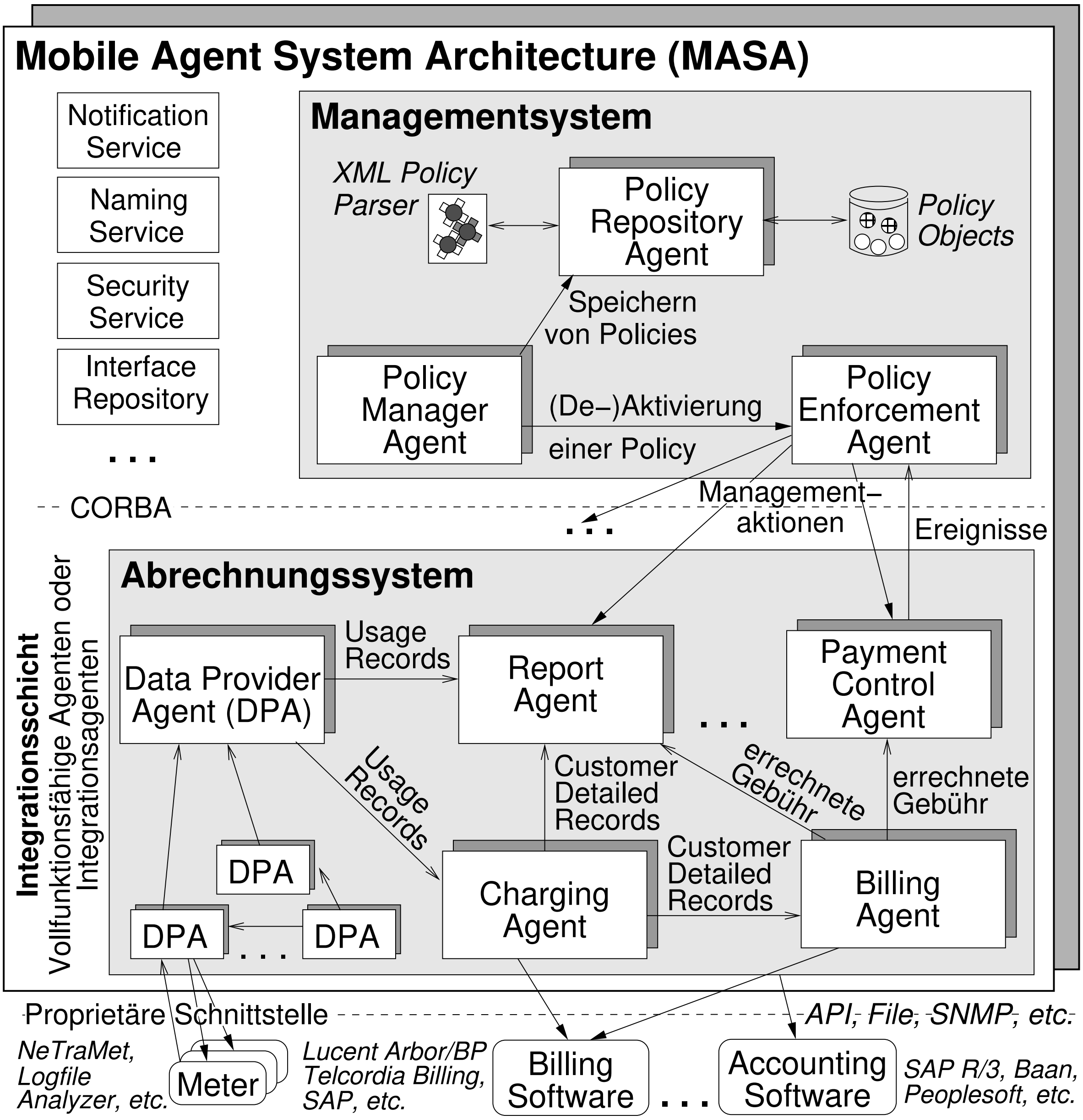

Abbildung 6.3: Überblick: Die Architektur der Implementierung

Managementsystems wird die Funktionalität der Managementanwendung in einen *Policy Manager Agent (PMA)*, einen *Policy Enforcement Agent (PEA)* und einen *Policy Repository Agent (PRA)* aufgeteilt.

Der Policy–Ersteller und Administrator interagiert mit dem PMA über eine entsprechende GUI, die Methoden zum Erzeugen, Löschen, Ändern sowie (De-)Aktivieren von Policies und Meta–Policies bereitstellt. Desweiteren bietet der PMA, mittels Interaktion mit dem PRA, die Möglichkeit, Policies, welche in der in Abschnitt 5.3.4 vorgestellten *Policy Description Lan-*

guage (PDL) vorliegen, persistent zu speichern. Die Grammatik der PDL ist in *XML–Schema* spezifiziert, so dass die Policies in Form von XML–Dokumenten vorliegen.

Der *Policy Repository Agent (PRA)* bietet Methoden zur persistenten Speicherung als auch zur Suche von Policies. Der PRA verwendet in der prototypischen Implementierung eigene Verwaltungsstrukturen, kann aber auch, falls vorhanden, auf LDAP–konforme Verzeichnisdienste zurückgreifen. Die Policies werden jeweils als XML–Dokumente gespeichert. Bei der tatsächlichen Instantiierung einer Policy, welches dem Kreieren eines entsprechenden Policy–Objekts entspricht, wird der *XML Policy–Parser* verwendet. Der Vorteil, der sich durch die Verwendung von XML ergibt, ist, dass bereits vorhandene, allgemeine XML–Parser verwendet werden können, um eine Policy zu dekodieren. Desweiteren können zusätzlich bereits vorhandene XML–Editoren adaptiert werden, um diese für die Erstellung von Policies zu verwenden.

Die Hauptaufgaben des *Policy Enforcement Agent (PEA)* ist die Verwaltung und Durchsetzung aktivierter Policies. Hierfür wird bei der Aktivierung einer Policy durch den PMA diese entsprechend ihrer Bestandteile zerlegt und in bestandteilspezifische Verwaltungsstrukturen eingefügt. Das heißt, dass die Ereignisse, Aktionen, Constraints, etc. jeweils in separaten Datenstrukturen verwaltet werden. Bei Auftreten eines Ereignisses sucht der PEA nach zutreffenden, registrierten Ereignissen. Ist dies der Fall, werden die zu dem Ereignis gehörenden Constraints, die durchaus zu mehreren Policies gehören können, ausgewertet. Bei den positiv evaluierten Constraints werden die dazugehörigen Aktionen, indem die entsprechenden Methoden am CORBA–Interface der Integrationsagenten aufgerufen werden, ausgeführt.

Die Integrationsschicht ist durch eine Vielzahl von MASA–Agenten realisiert. Den Designentscheidungen aus Abschnitt 5.3.2 folgend, entspricht jeder Integrationsagent einem identifizierten Akteur, die insgesamt im Zusammenspiel das zu managende Abrechnungssystem aus Prozesssicht repräsentieren. Wie bereits in Abschnitt 5.5.6 erläutert wurde, implementiert ein Integrationsagent eine, in diesem Fall CORBA–IDL–konforme, Schnittstelle, die der Funktionalität des korrespondierenden Akteurs entspricht. Diese IDL Schnittstellen basieren entweder auf bereits standardisierten Schnittstellen, wie sie in Kapitel 3 vorgestellt wurden, oder sie werden, wie es meist der Fall sein wird, durch das in Abschnitt 5.5.7 vorgestellte Verfahren auf Basis der in Abschnitt 4.3 spezifizierten Teilprozesse abgeleitet.

In den nachfolgenden Abschnitten werden die einzelnen Agenten näher erklärt. Die Agenten des Managementsystems, demnach also der PMA, PRA und PEA, wurden im Rahmen einer Diplomarbeit [Danc 03] entworfen und implementiert und werden jeweils in den Abschnitten 6.2.2–6.2.4 vorgestellt. Die Agenten der Integrationsschicht wurden unabhängig davon in einzelnen Fortgeschrittenenpraktika resp. Systementwicklungsprojekten implementiert und werden in Abschnitt 6.2.5 näher erklärt.

 Als Buch erhältlich im Verlag Dr. Hut, München, www.dr.hut-verlag.de (ISBN 3-934767-97-4)

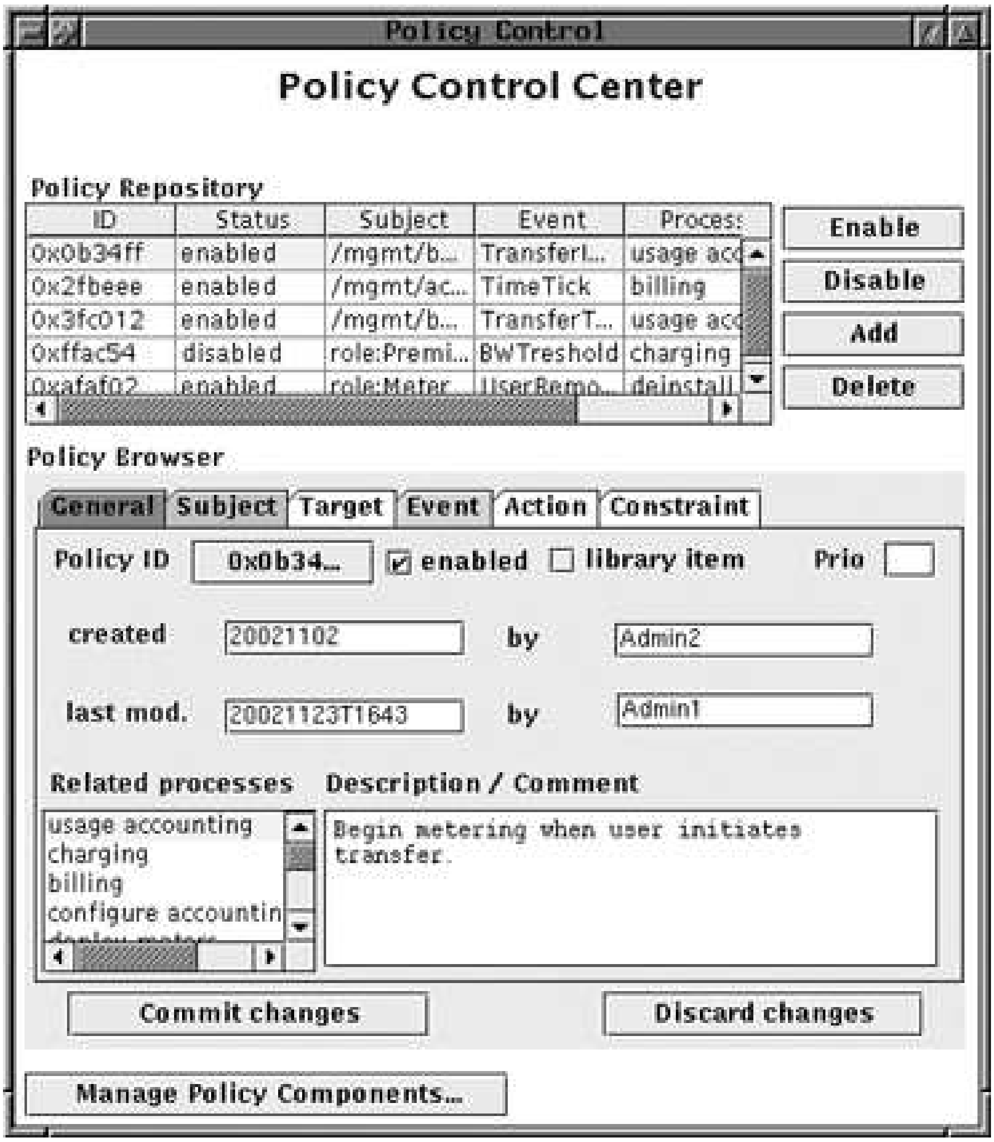

Abbildung 6.4: Überblick: Die GUI des Policy Manager Agents

6.2.2 Der Policy Manager Agent (PMA)

Der Policy Manager Agent implementiert die in Abschnitt 5.5.5 entworfene Schnittstelle der *PolicyManager*–Klasse. Die Hauptfunktionalität des PMAs liegt darin, sowohl dem Policy–Ersteller als auch dem Administrator ein Benutzerinterface zur Verwaltung von Policies anzubieten. Auch wenn die anderen beiden Agenten eigene Benutzerschnittstellen aufweisen, ist lediglich der PMA dazu konzipiert worden, direkt mit einem Anwender zu interagieren. Zur Erfüllung der Aufgaben kooperiert der PMA sowohl mit dem Policy Repository Agent (vgl. Abschnitt 6.2.3) zur persistenten Speicherung von Policies als auch mit dem Policy Enforcement Agent (vgl. Abschnitt 6.2.4) zur Durchsetzung von Policies über deren CORBA–Schnittstellen.

Abbildung 6.5: Subject–Ausschnitt der PMA–GUI

Neben der für alle MASA–Agenten üblichen Benutzerschnittstelle zum Starten, Anhalten und Migrieren des Agenten, ist in Abbildung 6.4 ein Überblick über das PMA–spezifische Benutzerinterface dargestellt. Unschwer ist zu erkennen, dass die GUI in drei Bereiche aufgeteilt ist:

- *Policy Repository*: Dieser Bereich zeigt in Form einer Tabelle die vom PMA verwalteten Policies an. Hierfür wird der dem PMA zugewiesene PRA abgefragt. Beim Auswählen einer konkreten Policy durch Anklicken, werden die Details im Policy Browser angezeigt.
- *Policy Browser*: Der Policy Browser zeigt die Details einer ausgewählten Policy. Die einzelnen Policy–Bestandteile sind durch Auswählen einer Karteikarte anzeigbar. In Abbildung 6.5 ist der Subject–spezifische Ausschnitt dargestellt, der beim Auswählen der ent-

sprechenden Karteikarte angezeigt wird.

- *Manage Policy Components*: Über dieses Bedienelement wird ein weiteres Fenster geöffnet, über das Policy–Bestandteile gruppiert werden können.

6.2.3 Der Policy Repository Agent (PRA)

Wie bereits erwähnt, liegt die Hauptfunktionalität des Policy Repository Agents in der persistenten Speicherung von erstellten Policies. Hierfür implementiert der PRA die in Abschnitt 5.5.5 entworfene Schnittstelle der *PolicyRepository*–Klasse. Desweiteren enthält der PRA als integrale Bestandteile einen *XML Policy–Parser*, der die Klasse *PolicyParser* implementiert, sowie die *PolicyFactory*–Klasse, mit der Policy–Objekte kreiert werden können.

Wird die *createPolicyObject()*–Methode der PRA–Schnittstelle vom PMA aufgerufen, so wird zunächst ein Syntaxbaum durch den XML Policy Parser aufgebaut. Der XML Policy–Parser verwendet hierzu den *Document Object Model (DOM)*–Parser des frei erhältlichen *Xerces*–Pakets[3], um auf Basis der PDL–Grammatik, die als XML–Schema vorliegt, den Syntaxbaum aufzubauen. Um letztendlich die Objektrepräsentation einer Policy durch Instantiierung der Klassen des PCIM–Packages zu kreieren, werden vom PRA die Knoten des Syntaxbaums durchlaufen und die entsprechenden Objekte durch Aufruf der Methoden der *PolicyFactory*–Schnittstelle instantiiert. Für das Traversieren durch den Syntaxbaum als auch für die Knotenmanipulation bietet die Java–API des Xerces–DOM–Parsers entsprechende Methoden an. Damit agiert der PRA in dieser Situation als Interpreter der Policy. In Abbildung 6.6 sind die beschriebenen 2 Phasen bildlich dargestellt.

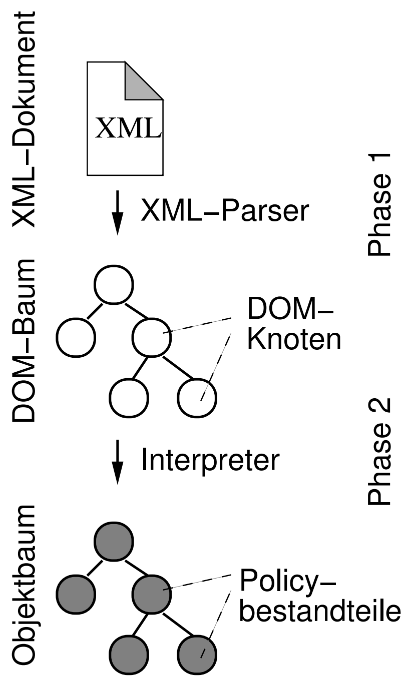

Abbildung 6.6: Übersetzung einer XML–Policy (aus [Danc 03])

Demnach wird die in Abschnitt 5.3.4 entwickelte Grammatik auf Basis von XML–Schema repräsentiert (siehe Anhang A). Der Vorteil von XML–Schema im Gegensatz zu einer DTD liegt darin, dass eine in XML–Schema festgelegte Grammatik selber ein wohlgeformtes XML–Dokument ist und somit dieselben Werkzeuge (wie z.B. Parser) zur Abarbeitung eines XML–Schema–Dokuments als auch eines einfachen XML–Dokuments verwendet werden können. Desweiteren ist es mit XML–Schema möglich, Datentypen und Wertebereiche festzulegen, welches mit einer DTD nicht möglich ist. Auf eine tiefergehende Betrachtung von XML und XML–Schema wird an dieser Stelle verzichtet und auf die Spezifikationen [XMLS-0, XMLS-1, XMLS-2] verwiesen.

[3] `http://xml.apache.org/xerces2-j/index.html`

Der PRA besitzt zwar wie jeder MASA–Agent eine GUI, jedoch bietet diese nur eine Anzeige von den verwalteten Policies an. Die Benutzerschnittstelle ist mit dem *Policy Repository*–Bereich des PMA nahezu identisch.

6.2.4 Der Policy Enforcement Agent (PEA)

Der Policy Enforcement Agent implementiert die in Abschnitt 5.5.5 entworfene Klasse *PolicyEnforcer* und bietet damit insbesondere eine Schnittstelle zum Registrieren von aktivierten Policies an. Diese wird in aller Regel ausschließlich von einem PMA genutzt. Grundsätzlich wartet der PEA auf das Eintreten eines registrierten Ereignisses, um daraufhin die dazugehörigen Constraints zu evaluieren und im positiven Fall die spezifizierten Managementaktionen auszuführen. Als zentraler Punkt der PEA–Implementierung wird nachfolgend der genaue Ablauf der Durchsetzung einer Policy beschrieben, welcher auch in Abbildung 6.7 als UML Sequenzdiagramm dargestellt ist.

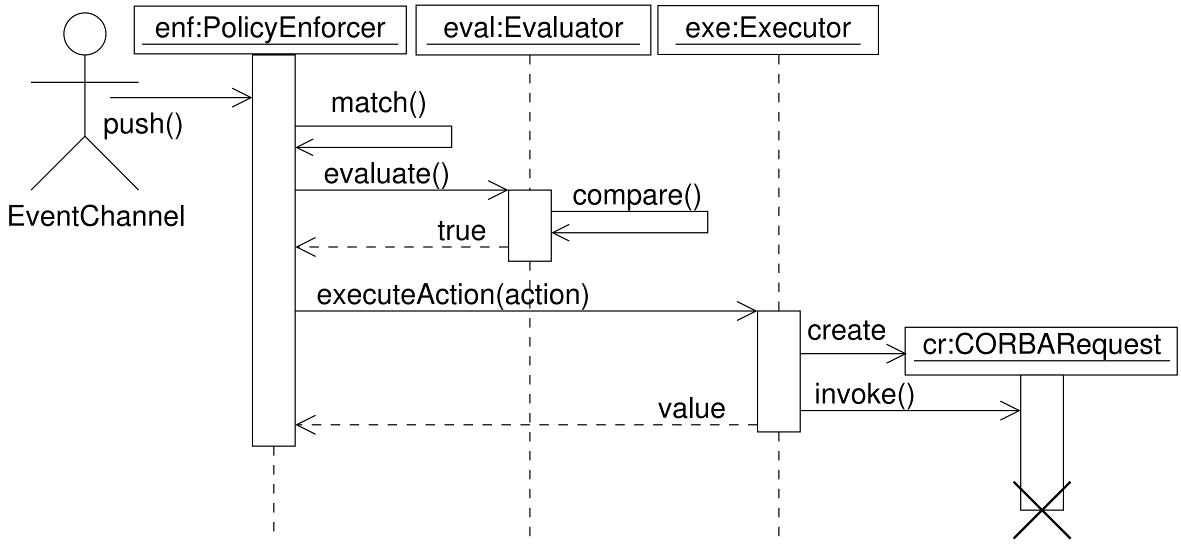

Abbildung 6.7: PEA: Durchsetzung einer Policy als UML Sequenzdiagramm (nach [Danc 03])

Für das Versenden und Empfangen von Ereignissen wird eine Orbacus–Implementierung des *CORBA Event Services* [OMG 01-03-01] verwendet. Der Event Service sieht sowohl ein Push–, Pull– als auch hybrides Modell zur Weiterleitung von Ereignissen vor. Im vorliegenden Fall wurde das Push–Modell gewählt, in welchem die Integrationsagenten die *Event–Push–Supplier* sind und der PEA der *Event–Push–Consumer* ist. Als „Transport–Medium" für die Ereignisse wird ein well–known Event–Channel im Agentensystem (der sog. *AgentInitChannel*) verwendet, in dem alle registrierten Event–Consumer, wie z.B. der PEA, die von den registrierten Event–Suppliern, wie z.B. die Integrationsagenten, Ereignisse empfangen. Bei Auftreten eines Ereignisses prüft der PEA zunächst, ob es ein evaluationsauslösendes Ereignis ist und demnach,

 Als Buch erhältlich im Verlag Dr. Hut, München, www.dr.hut-verlag.de (ISBN 3-934767-97-4)

ob der Ereignisname registriert ist. Sofern dies der Fall ist, werden die dazugehörigen Constraints evaluiert. Sofern die Evaluierung eines Constraints wahr ergibt, werden die dazugehörigen Aktionen ausgeführt. Zur Ausführung der Aktionen wird das *CORBA Dynamic Invocation Interface (DII)* [OMG 99-07-11] verwendet, welches das dynamische Aufrufen von Methoden an einer CORBA Schnittstelle zur Laufzeit unterstützt, die zum Zeitpunkt der Kompilierung des Programmcodes nicht bekannt sind[4]. Hierzu wird aus der in der Policy enthaltenden Beschreibung der Methode und des Zielobjekts zunächst die genaue Methodensignatur aus der *Interface Repository* [OMG 97-10-12] ermittelt. Die Interface Repository ist ein zum DII gehörender Dienst, welcher alle IDL–Interface–Beschreibungen der in einem CORBA–System vorhandenen CORBA–Objekte enthält[5]. Dadurch ist es möglich, einen CORBA–Aufruf zusammenzusetzen und diesen schließlich auszuführen. Bei Fehlschlag des Aufrufs wird der *onError*{}–Block der Aktionsdeklaration ausgeführt.

Auch der PEA hat eine agentenspezifische GUI, die allerdings nur die Auflistung der diesem PEA zugewiesenen, aktivierten Policies enthält.

6.2.5 Die Agenten der Integrationsschicht

In Abschnitt 5.5.6 wurden bereits generische Schnittstellen für die Agenten der Integrationsschicht entworfen, die durch die abstrakte Klasse *IntegrationAgent* zusammengeführt wurden. Betrachtet man nun diejenigen Agenten, die sich insbesondere mit der Datenmessung und Datenerfassung beschäftigen, so lässt sich feststellen, dass die Agenten zueinander als auch bezüglich der diesen Agenten zugewiesenen Subjekte immer in einer Erzeuger–Verbraucher–Beziehung stehen. Demnach ist diesen Agenten gemeinsam, dass Daten eingesammelt werden, diese Daten weiterverarbeitet und anschließend weitergegeben werden. Die Unterschiede der einzelnen Agenten liegen v.a. im Einsammel– bzw. Weitergabemodell und in der agenten–eigenen Weiterverarbeitung der eingesammelten Daten. Somit liegt der Gedanke nahe, einen generischen Agenten zu implementieren, der die von allen spezifischen Agenten benötigte Grundfunktionalität bereits realisiert. Bei dem *DataProvider–Agenten (DPA)* handelt es sich um exakt einen derartigen Agenten, der selber zwar nicht instantiierbar ist, aber als Superklasse resp. Superagent allen anderen Agenten in der Integrationsschicht, die mit der Aufgabe der Datenmessung resp. Datenerfassung betraut sind, dient. Der DPA erbt von der abstrakten Klasse *IntegrationAgent* und versieht all diejenigen Methoden mit einer Implementierung, die von Subagenten ohne weiteres übernommen werden können. Damit senkt sich der Aufwand zur Implementierung derartiger Integrationsagenten erheblich.

Folgende Bestandteile sind im DataProvider–Agenten bereits implementiert und können somit von Subagenten ohne weitere Anpassung übernommen werden:

[4] Normalerweise werden hierfür die zu einer IDL–Schnittstelle generierten *Client Stubs* verwendet, welche den entfernten Methodenaufruf wie einen lokalen erscheinen lassen.

[5] Allerdings müssen die IDL–Beschreibungen vorher bei der Interface Repository registriert werden.

- *Generisches Benutzerschnittstelle*
  Die GUI sieht eine tabellenartige Darstellung der dem DPA zugewiesenen Datenquellen vor. Eine Zeile entspricht einer Datenquelle, wobei einzelne Spalten den Werten der Konfigurationseinträge entsprechen. Desweiteren ist ebenfalls ein Fenster vorgesehen, in dem die durch die *execQuery()*–Methode des *IntegrationUsage*–Interfaces zurückgegebenen Daten dargestellt werden. Damit können die DP–Agenten nicht nur über Policies konfiguriert werden, sondern auch über die Benutzerschnittstelle.

- *RealSubjectManagement–Schnittstelle*
  Die *RealSubjectManagement*–Schnittstelle sieht Methoden zur Verwaltung der dem DPA zugewiesenen Subjects vor. Dies wird durch Bereitstellung geeigneter Datenstrukturen, die zur Verwaltung der Konfigurationseinträge verwendet werden, unterstützt.

- *IntegrationManagement–Schnittstelle*
  Die Implementierung der generischen *IntegrationManagement*–Schnittstelle sieht das automatische Mitprotokollieren der Methodenaufrufe der *RealSubjectManagement*– und *IntegrationUsage*–Schnittstelle vor. Das Protokoll wird durch Aufruf der Methode *getActivityRecord()* zurückgeliefert.

- *Gerüst für die IntegrationUsage–Schnittstelle*
  Das Methodengerüst der *IntegrationUsage*–Schnittstelle ist bereits mit Standard–Debug–Meldungen versehen, jedoch fehlt jegliche Implementierung der Funktionalität.

Demnach müssen folgende Bestandteile in einem Subagenten des Typs *DataProvider* implementiert werden:

- *Methoden der RealSubjectManagement–Schnittstelle*
  Die Konfigurationsmethoden dieser Schnittstelle müssen gemäß des Typs der vom DPA verwalteten Datenquellen präzisiert und implementiert werden. Hierbei ist auf das Kommunikationsmodell zwischen DPA und Datenquelle zu achten (Push/Pull/Eventbasiert) sowie auf die spezifischen Initialisierungs– und Konfigurationsvorgänge. Es kann notwendig sein, diese Schnittstelle mit Subject–spezifischen Managementmethoden zu erweitern (siehe Anmerkungen hierzu in Abschnitt 5.5.7).

- *Methoden der IntegrationUsage–Schnittstelle*
  Die Methode *execQuery()* dieser Schnittstelle muss gemäß des Typs der vom DPA verwalteten Datenquellen implementiert werden.

- *Datenstrukturen*
  Aufgrund der Heterogenität der Datenquellen müssen Datenstrukturen entsprechend ihrer Beschaffenheit implementiert werden. So muss beispielsweise Datenverlust der eingesammelten Daten ausgeschlossen werden, falls die Datenquelle nur flüchtigen Speicher bereitstellt etc. Dies kann beispielsweise durch ein eigen realisiertes Zwischenspeichern in Form einer einfachen Datenhaltung gelöst werden, die je nach zu erwartender Datenmenge in Form von Arrays/Vektoren oder auch Dateien realisiert ist.

 Als Buch erhältlich im Verlag Dr. Hut, München, www.dr.hut-verlag.de (ISBN 3-934767-97-4)

- *Verfeinerung der Aufzeichnungsqualität der Aktivitätsprotokolle*
  Die generische Implementierung der Protokollführung der Aktivitäten sieht lediglich das Aufzeichnen der Methodenaufrufe vor. Unter Umständen ist es aber sinnvoll, diese Aufzeichnung auf weitere, feingranularere Aktivitäten innerhalb der einzelnen Methoden zu erweitern. Hierzu sind innerhalb des Programmcodes die entsprechenden Aufrufe der internen Aufzeichnungsmethoden zu platzieren.

Im *DataProvider* ist also der generische Teil der Funktionalitätsimplementierung zu finden, der für alle Agenten gleichermaßen notwendig und wiederverwendbar ist, d.h. Gerüste für Schnittstellen, die Benutzeroberfläche, entsprechende Datentypen, -strukturen und Methoden. Im Besonderen ist die Tatsache hervorzuheben, dass es sich hierbei dennoch um eine abstrakte Klasse handelt, da in jedem Falle die eigentlichen Bestandteile zur Datensammlung bzw. -erfassung fehlen.

Demgegenüber muss die Schnittstelle im Subagenten entsprechend der spezifischen Anforderungen implementiert werden, d.h. das vorhandene Gerüst des *DataProvider* muß mit entsprechenden eigenen Methoden überschrieben werden.

Um die Entwicklung von entsprechenden Subagenten zu erleichtern, wurde die MASA Entwicklungsumgebung um *DataProvider*–spezifische Optionen erweitert: durch Angabe eines Namens für den Subagenten werden automatisch Packages, entsprechend Programmgerüste, etc. in Abhängigkeit von dem gewünschten Namen erzeugt. Dies kann auch von einer Policy aus aufgerufen werden (z.B. Implementierungspolicy).

Im Folgenden werden einige der bereits erstellten Subagenten vorgestellt.

Der CpuDataProvider-Agent

Funktionalität Der *CpuDataProvider (CDP)* stellt Daten über die CPU–Auslastung eines Prozesses zur Verfügung. Hierbei dient als Datenquelle ein auf dem zu überwachenden System laufender Prozess (*meterd*). Dieser sammelt und puffert die eigentliche Information, wobei die Steuerung dieses Überwachungsprozesses dem zugeordneten *CpuDataProvider* obliegt. Auf dessen Anfrage werden die gewünschten Informationen per Datagrammkommunikation übermittelt.

Implementierung der *RealSubjectManagement*–Schnittstelle Wie alle Subagenten des Typs *DataProvider* resp. *IntegrationAgent* differenziert sich der *CpuDataProvider* primär durch die spezifischen Konfigurationsparameter der *addSubjectConfig()*–Methode der *RealSubjectManagement*–Schnittstelle, welche vom zugewiesenen Datenquellen–Typ abhängig sind, in diesem Fall also vom Überwachungsprozess *meterd*. In Abbildung 6.8 ist das generische Frontpanel des CpuDataProviders dargestellt. Der CDP erweitert keine der generischen Schnittstellen um weitere Methoden.

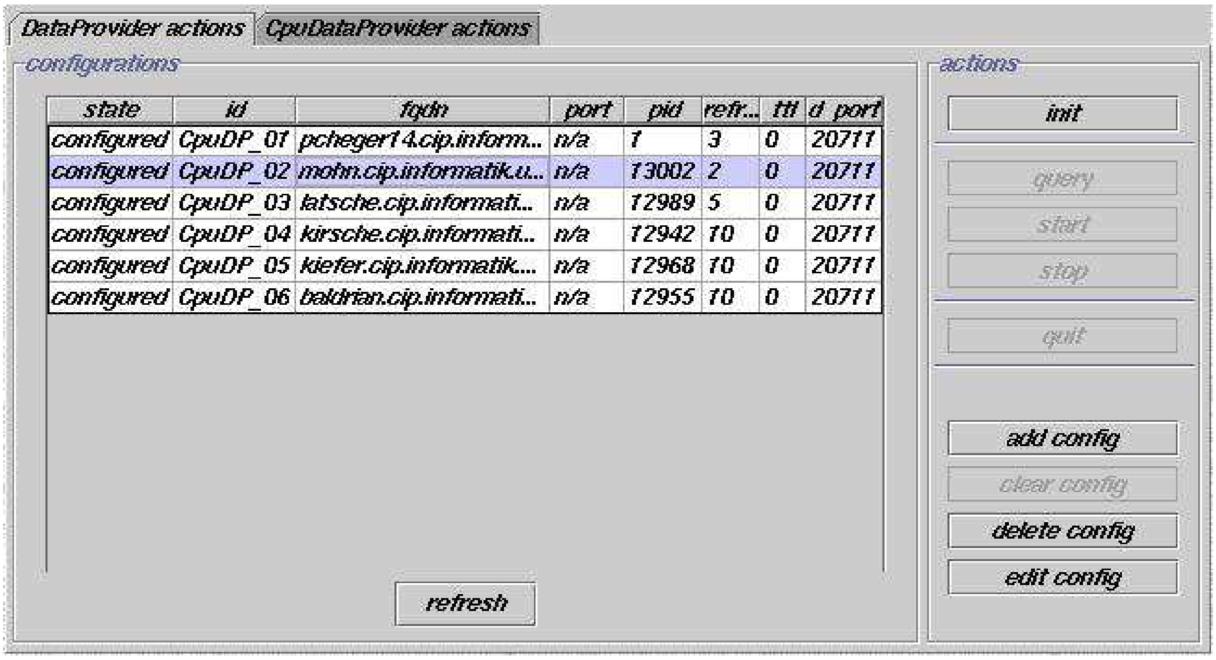

Abbildung 6.8: Frontpanel des *CpuDataProvider*

Bei Initialisierung eines neuen zu überwachenden Prozesses wird ein separater Kindprozess des *meterd* gestartet. Die folgenden Parameter bilden einen vollständigen Konfigurationseintrag (wird als String–Array übergeben):

- **fully qualified domain name (fqdn)**: Der *fqdn* identifiziert das zu überwachende System.
- **process id (pid)**: In diesem Wert ist die Prozeßnummer des zu überwachenden Prozesses abgelegt.
- **refresh**:Dieser Parameter spezifiziert das Messintervall (Angabe in Sekunden), in dem Daten über den durch *pid* identifizierten Prozess gesammelt werden.
- **select**: Mittels des *select*-Parameters wird das relevante Datum selektiert, wobei 0 die idle-Zeit selektiert, 1 die nice-Zeit, etc.
- **d_port**: Diese Portnummer dient der Kommunikation mit dem *meterd*–Vater–Prozess, der die Initialisierung und das Starten eines neuen Meters steuert. Diese ist prinzipiell systemabhängig frei bei Start des *meterd* wählbar, Defaulteinstellung ist 20711.

Anhand dieser Konfigurationsparameter kann der *CpuDataProvider* mit einem zugewiesenem *meterd* eindeutig per UDP kommunizieren und Daten austauschen. Das in Abbildung 6.8 dargestellte Frontpanel des CPDs visualisiert neben dem momentanen Zustand der Datenquelle und der *SubjectID* den vollständigen Konfigurationseintrag. Die übrigen in Abschnitt 5.5.6 vorgestellten Methoden der *RealSubjectManagement*–Schnittstelle haben im Kontext des *CpuDataProviders* die folgende Semantik:

initSubject(): Die Initialisierung einer Datenquelle wird mit den vorher zur Datenquelle gehörenden, konfigurierten Parametern *pid*, *refresh* und *select* durchgeführt. Zusätzlich wird eine Portnummer generiert, die für den Zugriff auf den jeweiligen *meter*–Kindprozess verwendet wird. Hierzu wird im Bereich ab 25000 ein freier Port gesucht. Ein zuvor auf der zu überwachenden Maschine gestarteter Vaterprozess übernimmt die Funktion eines Spawning Process, d.h. er generiert auf einen *init* hin einen Kindprozess, welcher die eigentliche Funktion der Datensammlung erfüllt.

startSubject(): Die Datensammlung und damit die Messung der von dem Prozess *pid* verursachte CPU–Auslastung wird gestartet.

stopSubject(): Dies bewirkt das Anhalten der Datensammlung eines bereits gestarteten Meters. Konfiguration und Vaterprozess bleiben für eine spätere Wiederaufnahme der Aufzeichnung erhalten.

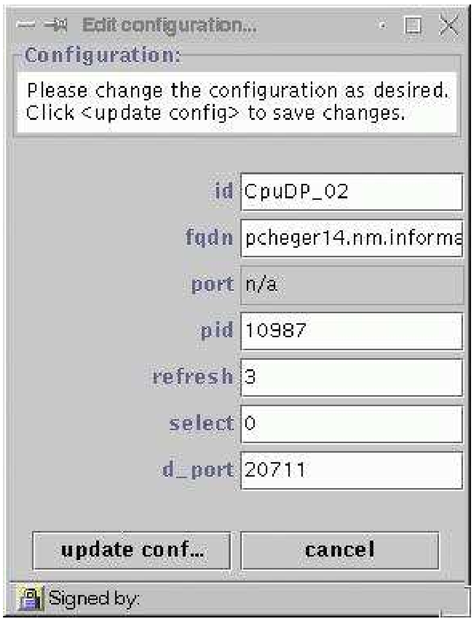

Abbildung 6.9: EditConfig–Dialog des *CpuDataProvider*

quitSubject(): Diese Methode terminiert den für die Datensammlung beauftragten Kindprozess auf dem jeweiligen System. Hierbei wird auch die temporär angelegte Aufzeichnungsdatei gelöscht, welche die bisher gesammelten Daten enthält.

In Abbildung 6.9 ist das `CpuDataProvider`–Dialogfenster zum Editieren der Konfiguration einer Datenquelle dargestellt.

Implementierung der *IntegrationUsage*–Schnittstelle Die *execQuery()*–Methode wurde recht einfach realisiert. Diese liefert für einen gegebenen Zeitraum die angefallenen Messdaten zurück. Hierfür wird immer der jeweilige *meter* abgefragt, sofern die Daten nicht bereits angefragt wurden und im Cache vorliegen. Als Parameter für die Suchanfrage werden die folgenden erwartet (werden als *Comma Separated List (CSL)* übergeben):

- **SubjectID**: Der Identifikator der Datenquelle (*meter*), von der die Messdaten einzusammeln sind.
- **StartTimeStamp**: Zeitstempel des frühesten Zeitpunkts, zu dem Messdaten von Interesse sind.
- **EndTimeStamp**: Zeitstempel des spätesten Zeitpunkts, zu dem Messdaten von Interesse sind.

Es wird eine Liste von Messdaten zurückgeliefert. Die Liste enthält für jede durchgeführte Messung einen Zeitstempel in Kombination mit dem gemessenen Wert.

Der SnmpDataProvider-Agent

Funktionalität Während der im vorhergehendem Abschnitt vorgestellte *CpuDataProvider* über eine proprietäre Schnittstelle einheitliche Informationen erfasst hat, implementiert der *SnmpDataProvider* den Zugriff auf Internet MIBs per SNMP. Damit können prinzipiell alle per SNMP zugänglichen Daten erfasst werden, wobei das Hauptaugenmerk hierbei auf Daten in numerischer Form liegt, welche insbesondere für die Abrechnung, im Gegensatz zu beispielsweise Informationen mit eher administrativen Charakter, relevant sind.

Es existieren zahlreiche MIBs, die sowohl system– als auch netzorientierte Parameter wie Disk– und Druckerquota, Anzahl übertragener Pakete/Bytes, etc. liefern. Damit bietet der *SnmpDataProvider* eine breite Basis zur Erfassung von abrechnungsrelevanten Daten.

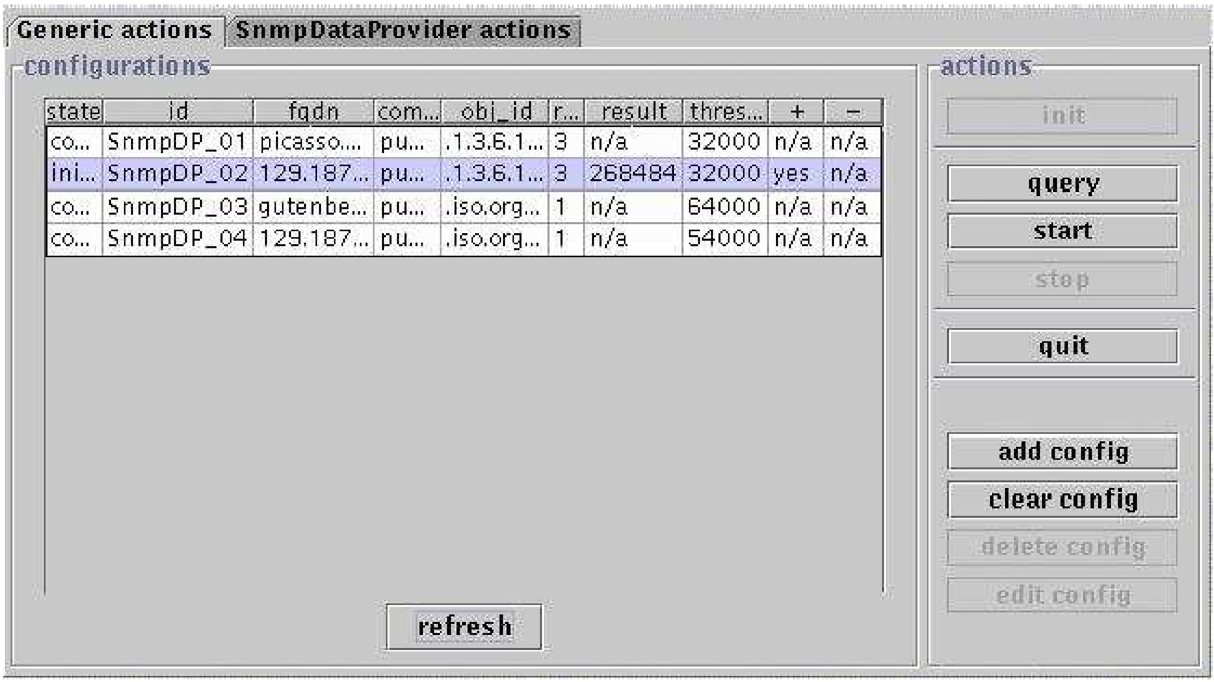

Abbildung 6.10: Frontpanel des *SnmpDataProvider*

Implementierung der *RealSubjectManagement*–Schnittstelle Die Konfiguration einer Datenquelle für den *SnmpDataProvider* enthält die folgenden spezifischen Parameter:

- **fully qualified domain name** (fqdn): Der *fqdn* des zu überwachenden Systems.
- **community string**: Der *read*–Community–String für die angegebene Komponente.
- **object id**: Der sogenannte Object Identifier der Managementvariable.
- **refresh**: Hiermit wird das Polling–Intervall für die regelmäßige Anfrage festgelegt.

- **threshold**: Falls ein numerischer Wert durch die Object ID bestimmt wird, kann mit diesem Parameter ein Schwellwert angegeben werden. In Abhängigkeit des Vergleichs zwischen *threshold* und dem zuletzt abgefragten Wert (in Abbildung 6.10 als *result* geführt) wird entweder das intern geführte + oder - Flag gesetzt.
- **event**: Je nachdem, ob ein + oder - gesetzt ist, wird bei Über– bzw. Unterschreiten des *threshold* ein Event generiert und an einen well–known Event Channel geschickt.

Mit Initialisierung und Start der Datenquelle werden in den spezifizierten Zeitintervallen die festgelegten Managementvariablen abgefragt und in internen Datenstrukturen gespeichert. Bei Aufruf von *execQuery()* werden die Daten bezogen auf eine Datenquelle in der gleichen Art und Weise wie beim CPDs zurückgeliefert.

Weitere realisierte Integrationsagenten

Neben den hier vorgestellten Agenten wurden noch folgende Integrationsagenten implementiert, die im Nachfolgenden nur genannt und kurz beschrieben werden:

- *TrafficDataProvider (TDP)*
  Der TrafficDataProvider [Saca 02] bietet eine Schnittstelle für die Konfiguration und das Management von Komponenten, welche Netzdatenverkehr aufzeichnen. Insbesondere integriert der TDP Messkomponenten auf Basis der RTFM–Messarchitektur (vgl. Abschnitt 3.2.5). Im vorliegenden Fall wurde der TDP mit *NeTraMet* getestet.
- *ScriptingAgent*
  Beim ScriptingAgent [Merk 02] handelt es sich um einen Multi–Hop–Agenten, der auf seinen konfigurierten Stationen konfigurierte Skripte ausführt. Die Skripte führen meist eine Aggregation von Log–Dateien durch, wie sie beispielsweise durch Web–Server angelegt werden. Die Skripte werden jeweils durch einen im Agenten integrierten Interpreter (im vorliegenden Fall: Python) ausgeführt, so dass die Ausführung vollkommen plattformunabhängig ist. Damit kann vor Ort eine Aggregation der Daten vorgenommen werden, so dass das Übertragen von großen Log–Dateien nicht notwendig ist.
- *AggregatingAgent*
  Der AggregatingAgent interagiert nur mit Agenten der Integrationsschicht. Hierbei werden die Ergebnisse der konfigurierten *execQuery()*–Anfrage per vordefinierter Funktion miteinander kombiniert. Beispielsweise stehen mehrere einfache Funktionen, wie Summe und Durchschnitt, zur Verfügung. Damit erfüllt der AggregatingAgent die Aufgabe eines aggregierenden Kollektors.

6.3 Gesamtbewertung der entwickelten Lösung

Um die in dieser Arbeit entwickelte Lösung zu bewerten, wird der Anforderungskatalog aus Abschnitt 2.3.5 angewendet. Insgesamt bilden fünf Aspekte die charakteristischen Merkmale der entwickelten Lösung, die sie von anderen Ansätzen unterscheiden und welche damit ausschlaggebend für die Erfüllung der in Kapitel 2 identifizierten Anforderungen sind. In den einzelnen Kapiteln dieser Arbeit wurden bei Einführung eines neuen Aspekts der Lösung bereits selektiv die dadurch erfüllten Anforderungen genannt. Nachfolgend werden diese zusammengefasst:

- *Prozessorientierung*
  Die prozessorientierte Sicht auf den Abrechnungsvorgang als auch auf den Abrechnungsmanagementvorgang sichert einerseits, dass der vollständige Dienstlebenszyklus aus Abrechnungssicht betrachtet wird (ALL 1), als auch andererseits, dass eine klare Trennung zwischen dem Abrechnungsvorgang und dem Abrechnungsmanagement besteht (ALL 2). Zudem wird dem Manager durch die Prozessorientierung eine integrative, einheitliche Sicht auf die Managementaufgabe präsentiert (MGT 1).

- *Anwendung policy–basierter Konzepte*
  Da Domänen innerhalb von Policies ausgedrückt werden können, um das *Subject* resp. *Target* näher zu spezifizieren, wird damit die Grundlage geschaffen, um eine domänenübergreifende Kooperation von sowohl abrechnungs– als auch managementrealisierenden Komponenten zu schaffen (IMP 3, MGT 6). Das Domänen–,Gruppen– und Rollenkonzept sichert zudem die Mandantenfähigkeit des Managementsystems (MGT 11). Die Policy–Sprache als die einzige „Schnittstelle“ zwischen Policy–Ersteller und Managementanwendung stellt zudem eine einheitliche, integrative Sicht auf das zu managende System (MGT 1) zur Verfügung. Zudem erreicht man durch die Anwendung policy–basierter Konzepte ein flexibles und automatisiertes Management, welches insbesondere die Durchführung von Change–Managementaktivitäten unterstützt (MGT 2, MGT 3, MGT 4). Da jederzeit ohne Einschränkung des „normalen“ Betriebs neue Policies spezifiziert und aktiviert werden können, ist das Management an sich sehr flexibel in seiner Erweiterbarkeit (MGT 7). Da die Anzahl der durch eine Policy gesteuerten Entitäten nicht beschränkt ist, skaliert der Ansatz auch bei einer großen Menge von Managementobjekten (MGT 8).

- *Managementarchitektur bestehend aus einer Integrationsschicht*
  Die Integrationsschicht setzt die klare Trennung zwischen der Managementanwendung und den zu managenden Entitäten tatsächlich um (ALL 2). Zudem werden mit den Agenten der Integrationsschicht, welche die tatsächlichen Akteure des Abrechnungsvorgangs repräsentieren, einheitliche und, sofern verfügbar, sogar standardisierte Nutzungs– und Managementschnittstellen implementiert (IMP 1, IMP 2). Diese verschatten damit auch die Heterogenität einer zusammengestellten Abrechnungslösung (MGT 5). Ebenso sichert die Integrationsschicht die grundsätzliche Erweiterbarkeit gegenüber neuen zu managen-

 Als Buch erhältlich im Verlag Dr. Hut, München, www.dr.hut-verlag.de (ISBN 3-934767-97-4)

den Entitäten (IMP 9). Da sich durch die Schichtenarchitektur Änderungen an den abrechnungsrealisierenden Komponenten nicht direkt auf bereits existierende und aktivierte Policies auswirken, werden Änderungsaktivitäten dieser Art sehr gut unterstützt (MGT 4).

- *MASA als Entwicklungs– und Laufzeitumgebung*
  Durch den Einsatz von Java/CORBA innerhalb von MASA ist die darauf realisierte Lösung mittels der jeweilig dazugehörende dynamischen Konzepte wie Java Classloader und CORBA DII/DSI sehr flexibel erweiterbar (MGT 7) und grundsätzlich offen (MGT 13). Mit den realisierten Zusatzdiensten (z.B. MAFFinder [Gilg 02]) ist eine domänenübergreifende Kooperation der mit MASA realisierten Agenten möglich (IMP 3, MGT 6). Da MASA–Agenten als mobile Agenten implementiert werden können, wird durch das Konzept der Agentenmigration die Ausfallsicherheit erhöht (IMP 8, MGT 9). Der als integraler Bestandteil von MASA realisierte Security Service [Reis 01, Roel 99] erfüllt die Anforderung, dass die Komponenten sicher miteinander interagieren (IMP 4).

- *Implementierung der entworfenen Schnittstellen*
  Bei der prototypischen Implementierung der entworfenen Managementlösung wurde insbesondere bei den Integrationsagenten darauf geachtet, dass sowohl die Mandantenfähigkeit (IMP 5) umgesetzt wird als auch, dass alle durchgeführten Aktionen im Nachhinein nachvollziehbar sind (IMP 6, MGT 12). Um die Güte der Interaktion zwischen den Agenten zu erhöhen (IMP 7), wurde darauf geachtet, dass sowohl mehrere Interaktionsmodelle unterstützt werden als auch, dass nur die tatsächlich notwendigen Daten übertragen werden.

In Tabelle 6.1 ist die vollständige Evaluierung der entwickelten Lösung durch Anwendung des Anforderungskatalogs übersichtlich zusammengefasst. Wie daraus ersichtlich ist, wird die Mehrzahl der Anforderungen bereits durch die eingesetzten Konzepte (Prozesse, Policies, Integrationsschicht) erfüllt, so dass grundsätzlich auch der Einsatz einer anderen Entwicklungs– und Laufzeitumgebung abgesehen von MASA denkbar ist. In diesem Fall ist aber sowohl bei der Wahl der Plattform als auch bei der Implementierung der Bestandteile der Lösung auf die Erfüllung derjenigen Anforderungen zu achten, die *alleinig* durch MASA resp. der tatsächlichen Implementierung erfüllt werden.

	Prozess–orientierung	Policy–Konzepte	Integrations–schicht	MASA	Implemen–tierung
ALL 1	✓				
ALL 2	✓		✓		
IMP 1			✓		✓
IMP 2			✓		✓
IMP 3		✓		✓	
IMP 4				✓	
IMP 5					✓
IMP 6					✓
IMP 7				✓	✓
IMP 8				✓	✓
IMP 9			✓		
MGT 1	✓	✓			
MGT 2		✓			
MGT 3		✓			
MGT 4		✓	✓		
MGT 5			✓		
MGT 6		✓		✓	
MGT 7		✓		✓	
MGT 8		✓	✓		
MGT 9				✓	
MGT 10				✓	
MGT 11		✓			
MGT 12					✓
MGT 13				✓	

Tabelle 6.1: Gesamtbewertung der entwickelten Lösung

 Als Buch erhältlich im Verlag Dr. Hut, München, www.dr.hut-verlag.de (ISBN 3-934767-97-4)

6.4 Zusammenfassung

In diesem Kapitel wurde die prototypische Implementierung des in Kapitel 5 entwickelten Lösungsansatzes vorgestellt. Als Laufzeit– und Entwicklungsumgebung wurde die *Mobile Agent System Architecture (MASA)* gewählt. Die eigentliche Managementanwendung wurde gemäß des objektorientierten Entwurfs in Abschnitt 5.5 in drei separate Agenten (PMA, PEA, PRA) aufgeteilt, die in Kooperation die notwendige Funktionalität erbringen. Die Managementanwendung an sich ist nicht auf das dienstorientierte Abrechnungsmanagement beschränkt und kann somit auch in anderen Managementfunktionsbereichen für ein prozessorientiertes IT Management eingesetzt werden. Anschließend wurden einige umgesetzte Integrationsagenten vorgestellt. Aufgrund der Zusammenführung der Implementierung von generischen Teilen von Datenerfassungs– und Messagenten in den sog. DataProvider–Agenten wurde die Implementierung von neuen Agenten gleichen Typs wesentlich vereinfacht.

Zum Schluss erfolgte schließlich die Evaluierung der entwickelten Lösung durch Anwendung des Anforderungskatalogs. Jedem in dieser Arbeit eingesetzten Konzept wurde hierbei dediziert die dadurch erfüllten Anforderungen zugeordnet.

 Als Buch erhältlich im Verlag Dr. Hut, München, www.dr.hut-verlag.de (ISBN 3-934767-97-4)

Kapitel 7

Zusammenfassung und Ausblick

Ausgehend von der Beobachtung, dass gegenwärtig verfügbare Produkte als auch Managementansätze nicht in der Lage sind, insbesondere die hohe Änderungsdynamik im dienstorientierten Abrechnungsmanagement zu unterstützen, wurde in der vorliegenden Arbeit ein prozessorientierter, auf policy–basierten Konzepten beruhender Ansatz für ein integriertes, dienstorientiertes Abrechnungsmanagement entwickelt. Der hierbei entwickelte grundlegende Lösungsansatz ist derart allgemein anwendbar, dass er nicht auf das Abrechnungsmanagement beschränkt ist, sondern in gleicher Art und Weise in anderen Managementfunktionsbereichen (Fehler–, Leistungs–, Sicherheits– und Konfigurationsmanagement) einsetzbar ist, um ein prozessorientiertes IT Management durchzusetzen.

Zusammenfassung und Ergebnisse dieser Arbeit

Da sich noch keine einheitliche Terminologie im dienstorientierten Abrechnungsmanagement durchgesetzt hat, wurden zunächst auf Basis des MNM Dienstmodells [GHK+ 01] die grundlegenden Begriffe in diesem Bereich definiert und erläutert. Ausgehend von einer detaillierten Szenarioanalyse (Outsourcing von Individualdiensten im Großkundenbereich), wurden die besonderen Charakteristika des daraus resultierenden Abrechnungsmanagements beschrieben. Es stellte sich heraus, dass die Abrechnung in gleicher Weise wie die Dienstrealisierung immer individuell den Kundenanforderungen entsprechend zusammengestellt wird. Da die dafür eingesetzten HW/SW–Komponenten über keine standardisierten Nutzungs– und Managementschnittstellen verfügen, ist ebenso das Abrechnungsmanagement immer individuell an die Realisierung des Abrechnungsvorgangs angepasst. Hierbei stellte sich ebenfalls heraus, dass das Management der an der Abrechnung beteiligten Komponenten meist isoliert und nicht integriert verwirklicht wird. Die aus den langen Vertragslaufzeiten der Dienstvereinbarung resultierende hohe Änderungsdynamik, welche als die Regel und nicht als die Ausnahme in den fokussierten Szenarios zu betrachten ist, hat zur Folge, dass bei jeder Änderung der Abrechungsimplementierung auch das dazugehörige Abrechnungsmanagement adaptiert werden muss. Dies ist bisher meist manuell durch die Administratoren durchgeführt worden, welches sich als sehr fehleranfällig und aufwändig herausstellte. Diese Szenariocharakteristika dienten schließlich als Ausgangspunkt, um Anforderungen an ein integriertes, dienstorientiertes Abrechnungsmanagement systematisch abzuleiten. Ergebnis war ein *strukturierter Anforderungskatalog*, der sowohl zur Bewertung gegenwärtiger Ansätze als auch zur Entwicklung eines neuen Lösungsansatzes verwendet wurde.

Die Analyse gegenwärtiger Arbeiten, die innerhalb von Standardisierungs– und Industriegremien sowie in der Forschung angefertigt wurden, zeigte, dass zwar bereits wertvolle Ergebnisse in diesem Bereich erzielt wurden, aber bisher ein umfassender, integrierter Ansatz für ein dienst– und kundenorientiertes Abrechnungsmanagement fehlte. Insbesondere fanden die in Outsourcing–Szenarios auftretenden dynamischen Aspekte im Abrechnungsmanagement nicht genügend Beachtung in den bisher entwickelten Managementansätzen. Verfolgtes Ziel musste es somit sein, eine Managementlösung zu konzipieren, welche (a) nicht mehr nur die abrechnungsrealisierenden Komponenten ausschließlich der Betriebsphase und (b) diese Komponenten nicht mehr in isolierter Art und Weise steuert. Statt dessen ist ein Abrechnungsmanagement gefragt, das v.a. die dynamischen Beziehungen in das Management miteinbezieht, welche durch die Kooperation der abrechnungsrealisierenden Komponenten entstehen.

Aus diesem Grund wurde in einem nächsten Schritt ein detailliertes, auf die Abrechnung und das Abrechnungsmanagement fokussiertes *Prozessmodell* entwickelt, welches die dynamischen Aspekte der Abrechnung ausdrückt. Hierbei wurde der Abrechnungs– und Abrechnungsmanagementvorgang entlang des vollständigen Dienstlebenszyklus in Teilprozesse gegliedert, welche für sich gesehen eine Teilfunktionalität der Abrechnung resp. des dazugehörigen Managements erfüllen. Jeder Teilprozess wurde durch ein eigenständiges *Teilprozessmodell* repräsentiert, welches die innerhalb eines Teilprozesses auftretenden Aktivitäten, ausführende Akteure und Ein–/Ausgabeentitäten beschreibt. Die Akteure und identifizierten Entitäten wurden schließlich in ein *Abrechnungsdienstmodell* zusammengeführt, welches, als Erweiterung des generischen MNM Dienstmodells, die Assoziationen zwischen den an der Abrechnung beteiligten Akteure und Entitäten bei ausgeblendeter Zeitdimension darstellt. Beide Modelle dienen als notwendige Grundlage für die im Abrechnungsmanagement zu spezifizierenden Managementanweisungen.

Die im Anschluss entworfene Managementlösung basiert auf der grundlegenden Idee, den Abrechnungsvorgang als solchen, zunächst ungeachtet der realisierenden Komponenten, aus *Prozesssicht* zu steuern und zu überwachen. Damit sollte eine ganzheitliche, integrierte und insbesondere die dynamischen Aspekte bereits enthaltende Sicht auf die Managementaufgabe verwirklicht werden. Für die tatsächliche Umsetzung der Managementaufgabe wurden *policy–basierte Konzepte* angewendet. Dieser grundlegende Lösungsansatz ist hierbei nicht auf das dienst– und kundenorientierte Abrechnungsmanagement beschränkt, so dass darauf geachtet wurde, dass die Lösung generell auch für andere Managementfunktionsbereiche zur Durchsetzung eines prozessorientierten IT Managements einsetzbar ist. Hierfür wurde auf Basis der grundlegenden Elemente eines Prozessmodells die notwendige Ausdrucksmächtigkeit der

 Als Buch erhältlich im Verlag Dr. Hut, München, www.dr.hut-verlag.de (ISBN 3-934767-97-4)

Policy–Sprache untersucht, um darauf aufbauend eine *Policy Description Language (PDL)* zu spezifizieren. Mittels der PDL ist es möglich, in prozessorientierter Art und Weise Managementanweisungen zu formulieren, welche automatisiert mittels einer entsprechenden policy–basierten Managementanwendung durchgesetzt werden können. Durch die für den Administrator einheitliche Managementschnittstelle in Form der Policy–Sprache und der gewonnenen Automatisierung von wiederkehrenden Managementvorgängen wird der Administrator wesentlich entlastet. Um den Administrator und Policy–Ersteller bei der Spezifikation von Policies anzuleiten und um damit den Vorgang der Policy–Erstellung zu erleichtern, wurde eine allgemeine *Methodik zur Policy–Spezifikation* erarbeitet. Die Methodik sieht das Einbeziehen von Prozessmodellen und statischen Dienstmodellen, wie sie im Rahmen dieser Arbeit für das Abrechnungsmanagement entwickelt wurden, vor. Anschließend wurde ein *objektorientierter Entwurf des Managementsystems* durchgeführt. Dieses ist aufgeteilt in eine *Managementanwendung*, welche die Schnittstelle zum Administrator bildet sowie die Policies automatisch durchsetzt, und in eine *Integrationsschicht*, welche es ermöglicht, bereits existierende, vorgangsrealisierende Komponenten in das Management miteinzubeziehen. Sowohl für die Managementanwendung als auch für die Integrationsschicht wurden Klassen und Schnittstellen spezifiziert. Die Integrationsschicht spielt für die Tragfähigkeit des Gesamtsystems eine wesentliche Rolle, da damit bereits durchgeführte Investitionen in diesem Bereich geschützt werden.

Als Tragfähigkeitsnachweis der Umsetzbarkeit des entwickelten Lösungskonzepts wurde die Beschreibung der auf MASA basierenden *prototypischen Implementierung* herangezogen. Die Managementanwendung wurde in Form von drei miteinander kooperierenden, mobilen Agenten realisiert. Die Grammatik der PDL wurde tatsächlich mittels XML–Schema umgesetzt, so dass ein Standard–XML–Parser für die Policy–Interpretierung verwendet werden konnte. Zusätzlich wurden einige Agenten der Integrationsschicht beschrieben. Zum Abschluss erfolgte die *Evaluierung* des in dieser Arbeit entwickelten Lösungsansatzes mittels Anwendung des erstellten Anforderungskatalogs. Jedem in dieser Arbeit eingesetzten Konzept wurde dediziert die dadurch erfüllten Anforderungen zugewiesen. Es stellte sich heraus, dass die zu Beginn identifizierten Anforderungen alle erfüllt wurden.

Offene Forschungsfragestellungen

Die in dieser Arbeit spezifizierte Methodik zur Erstellung von Policies bietet sich hervorragend dafür an, eine entsprechende Werzeugunterstützung zu konzipieren, welche in das bestehende System integriert wird. Damit wird der Policy–Ersteller von Aktivitäten während der Policy–Spezifikation befreit, die automatisch durchgeführt werden könnten. Da die in dieser Arbeit entwickelte Policy–Sprache explizit die Wiederverwendung von Policies auf Basis des dadurch gesteuerten Teilprozesses unterstützt, wäre ein Werkzeug denkbar, das dies als aktive Wissensbasis nutzt. In [Schm 02a] wurde ein generisches Tool zur worflow–basierten Unterstützung von beliebigen Methodiken entwickelt, welches sich hervorragend als Grundlage für diese Aufgabe eignet.

Wie bereits mehrmals erwähnt, ist der entwickelte Lösungsansatz bestehend aus der PDL, der Methodik sowie der entworfenen und implementierten Managementanwendung ohne weiteres in anderen Managementfunktionsbereichen einsetzbar. Hierzu ist allerdings die Entwicklung von sowohl bereichsspezifischen Prozessmodellen als auch statischen Dienstmodellen, wie sie im Rahmen dieser Arbeit für das dienstorientierte Abrechnungsmanagement angefertigt wurden, notwendig, um sie als Basis für die Spezifikation von Policies zu verwenden.

Eine trotz der zahlreichen Arbeiten in diesem Bereich immer noch ungelöste Fragestellung im Abrechnungsmanagement ist die sichere Prognostizierung von entstehenden Kosten und damit zusammenhängend die kostendeckende Tarifierung. Da das zukünftige Verhalten sowohl des Kunden als auch des Nutzers abgeschätzt werden muss, damit ein kostendeckendes Tarifmodell entwickelt werden kann, gestaltet sich diese Aufgabe besonders schwierig. Bei Individualdiensten ist die Entwicklung eines Tarifmodells gegenüber Standarddiensten insofern leichter, da der Kunde dem Dienstleister seine Wünsche bzgl. des Tarifmodells aktiv äußert. Demnach ist das „Kauf“verhalten sehr genau prognostizierbar. Allerdings wird die Tarifierung für Individualdienste im Gegenzug dadurch erschwert, dass der Dienst per se noch nie in dieser Form existiert hat und damit weder Kenntnisse über die zu erwartenden Kosten noch das Nutzungsverhalten verfügbar sind. Im Rahmen einer am Lehrstuhl durchgeführten Diplomarbeit [Kali 02] wurde untersucht, inwiefern dieses Problem durch Anwendung wissensbasierter Systeme aus der Künstlichen Intelligenz (KI) gelöst werden kann. Idee war hierbei, die auf Basis der Bereitstellung von Individualdiensten gesammelten Erfahrungen auch in Bezug zu den entstandenen Kosten für die Tariferstellung zu nutzen, sofern der Individualdienst aus einer individuellen Zusammenstellung von Standarddiensten besteht. Hierbei konnten allerdings nur erste (Daten–)Modelle entwickelt werden, so dass noch vielfältige Arbeit in diese Richtung getätigt werden muss, um das grundsätzliche Prinzip auf Tauglichkeit bewerten zu können.

Eine kostengünstige, effiziente und effektive Kooperation von heterogen zusammengestellten Abrechnungslösungen ist nur durch standardisierte Nutzungs– und Managementschnittstellen möglich. Erfreulicherweise zeigen sich für Nutzungsschnittstellen im Bereich der Nutzungserfassung vielversprechende Initiativen (z.B. IPDR). Allerdings fehlt immer noch eine vergleichbare Initiative für die Rechnungsstellung und Gebührenberechnung als auch für Managementschnittstellen. Dies würde allerdings voraussetzen, dass die Hersteller die durchaus vorhandenen Schnittstellen offen legen, um dies als Grundlage für eine Standardisierung zu verwenden. Dies wird bisher aus konkurrenz– und marktpolitischen Gründen von den Herstellern abgelehnt. Hier sind sowohl die Standardisierungsgremien als auch die Käufer dieser Produkte gefragt, Druck auf die jeweiligen Hersteller auszuüben.

Die vorliegende Arbeit beschäftigte sich mit der Entwicklung einer Abrechnungsmanagementlösung für Individualdienste. Charakteristisch hierbei ist, dass sowohl die einzelnen Phasen als auch der Dienstlebenszyklus selber relativ lange andauern. Betrachtet man die gegenwärtig anvisierten Szenarios im Mobilfunkbereich (UMTS) bzgl. der sogenannten context–aware ser-

 Als Buch erhältlich im Verlag Dr. Hut, München, www.dr.hut-verlag.de (ISBN 3-934767-97-4)

vices, welche die schnelle, individuelle Zusammenstellung eines kontextabhängigen Dienstes vorsehen, so steht das Abrechnungsmanagement vor vollständig neuen Herausforderungen: Der Dienstlebenszyklus und damit die einzelnen Phasen sind um ein vielfaches kürzer, so dass die Dynamik entsprechend höher ist. Damit muss das Managementsystem entsprechend performant ausgelegt werden. Eine umfassende Standardisierung ist in diesem Bereich zudem notwendig, wie z.B. von Abrechnungseinheiten, Tarifstrukturen, Schnittstellen, etc., damit die mobile Nutzung dieser Art der Dienste, insbesondere im Falle des Roamings, ermöglicht wird.

 Als Buch erhältlich im Verlag Dr. Hut, München, www.dr.hut-verlag.de (ISBN 3-934767-97-4)

Anhang A

Die Grammatik der PDL in XML–Schema

Nachfolgend ist die unter Verwendung von XML–Schema in [Danc 03] spezifizierte Grammatik der PDL notiert.

```
<?xml version="1.0" encoding="UTF-8"?>
<xsd:schema
        xmlns:xsd="http://www.w3.org/2001/XMLSchema">
        targetNamespace="pdl"
        xmlns:pdl="http://www.nm.informatik.uni-muenchen.de/pdl">
<xsd:annotation>
 <xsd:documentation xml:lang="de">
        Sprache zur Definition von Policies .
         Vitalian A. Danciu, 2002
         Version 0.7
 </xsd:documentation>
</xsd:annotation>
<!-- ROOT ELEMENTS AND THEIR TYPES -->
<xsd:element name="policySet" type="policySetType"/>
<xsd:element name="comment" type="xsd:string"/>
<!-- The elements that MAY be defined standalone in document instances -->
<xsd:complexType name="policySetType">
 <xsd:sequence>
  <xsd:element name="policy" type="policyType"
                         minOccurs="1" maxOccurs="unbounded"/>
  <xsd:element name="roleDefinition" type="roleDefinitionType"
                         minOccurs="0" maxOccurs="unbounded"/>
  <xsd:element name="subject" type="entityContainer"
                         minOccurs="0" maxOccurs="unbounded"/>
  <xsd:element name="target" type="entityContainer"
                         minOccurs="0" maxOccurs="unbounded"/>
  <xsd:element name="event" type="eventType"
                         minOccurs="0" maxOccurs="unbounded"/>
  <xsd:element name="action" type="actionType"
```

```
                              minOccurs="0" maxOccurs="unbounded"/>
  <xsd:element name="constraint" type="constraintType"
                              minOccurs="0" maxOccurs="unbounded"/>
  <xsd:element name="policyGroup" type="group"
                              minOccurs="0" maxOccurs="unbounded"/>
  <xsd:element ref="comment" minOccurs="0"/>
        </xsd:sequence>
</xsd:complexType>
<!-- END ROOT ELEMENTS AND THEIR TYPES -->
<!-- BASE TYPES -->
<!-- Abstract base type for all nodes that are referable by ID.
                These MAY be library items. -->
<xsd:complexType name="referable" abstract="true">
 <xsd:sequence>
        <xsd:element name="id" type="xsd:ID"/>
        <xsd:element ref="comment" />
        <xsd: attribute name="libraryItem" type="xsd:boolean"
                 use="optional" default="false"/>
 </xsd:sequence>
</xsd:complexType>
<!-- Referable items can be assigned to one or more processes.
A list of valid process names is embedded into the type definition . -->
<xsd:complexType name="processAssignable" abstract="true">
 <xsd:complexContent>
  <xsd:extension base="pdl:referable">
   <xsd:sequence>
        <xsd:element name="relatedProcess">
                <xsd:simpleType>
                 <xsd:union>
                 <!-- A list of processes that apply.
                Process names MUST NOT contain whitespace. -->
                  <xsd:simpleType>
                        <xsd: list itemType="xsd:String">
                                <xsd: restriction base="xsd:string">
                                        <xsd:enumeration value="accounting"/>
                                        <xsd:enumeration value="charging"/>
                                        <xsd:enumeration value="billing"/>
                                        <xsd:enumeration value="change"/>
                                        <xsd:enumeration value="deployMeters"/>
                                        <!-- ... -->
                                </xsd: restriction >
                        </xsd: list >
                  </xsd:simpleType>
                  <!-- MEANING: this item is related to all processes -->
```

 Als Buch erhältlich im Verlag Dr. Hut, München, www.dr.hut-verlag.de (ISBN 3-934767-97-4)

```
                  <xsd:simpleType>
                        <xsd: restriction  base="xsd:string">
                              <xsd:enumeration value="ALL"/>
                        </xsd: restriction >
                  </xsd:simpleType>
                   <xsd:simpleType>
                        <xsd: restriction  base="xsd:string">
                              <xsd:enumeration value="OTHER"/>
                        </xsd: restriction >
                  </xsd:simpleType>
                 </xsd:union>
                </xsd:simpleType>
        </xsd:element>
    </xsd:sequence>
   </xsd: extension >
  </xsd:complexContent>
</xsd:complexType>
<xsd:complexType name="abstractValueType" abstract="true">
        <xsd: attribute  name="type">
                <xsd:simpleType>
                        <xsd: restriction  base="xsd:token">
                              <xsd:enumeration value="float"/>
                              <xsd:enumeration value="integer"/>
                              <xsd:enumeration value="string"/>
                              <xsd:enumeration value="boolean"/>
                              <xsd:enumeration value="dateTime"/>
                        </xsd: restriction >
                </xsd:simpleType>
                </xsd: attribute >
</xsd:complexType>
<!-- END BASE TYPES -->
<!-- STANDALONE COMPONENT TYPES -->
<!-- A group is a list of references to items of the same type.
Groups themselves CANNOT be referenced.
There is one generic group defined for grouping all
 sorts of elements like policies , targets , events etc .
Beware: this implies that the consistency of the group
 content CANNOT be assured by the parser , i . e . a targetSet
 containig a group consisting of actions and events
DOES NOT violate this schema !!
Such anomalies MUST be checked by the interpreter . -->
<xsd:complexType name="group">
    <xsd:sequence>
        <xsd:element name="items" type="xsd:IDREFS"/>
```

```
        <xsd: attribute  name="itemsType" use="required">
                <xsd:simpleType>
                        <xsd: restriction  base="xsd:string">
                                <xsd:enumeration value="target"/>
                                <xsd:enumeration value="event"/>
                                <xsd:enumeration value="action"/>
                                <xsd:enumeration value="policy"/>
                        </xsd: restriction >
                </xsd:simpleType>
        </xsd: attribute >
    </xsd:sequence>
</xsd:complexType>
<!-- This complexType is defined  for  the  SUBJECT and
TARGET elements that have many  similarities  -->
<xsd:complexType name="entityContainer">
        <xsd:sequence>
                <xsd:choice>
                        <xsd:sequence>
                                <xsd:element name="domain" type="domainType"/>
                                <xsd:element name="entity" type="xsd:string"/>
                        </xsd:sequence>
                        <xsd:element name="role" type="xsd:IDREF"/>
                </xsd:choice>
        </xsd:sequence>
</xsd:complexType>
<!-- The domain type is  just  a  string  containing  an  absolute
or  relative  path e.g. "/a/bxx/cx" or "axxx/bx/cxx".
It  is  encapsulated  in a named type  of  its  own to  allow  for  extensibility . -->
<xsd:simpleType name="domainType">
        <xsd: restriction  base="xsd:token">
                <xsd:pattern  value="/?([0-9a-zA-Z]/){1,}"/>
<!--
In  EBNF:
pattern     ::= ['/']{('0'|'1'|..|'9'|'    a'|'b  '|..|'  z'|'A'|'B  '|..|' Z')'/'}
-->
        </xsd: restriction >
</xsd:simpleType>
<!-- Roles are to be  defined  separately  using  this  type.
When actually  used,  an  existing  role  MUST be referenced
using  an element  of  the  type  roleType -->
<xsd:complexType name="roleDefinitionType">
<xsd:complexContent>
  <xsd:extension  base="pdl:referable">
        <xsd:sequence>
```

```
            <xsd:element name="name" type="xsd:token"/>
            <xsd:element name="description" type="xsd:string"/>
        </xsd:sequence>
  </xsd: extension >
 </xsd:complexContent>
</xsd:complexType>
<!-- When assigning a role to an item , the role MUST already
be defined so it can be assigned by using its ID-reference.
One or more roles can be assigned to the same item. -->
<xsd:simpleType name="roleType">
        <xsd: restriction base="xsd:IDREFS"/>
</xsd:simpleType>

<!-- Targets can be single ones or reference to a group of targets . -->
<xsd:complexType name="targetSetType">
        <xsd:sequence>
            <xsd:choice>
                <xsd:element name="target" type="entityContainer"/>
                <xsd:element name="targetRef" type="xsd:IDREF"/>
                <xsd:element name="targetGroup" type="group" />
            </xsd:choice>
        </xsd:sequence>
</xsd:complexType>
<!-- Events can be single ones or a reference to a group of events -->
<xsd:complexType name="eventSetType">
        <xsd:sequence>
        <xsd:choice>
            <xsd:element name="event" type="eventType"/>
            <xsd:element name="eventRef" type="xsd:IDREF"/>
            <xsd:element name="eventGroup" type="group"/>
        </xsd:choice>
        </xsd:sequence>
</xsd:complexType>
<!-- A single event with a name -->
<xsd:complexType name="eventType">
<xsd:complexContent>
  <xsd:extension base="pdl:processAssignable">
        <xsd:sequence>
            <xsd:element name="eventName" type="xsd:Name"/>
        </xsd:sequence>
  </xsd: extension >
 </xsd:complexContent>
</xsd:complexType>
<!-- An action set contains one action or a group of actions . -->
```

```
<xsd:complexType name="actionSetType">
        <xsd:sequence>
         <xsd:choice>
                <xsd:element name="action" type="actionType"/>
                <xsd:element name="actionRef" type="xsd:IDREF"/>
                <xsd:element name="actionGroup" type="group"/>
         </xsd:choice>
        </xsd:sequence>
</xsd:complexType>
<!-- An action consists of a generic action to be executed
and an optional error action to be executed if the genericAction fails .
 Alternatively , an event can be generated and propagated -->
<xsd:complexType name="actionType">
<xsd:complexContent>
  <xsd:extension base="pdl:processAssignable">
        <xsd:sequence>
         <xsd:choice>
                 <xsd:sequence>
                 <xsd:element name="default" type="genericActionType"/>
                 <xsd:element name="error" type="genericActionType"
                  minOccurs="0"/>
                 </xsd:sequence>
         </xsd:choice>
        </xsd:sequence>
  </xsd: extension >
 </xsd:complexContent>
</xsd:complexType>
<!-- END STANDALONE COMPONENT TYPES -->
<!-- BUILDING BLOCKS FOR STANDALONE COMPONENTS -->
<xsd:complexType name="binaryLogicalOperator">
        <xsd:sequence>
                <xsd:element name="constraint" type="constraintType"
                        maxOccurs="unbounded"/>
        </xsd:sequence>
</xsd:complexType>
<!-- A constraint set contains either a single condition or
an expression in either conjunctive or disjuctive normal form.
-->
<xsd:complexType name="constraintSetType">
  <xsd:sequence>
   <xsd:choice>
        <xsd:element name="constraintCNF">
                <xsd:complexType>
                        <xsd:sequence>
```

 Als Buch erhältlich im Verlag Dr. Hut, München, www.dr.hut-verlag.de (ISBN 3-934767-97-4)

```
                        <xsd:element name="or" type="binaryLogicalOperator"
                            maxOccurs="unbounded"/>
                    </xsd:sequence>
                </xsd:complexType>
            </xsd:element>
            <xsd:element name="constraintDNF">
                <xsd:complexType>
                    <xsd:sequence>
                        <xsd:element name="and" type="binaryLogicalOperator"
                            maxOccurs="unbounded"/>
                    </xsd:sequence>
                </xsd:complexType>
            </xsd:element>
            <xsd:element name="constraint" type="constraintType" />
            <xsd:element name="constraintRef" type="xsd:IDREF"/>
   </xsd:choice>
  </xsd:sequence>
</xsd:complexType>
<!-- A constraint is an expression that can be a unary or binary predicate. -->
<xsd:complexType name="constraintType">
        <xsd:choice>
                <xsd:element name="equal" type="binaryPredicate"/>
                <xsd:element name="smaller" type="binaryPredicate"/>
                <xsd:element name="greater" type="binaryPredicate"/>
                <xsd:element name="greaterEqual" type="binaryPredicate"/>
                <xsd:element name="smallerEqual" type="binaryPredicate"/>
                <xsd:element name="predicate" type="valueType"/>
                <xsd:element name="negatedPredicate" type="valueType"/>
        </xsd:choice>
</xsd:complexType>
<xsd:complexType name="binaryPredicate">
        <xsd:sequence>
                <xsd:element name="value" type="valueType"
                        minOccurs="2" maxOccurs="2"/>
        </xsd:sequence>
</xsd:complexType>
<xsd:complexType name="genericActionType">
        <xsd:sequence>
         <xsd:choice>
                <xsd:element name "invoke" type="invocation"/>
                <xsd:element name="generateEvent">
                        <xsd:complexType>
                        <xsd:sequence>
                         <xsd:element name="eventName" type="xsd:string"/>
```

```
                    <xsd:element name="parameterSet" type="parameterSetType"
                    minOccurs="0"/>
                   </xsd:sequence>
                   </xsd:complexType>
            </xsd:element>
       </xsd:sequence>
      </xsd:choice>
     </xsd:sequence>
</xsd:complexType>
<!-- A method invocation consists of an optional object,
a method name and an optional parameterSet.
The object can either be a Name or a <subject/> element;
if it is omitted, the targets of the policy are used as object. -->
<xsd:complexType name="invocation">
     <xsd:sequence>
            <xsd:choice>
                   <xsd:element name="object" type="xsd:Name" minOccurs="0"/>
                   <xsd:element name="object">
                          <xsd:complexType>
                                 <element name="subject" empty=true minOccurs="0" />
                          </xsd:complexType>
                   </xsd:element>
            <xsd:element name="method" type="xsd:Name"/>
            <xsd:element name="parameterSet" type="parameterSetType" minOccurs="0"/>
       </xsd:sequence>
</xsd:complexType>
<!-- A value is one of float, int, string, boolean or ISO dateTime.
It can be either
- a constant/literal, or
- a value retrieved from an attribute of an object, or
- a return value of a method invocation
Arrays of constants are also values.
-->
<xsd:complexType name="valueType">
 <xsd:complexContent>
  <xsd:extension base="abstractValueType">
     <xsd:choice>
     <xsd:element name="literal">
      <xsd:simpleType>
<xsd:union memberTypes="xsd:float xsd:integer xsd:string xsd:boolean xsd:dateTime"/>
      </xsd:simpleType>
     </xsd:element>
      <xsd:element name="array" type="arrayType"/>
      <xsd:element name="functionValue" type="genericActionType"/>
```

```
                <xsd:element name="attributeValue">
                        <xsd:complexType>
                                <xsd:sequence>
                                <xsd:element name="object" type="xsd:Name"/>
                                <xsd:element name="attribute" type="xsd:Name"/>
                                </xsd:sequence>
                        </xsd:complexType>
                </xsd:element>
        </xsd:choice>
  </xsd: extension >
 </xsd:complexContent>
</xsd:complexType>
<!-- An array is a list of items of the same type.
A string array is a whitespace delimited list of strings. -->
<xsd:complexType name="arrayType">
 <xsd:sequence>
  <xsd:element name="arrayContent">
        <xsd:simpleType>
                <xsd:union>
                 <xsd:simpleType><xsd:list itemType="xsd:float"/></xsd:simpleType>
                 <xsd:simpleType><xsd:list itemType="xsd:integer"/></xsd:simpleType>
                 <xsd:simpleType><xsd:list itemType="xsd:string"/></xsd:simpleType>
                 <xsd:simpleType><xsd:list itemType="xsd:boolean"/></xsd:simpleType>
                 <xsd:simpleType><xsd:list itemType="xsd:dateTime"/></xsd:simpleType>
                 <xsd:simpleType>
                        <xsd: restriction base="xsd:string">
                                <xsd:enumeration value="null"/>
                        </xsd: restriction >
                 </xsd:simpleType>
                </xsd:union>
                </xsd:simpleType>
         </xsd:element>
        </xsd:sequence>
</xsd:complexType>
<!-- END BUILDING BLOCKS FOR STANDALONE COMPONENTS -->
<!-- POLICY -->
<xsd:complexType name="policyType">
 <xsd:complexContent>
  <xsd:extension base="pdl:processAssignable">
        <xsd:sequence>
         <xsd:element name="policyDomain" type="domainType" minOccurs="0"/>
         <xsd:element name="subject" type="entityContainer" minOccurs="0"/>
         <xsd:element name="targetSet" type="targetSetType"/>
         <xsd:element name="eventSet" type="eventSetType"/>
```

```
        <xsd:element name="constraintSet" type="constraintSetType" minOccurs="0"/>
        <xsd:element name="actionSet" type="actionSetType"/>
        <xsd:element name="policyDescriptor">
         <xsd:complexType>
          <xsd:sequence>
               <xsd:element name="createdBy" type="xsd:token" minOccurs="0"/>
               <xsd:element name="dateOfCreation" type="xsd:date" minOccurs="0"/>
               <xsd:element name="lastModified" type="xsd:dateTime" minOccurs="0"/>
               <xsd:element name="lastModifiedBy" type="xsd:token" minOccurs="0"/>
               <xsd:element name="expires" type="xsd:dateTime" minOccurs="0"/>
          </xsd:sequence>
         </xsd:complexType>
        </xsd:element>
       </xsd:sequence>
 <xsd: attribute  name="isEnabled" type="xsd:boolean"
               use="optional" default="true"/>
 <xsd: attribute  name="priority" use="optional" default="0">
               <xsd:simpleType>
                      <xsd: restriction  base="xsd:integer">
                             <xsd:minInclusive  value="0"/>
                             <xsd:maxInclusive  value="99"/>
                      </xsd: restriction >
               </xsd:simpleType>
       </xsd: attribute >
       <xsd: attribute  name="version" type="xsd:float"
               use="optional" default="1.0"/>
  </xsd: extension >
 </xsd:complexContent>
</xsd:complexType>
</xsd:schema>
```

ABKÜRZUNGEN

— A —

AAA	Authentication, Authorization and Accounting Group
ADIF	Accounting Data Interchange Format
ARM	Application Response Measurement API

— B —

BPO	Business Process Outsourcing
BSS	Business Support Systems

— C —

CDP	CpuDataProvider
CDR	Customer Detailed Record
CIM	Common Information Model
CORBA	Common Object Request Broker Architecture
COPS	Common Open Policy Service
CP	Cumulus Point
CPC	Cumulus Pricing Contract
CPS	Cumulus Pricing Scheme
CSM	Customer Service Management

— D —

DII	Dynamic Invocation Interface
DMTF	Distributed Management Task Force
DPA	DataProviderAgent
DSI	Dynamic Skeleton Interface

— E —

ETSI	European Telecommunications Standards Institute
EBNF	Extended Backus-Naur Form

— F —

FQDN	Fully Qualified Domain Name
FTAM	File Transfer, Access and Management (Protocol)

— G —

GUI	Graphical User Interfaces

— I —

ID	Identifikator
IESA	Intranet Extranet Service Area
IETF	Internet Engineering Task Force
IPDR	IP Detailed Record
IPPM	IP Performance Metrics
ISA	Intranet Service Area
ISP	Internet Service Provider
ITIL	IT Infrastructure Library

— J —

JNDI	Java Naming and Directory Interface

— L —

LDAP	Lightweight Directory Access Protocol

— M —

MASA	Mobile Agent System Architecture
MASIF	Mobile Agent System Interoperability Facilities
MIB	Management Information Base
MHS	Message Handling System
MNM	Munich Network Management

— N —

NGOSS	Next Generation OSS
NMF	Network Management Forum

— O —

OGC	Office of Government Commerce
OMA	Object Management Architecture
OMG	Object Management Group
ORB	Object Request Broker
OSS	Operations Support Systems

— P —

PCIM	Policy CIM
PDL	Policy Description Language
PEA	Policy Enforcement Agent
PMA	Policy Manager Agent
PMP	Paris Metro Pricing
PRA	Policy Repository Agent
POD	Point of Decision
POE	Point of Execution

— R —

RFC	Request for Comments
RMI	Remote Method Invocation
RSVP	Resource Reservation Protocol

 Als Buch erhältlich im Verlag Dr. Hut, München, www.dr.hut-verlag.de (ISBN 3-934767-97-4)

RTFM	Realtime Traffic Flow Measurement Group

— S —

SLA	Service Level Agreement
SNMP	Simple Network Management Protocol
SRL	Simple Ruleset Language
SSL	Secure Socket Layer

— T —

TMForum	TeleManagement Forum
TMN	Telecommunications Management Network
TINAC	Telecommunications Information Networking Architecture Consortium
TOM	Telecom Operations Map

— U —

UML	Unified Modeling Language

— X —

XML	Extensible Markup Language

ABBILDUNGEN

 Als Buch erhältlich im Verlag Dr. Hut, München, www.dr.hut-verlag.de (ISBN 3-934767-97-4)

 Als Buch erhältlich im Verlag Dr. Hut, München, www.dr.hut-verlag.de (ISBN 3-934767-97-4)

TABELLEN

LITERATUR

[AlCh 01] ALTMANN, J. und K. CHU: *How to charge for network services — flat–rate or usage–based?* Computer Networks, 36(5–6):519 – 531, August 2001.

[Asse 02] ASSENMACHER, W.: *Einführung in die Ökonometrie*. Oldenbourg, 6. Auflage, 2002.

[BHP+ 00] BROY, M., H.-G. HEGERING, A. PICOT, A. BUTTERMANN, M. GARSCHHAMMER, R. HAUCK und S. VOGEL: *Kommunikations– und Informationstechnik 2010, Trends in Technologie und Markt*. SecuMedia Verlag, ISBN 3-922746-35-7, Ingelheim, September 2000.

[Boet 90] BÖTSCH, E.: *Ein umfassendes Abrechnungsmodell für ein integriertes Netzmanagement*. Dissertation, Technische Universität München, März 1990.

[BRS 02] BRENNER, M., I. RADISIC und M. SCHOLLMEYER: *A Criteria Catalog based Methodology for Analyzing Service Management Processes*. In: FERIDUN, M., P. KROPF und G. BABIN (Herausgeber): *Proceedings of the 13th IFIP/IEEE International Workshop on Distributed Systems: Operations & Management (DSOM 2002)*, Lecture Notes in Computer Science (LNCS) 2506, Seiten 145–156, Montreal, Canada, Oktober 2002. IFIP/IEEE, Springer, `http://www.nm.informatik.uni-muenchen.de/Literatur/MNMPub/Publikationen/brs02/brs02.shtml`.

[BTLD 01] BHUSHAN, B., M. TSCHICHOLZ, E. LERAY und W. DONNELLY: *Federated Accounting: Service Charging and Billing in Business-to-Business Environment*. In: PAVLOU, G., N. ANEROUSIS und A. LIOTTA (Herausgeber): *Proceedings of the 7th International IEEE/IFIP Symposium on Integrated Network Management (IM 2001)*, Seiten 107–121, Seattle, Washington USA, Mai 2001. IEEE Publishing.

[C014] *Application Response Measurement, Issue 3.0 — Java Binding*. Technical Standard C014, The Open Group, Oktober 2001.

[C807] *Application Response Measurement (ARM) API* . Technical Standard C807, The Open Group, Juli 1998.

[CZS 01] CARLE, G., S. ZANDER und T. SZEBY: *Policy-basiertes Metering für IP–Netze*. In: KILLAT, U. und W. LAMERSDORF (Herausgeber): *Kommunikation in Verteilten Systemen (KiVS)*, Seiten 21 – 33, Hamburg, Deutschland, September 2001. Gesellschaft für Informatik (GI), Springer.

[Dami 02] DAMIANOU, N. C.: *A Policy Framework for Management of Distributed Systems*. Doktorarbeit, Imperial College of Science, Technology and Medicine, University of London, Department of Computing, Februar 2002.

[Danc 03] DANCIU, V.: *Entwicklung einer policy–basierten Managementanwendung für ein prozessorientiertes Abrechnungsmanagement*. Diplomarbeit, Ludwig-Maximilians-Universität München, Januar 2003.

[DDGH 02] DIAZ, G., S. DUFLOS, V. GAY und E. HORLAIT: *A Comparative Study of Policy Specification Languages for Secure Distributed Applications*. In: FERIDUN, M. und P. KROPF (Herausgeber): *Proceedings of the 13th IFIP/IEEE International Workshop on Distributed Systems: Operations & Management (DSOM 2002)*, Lecture Notes in Computer Science (LNCS), Montreal, Canada, Oktober 2002. IFIP/IEEE, Springer.

[DDLS 00] DAMIANOU, N., N. DULAY, E. C. LUPU und M. S. SLOMAN: *Ponder: A language for Specifying Security and Management Policies for Distributed Systems. The Language Specification Version 2.3*. Imperial College Research Report DoC 2000/1, Imperial College of Science, Technology and Medicine, University of London, Department of Computing, Oktober 2000.

[DLSD 01] DULAY, N., E. LUPU, M. SLOMAN und N. DAMIANOU: *A Policy Deployment Model for the Ponder Language*. In: PAVLOU, G., N. ANEROUSIS und A. LIOTTA (Herausgeber): *Proceedings of the 7th IEEE/IFIP International Symposium on Integrated Network Management (IM 2001)*, Seattle, Washington, USA, Mai 2001. IFIP/IEEE, IEEE Publishing.

[DSP 0108] DMTF SERVICE LEVEL AGREEMENT WORKING GROUP: *CIM Core Policy Model White Paper*. White Paper, Distributed Management Task Force, März 2002, `http://www.dmtf.org/standards/documents/CIM/DSP0108.pdf`.

[EDI 96] KANTOR, M. und A. PRABHAKAR: *Electronic Data Interchange (EDI)*. Federal Information Processing Standards Publications FIPS PUB 161-2, U.S. Department of Commerce/National Institute of Standards and Technology (NIST), April 1996.

[Fisc 01] FISCHER, M.: *Evaluierung von Werkzeugen zur Antwortzeitüberwachung bei der DeTeSystem*. Systementwicklungsprojekt, Technische

Universität München, April 2001, `http://www.nm.informatik.uni-muenchen.de/common/Literatur/MNMPub/Fopras/fisc01/fisc01.shtml`.

[GB 910] *Telecom Operations Map*. Technischer Bericht GB 910 Approved Version 2.1, TeleManagement Forum, März 2000.

[GB 921] *enhanced Telecom Operations Map (eTOM), The Business Process Framework For The Information and Communications Services Industry*. Technischer Bericht GB 921 Approved Version 3.0, TeleManagement Forum, Juni 2002.

[GHH+ 01] GARSCHHAMMER, M., R. HAUCK, H.-G. HEGERING, B. KEMPTER, M. LANGER, M. NERB, I. RADISIC, H. ROELLE und H. SCHMIDT: *Towards generic Service Management Concepts – A Service Model Based Approach*. In: PAVLOU, G., N. ANEROUSIS und A. LIOTTA (Herausgeber): *Proceedings of the 7th International IFIP/IEEE Symposium on Integrated Management (IM 2001)*, Seiten 719–732, Seattle, Washington, USA, Mai 2001. IFIP/IEEE, IEEE Publishing, `http://www.nm.informatik.uni-muenchen.de/Literatur/MNMPub/Publikationen/smtf01/smtf01.shtml`.

[GHH+ 02] GARSCHHAMMER, M., R. HAUCK, H.-G. HEGERING, B. KEMPTER, I. RADISIC, H. ROELLE und H. SCHMIDT: *A Case–Driven Methodology for Applying the MNM Service Model*. In: STADLER, R. und M. ULEMA (Herausgeber): *Proceedings of the 8th International IFIP/IEEE Network Operations and Management Symposium (NOMS 2002)*, Seiten 697–710, Florence, Italy, April 2002. IFIP/IEEE, IEEE Publishing, `http://www.nm.informatik.uni-muenchen.de/Literatur/MNMPub/Publikationen/ghhk02/ghhk02.shtml`.

[GHJV 95] GAMMA, E., R. HELM, R. JOHNSON und J. VLISSIDES: *Design Patterns — Elements of Reusable Object–Oriented Software*. Addison-Wesley Professional Computing Series. Addison-Wesley, 1995. ISBN 0-201-63361-2.

[GHK+ 01] GARSCHHAMMER, M., R. HAUCK, B. KEMPTER, I. RADISIC, H. ROELLE und H. SCHMIDT: *The MNM Service Model — Refined Views on Generic Service Management*. Journal of Communications and Networks, 3(4):297–306, Dezember 2001, `http://www.nm.informatik.uni-muenchen.de/Literatur/MNMPub/Publikationen/ghkr01/ghkr01.shtml`.

[GHR 99] GRUSCHKE, B., S. HEILBRONNER und H. REISER: *Mobile Agent System Architecture — Eine Plattform für flexibles IT–Management*. Technischer Bericht 9902, Ludwig-Maximilians-Universität München, Institut für Informatik, München, August 1999.

[Gigl 00] GIGL, J. G.: *Implementierung der MAFFinder–Schnittstelle für die Mobile Agent System Architecture.* Studienarbeit für das Aufbaustudium Informatik, Technische Universität München, Juli 2000, `http://www.nm.informatik.uni-muenchen.de/common/Literatur/MNMPub/Fopras/gigl00/gigl00.shtml`.

[Gilg 02] GILG, A.: *Aufbau eines Verwaltungssystems für Publikationen am Lehrstuhl.* Systementwicklungsprojekt, Technische Universität München, 2002.

[HaCa 99] HARTANTO, F. und G. CARLE: *Policy-based Billing Architecture for Differentiated Services.* In: *Proceedings of IFIP Fifth International Conference on Broadband Communications (BC'99)*, September 1999.

[HAN 99] HEGERING, H.-G., S. ABECK und B. NEUMAIR: *Integrated Management of Networked Systems – Concepts, Architectures and their Operational Application.* Morgan Kaufmann Publishers, ISBN 1-55860-571-1, 1999. 651 p.

[HAN 99a] HEGERING, H.-G., S. ABECK und B. NEUMAIR: *Integriertes Management vernetzter Systeme — Konzepte, Architekturen und deren betrieblicher Einsatz.* dpunkt-Verlag, ISBN 3-932588-16-9, 1999, `http://www.dpunkt.de/produkte/management.html`. 607 S.

[HaRa 00] HAUCK, R. und I. RADISIC: *Monitoring Application Service Performance – Classification and Analysis of Existing Approaches.* In: *Proceedings of the 7th International Workshop of the HP OpenView University Association (HPOVUA 2000)*, Santorini, Greece, Juni 2000.

[HaRa 01] HAUCK, R. und I. RADISIC: *Service Oriented Application Management — Do Current Techniques Meet the Requirements?* In: *New Developments in Distributed Applications and Interoperable Systems, Proceedings of the 3rd IFIP International Working Conference (DAIS 2001)*, Seiten 295–304, Krakow, Poland, September 2001. Kluwer Academic Publishers, `http://www.nm.informatik.uni-muenchen.de/Literatur/MNMPub/Publikationen/hara01/hara01.shtml`.

[Hauc 01] HAUCK, R.: *Architektur für die Automation der Managementinstrumentierung bausteinbasierter Anwendungen.* Dissertation, Ludwig-Maximilians-Universität München, Juli 2001, `http://www.nm.informatik.uni-muenchen.de/Literatur/MNMPub/Dissertationen/hauc01/hauc01.shtml`.

[Heil 00] HEILBRONNER, S.: *Konzeption einer Architektur für das integrierte Management der Ressourcennutzung nomadischer Systeme in Datennetzen.* Dissertation, Ludwig-Maximilians-Universität München, Januar 2000.

[Heil 98a] HEILBRONNER, S.: *Requirements for Policy-based Management of Nomadic Computing Systems*. In: SETHI, A. S. (Herausgeber): *Proceedings of the 9th IFIP/IEEE International Workshop on Distributed Systems: Operations & Management (DSOM 98)*, Newark, DE, USA, Oktober 1998.

[HuTh 02] HUNTER, J. und M. THIEBAUD: *Telecommunications Billing Systems: Implementing and Upgrading for Profitability*. McGraw-Hill Telecom Professional, Oktober 2002.

[IPDR 311] COTTON, S. und OTHERS: *Network Data Management – Usage (NDM–U) For IP-Based Services*. Technischer Bericht Version 3.1.1, IPDR, Inc., Oktober 2002.

[ISO 10164-10] *Information Technology – Open Systems Interconnection – Systems Management – Part 10: Usage Metering Function for Accounting Purposes*. IS 10164-10, International Organization for Standardization and International Electrotechnical Committee, 1995.

[ISO 10164-6] *Information Technology – Open Systems Interconnection – Systems Management – Part 6: Log Control Function*. IS 10164-6, International Organization for Standardization and International Electrotechnical Committee, 1993.

[ISO 10164-x] *Information Technology – Open Systems Interconnection – Systems Management – Management Functions*. IS 10164-x, International Organization for Standardization and International Electrotechnical Committee, 1991-97.

[ISO 14977] *Information Technology – Syntactic metalanguage – Extended BNF*. IS 14977, International Organization for Standardization and International Electrotechnical Committee, 1996.

[ISO 7498-4] *Information Processing Systems – Open Systems Interconnection – Basic Reference Model – Part 4: Management Framework*. IS 7498-4, International Organization for Standardization and International Electrotechnical Committee, 1989.

[ISO 7498] *Information Processing Systems – Open Systems Interconnection – Basic Reference Model*. IS 7498, International Organization for Standardization and International Electrotechnical Committee, 1984.

[ITIL 00] OFFICE OF GOVERNMENT COMMERCE (OGC) (Herausgeber): *Service Support*. IT Infrastructure Library (ITIL). The Stationary Office, Norwich, UK, 2000.

[ITIL 01] OFFICE OF GOVERNMENT COMMERCE (OGC) (Herausgeber): *Service Delivery*. IT Infrastructure Library (ITIL). The Stationary Office, Norwich, UK, 2001.

[ITU M.3000] *Overview of TMN Recommendations*. Recommendation M.3000, ITU, 1994.

[ITU M.3010] *Principles for a Telecommunications Management Network*. Recommendation M.3010, ITU, 1996.

[ITU M.3400] *TMN Management Functions*. Recommendation M.3400, ITU, April 1997.

[Kali 02] KALIX, E.: *Konzeption und prototypische Implementierung eines Werkzeugs für die dienstorientierte Tariferstellung*. Diplomarbeit, Ludwig-Maximilians-Universität München, September 2002, `http://wwwmnmteam.informatik.uni-muenchen.de/common/Literatur/MNMPub/Diplomarbeiten/kali02/kali02.shtml`.

[Kell 98] KELLER, A.: *CORBA-basiertes Enterprise Management: Interoperabilität und Managementinstrumentierung verteilter kooperativer Managementsysteme in heterogener Umgebung*. Dissertation, Technische Universität München, Dezember 1998.

[Kemp 02] KEMPTER, B.: *Anforderungsanalyse und Klassifizierung von Konflikten im Dienstmanagement*. In: *1. GI–Fachgespräch Applikationsmanagement*, Karlsruhe, Februar 2002. Gesellschaft für Informatik e.V. (GI).

[KKK 96] KOCH, T., C. KRELL und B KRÄMER: *Policy Definition Language for Automated Management of Distributed Systems*. In: *IEEE Workshop on Systems Management*. IEEE, 1996.

[Koch 97] KOCH, T.: *Automated Management of Distributed Systems*. Doktorarbeit, Fern-Universität Hagen, Germany, 1997.

[KRRV01] KEMPTER, B., H. REISER, H. ROELLE und G. VOGT: *Implementierung eines MASIF konformen Agentensystems — Die Mobile Agent System Architecture (MASA) —* . PIK – Praxis der Informationsverarbeitung und Kommunikation, 24(3):141–148, September 2001, `http://www.nm.informatik.uni-muenchen.de/Literatur/MNMPub/Publikationen/krrv01/krrv01.shtml`.

[LuSl 97] LUPU, E. und M. SLOMAN: *Conflict Analysis for Management Policies*. In: LAZAR, A., R. SARACCO und R. STADLER (Herausgeber): *Proceedings of the 5th International Symposium on Integrated Network Management (IM'97)*, San Diego, USA, Mai. Chapman & Hall.

[LuSl 99] LUPU, E. C. und M. S. SLOMAN: *Conflicts in Policy-Based Distributed Systems Management*. IEEE Transactions on Software Engineering, 25(6):852–869, November 1999.

[Mari 97] MARRIOTT, D. A.: *Policy Service for Distributed Systems*. Doktorarbeit, Imperial College London, Juni 1997.

[MASIF] *Mobile Agent System Interoperability Facilities Specification*. OMG TC Document orbos/98-03-09, Object Management Group, März 1998.

[MaSl 96] MARRIOTT, D. und M. SLOMAN: *Implementation of a Management Agent for Interpreting Obligation Policy*. In: *7th International Workshop on Distributed Systems Operations and Management (DSOM 96)*. IFIP/IEEE, Oktober 1996.

[Maul 99] MAUL, O.: *Implementierung des CORBA Notification Service in der MASA-Java-Agentenumgebung*. Fortgeschrittenenpraktikum, Technische Universität München, Dezember 1999.

[MaVa 93] MACKIE-MASON, J. und H. VARIAN: *Some Economics of the Internet*. In: *Tenth Michigan Public Utility Conference at Western Michigan University*, März 1993.

[Merk 02] MERK, S.: *Entwicklung eines Mobilen Agenten zur plattformunabängigen Ausführung von Skripten*. Fortgeschrittenenpraktikum, Ludwig-Maximilians-Universität München, 2002.

[MoSl 93] MOFFETT, J. D. und M. S. SLOMAN: *Policy Conflict Analysis in Distributed System Management*. Journal of Organizational Computing, 1993.

[Moun 97] MOUNTZIA, M.-A.: *Flexible Agents in Integrated Network and Systems Management*. Dissertation, Technische Universität München, Dezember 1997.

[Odly 01] ODLYZKO, A.: *Internet pricing and the history of communications*. Computer Networks, 36(5–6):493 – 517, August 2001.

[Odly 99] ODLYZKO, A.: *Paris Metro Pricing for the Internet*. In: *ACM Conference on Electronic Commerce*, Seiten 140–147, 1999.

[OMG 00-06-19] *Naming Service stand-alone document*. OMG Specification formal/00-06-19, Object Management Group, Juni 2000, `ftp://ftp.omg.org/pub/docs/formal/00-06-19.pdf`.

[OMG 00-06-20] *Notification Service stand-alone document*. OMG Specification formal/00-06-20, Object Management Group, Juni 2000, `ftp://ftp.omg.org/pub/docs/formal/00-06-20.pdf`.

[OMG 01-02-33] *CORBA 2.4.2 full specification*. OMG Specification formal/01-02-33, Object Management Group, Februar 2001, `ftp://ftp.omg.org/pub/docs/formal/01-02-33.pdf`.

[OMG 01-02-39] *CORBA 2.4.2 OMG IDL Syntax and Semantics chapter*. OMG Specification formal/01-02-39, Object Management Group, Februar 2001, `ftp://ftp.omg.org/pub/docs/formal/01-02-39.pdf`.

[OMG 01-03-01] *Event Service, v1.1.* OMG Specification formal/01-03-01, Object Management Group, März 2001, `ftp://ftp.omg.org/pub/docs/formal/01-03-01.pdf`.

[OMG 02-09-01] *Federated Charging and Rating Facility (FCR).* TC Document Telecom/2002-09-01, Object Management Group, September 2002, `ftp://ftp.omg.org/pub/docs/telecom/02-09-01.pdf`.

[OMG 92-11-1] *Object Management Architecture Guide.* OMG TC Document 92-11-1, Object Management Group, September 1992.

[OMG 97-10-12] *CORBA/IIOP 2.1 - Chapter 7 (Interface repository) (w/changebars).* OMG Specification formal/97-10-12, Object Management Group, Oktober 1997, `ftp://ftp.omg.org/pub/docs/formal/97-10-12.pdf`.

[OMG 98-12-9] *CORBAservices: Common Object Services Specification.* Document 98-12-09, Object Management Group, Dezember 1998.

[OMG 99-07-11] *CORBA 2.3 chapter 7 - Dynamic Invocation Interface.* OMG Specification formal/99-07-11, Object Management Group, Juli 1999, `ftp://ftp.omg.org/pub/docs/formal/99-07-11.pdf`.

[ORBacus] OBJECT–ORIENTED CONCEPTS, INC., www.ooc.com: *ORBacus for C++ and Java, Version 3.1.2*, 1999, `http://www.ooc.com/ob/`.

[Radi 02] RADISIC, I.: *Using Policy–Based Concepts to Provide Service Oriented Accounting Management.* In: STADLER, R. und M. ULEMA (Herausgeber): *Proceedings of the 8th International IFIP/IEEE Network Operations and Management Symposium (NOMS 2002)*, Seiten 313–326, Florence, Italy, April 2002. IFIP/IEEE, IEEE Publishing, `http://www.nm.informatik.uni-muenchen.de/Literatur/MNMPub/Publikationen/radi02/radi02.shtml`.

[Radi 02d] RADISIC, I.: *Gesammelte Projektberichte zum Kooperationsprojekt Abrechnungsmanagement 1999 – 2002.* Berichtesammlung, 2002. Kooperation DeTeSystem – MNMTeam.

[Reis 01] REISER, H.: *Sicherheitsarchitektur für ein Managementsystem auf der Basis Mobiler Agenten.* Dissertation, Ludwig-Maximilians-Universität München, Dezember 2001.

[RFC 1272] MILLS, C., D. HIRSH und G.R. RUTH: *RFC 1272: Internet Accounting: Background.* Request for Comments (RFC), Internet Engineering Task Force (IETF), November 1991, `ftp://ftp.isi.edu/in-notes/rfc1272.txt`.

[RFC 1514] GRILLO, P. und S. WALDBUSSER: *RFC 1514: Host Resources MIB*. Request for Comments (RFC), Internet Engineering Task Force (IETF), September 1993, `ftp://ftp.isi.edu/in-notes/rfc1514.txt`.

[RFC 1565] KILLE, S. und N. FREED: *RFC 1565: Network Services Monitoring MIB*. Request for Comments (RFC), Internet Engineering Task Force (IETF), Januar 1994, `ftp://ftp.isi.edu/in-notes/rfc1565.txt`.

[RFC 1662] SIMPSON, W. und ED.: *RFC 1662: PPP in HDLC-like Framing*. Request for Comments (RFC), Internet Engineering Task Force (IETF), Juli 1994, `ftp://ftp.isi.edu/in-notes/rfc1662.txt`.

[RFC 2002] PERKINS, C. und ED.: *RFC 2002: IP Mobility Support*. Request for Comments (RFC), Internet Engineering Task Force (IETF), Oktober 1996, `ftp://ftp.isi.edu/in-notes/rfc2002.txt`.

[RFC 2287] KRUPCZAK, C. und J. SAPERIA: *RFC 2287: Definitions of System-Level Managed Objects for Applications*. Request for Comments (RFC), Internet Engineering Task Force (IETF), Februar 1998, `ftp://ftp.isi.edu/in-notes/rfc2287.txt`.

[RFC 2330] PAXSON, V., G. ALMES, J. MAHDAVI und M. MATHIS: *RFC 2330: Framework for IP Performance Metrics*. Request for Comments (RFC), Internet Engineering Task Force (IETF), Mai 1998, `ftp://ftp.isi.edu/in-notes/rfc2330.txt`.

[RFC 2564] KALBFLEISCH, C., C. KRUPCZAK, R. PRESUHN und J. SAPERIA: *RFC 2564: Application Management MIB*. Request for Comments (RFC), Internet Engineering Task Force (IETF), Mai 1999, `ftp://ftp.isi.edu/in-notes/rfc2564.txt`.

[RFC 2620] ABOBA, B. und G. ZORN: *RFC 2620: RADIUS Accounting Client MIB*. Request for Comments (RFC), Internet Engineering Task Force (IETF), Juni 1999, `ftp://ftp.isi.edu/in-notes/rfc2620.txt`.

[RFC 2621] ZORN, G. und B. ABOBA: *RFC 2621: RADIUS Accounting Server MIB*. Request for Comments (RFC), Internet Engineering Task Force (IETF), Juni 1999, `ftp://ftp.isi.edu/in-notes/rfc2621.txt`.

[RFC 2720] BROWNLEE, N.: *RFC 2720: Traffic Flow Measurement: Meter MIB*. Request for Comments (RFC), Internet Engineering Task Force (IETF), Oktober 1999, `ftp://ftp.isi.edu/in-notes/rfc2720.txt`.

[RFC 2722] BROWNLEE, N., C. MILLS und G. RUTH: *RFC 2722: Traffic Flow Measurement: Architecture*. Request for Comments (RFC), Internet Engineering Task Force (IETF), Oktober 1999, `ftp://ftp.isi.edu/in-notes/rfc2722.txt`.

[RFC 2723] BROWNLEE, N.: *RFC 2723: SRL: A Language for Describing Traffic Flows and Specifying Actions for Flow Groups*. Request for Comments (RFC), Internet Engineering Task Force (IETF), Oktober 1999, `ftp://ftp.isi.edu/in-notes/rfc2723.txt`.

[RFC 2724] HANDELMAN, S., S. STIBLER, N. BROWNLEE und G. RUTH: *RFC 2724: RTFM: New Attributes for Traffic Flow Measurement*. Request for Comments (RFC), Internet Engineering Task Force (IETF), Oktober 1999, `ftp://ftp.isi.edu/in-notes/rfc2724.txt`.

[RFC 2748] DURHAM, D., ED., J. BOYLE, R. COHEN, S. HERZOG, R. RAJAN und A. SASTRY: *RFC 2748: The COPS (Common Open Policy Service) Protocol*. Request for Comments (RFC), Internet Engineering Task Force (IETF), Januar 2000, `ftp://ftp.isi.edu/in-notes/rfc2748.txt`.

[RFC 2749] HERZOG, S., ED., J. BOYLE, R. COHEN, D. DURHAM, R. RAJAN und A. SASTRY: *RFC 2749: COPS usage for RSVP*. Request for Comments (RFC), Internet Engineering Task Force (IETF), Januar 2000, `ftp://ftp.isi.edu/in-notes/rfc2749.txt`.

[RFC 2750] HERZOG, S.: *RFC 2750: RSVP Extensions for Policy Control*. Request for Comments (RFC), Internet Engineering Task Force (IETF), Januar 2000, `ftp://ftp.isi.edu/in-notes/rfc2750.txt`.

[RFC 2819] WALDBUSSER, S.: *RFC 2819: Remote Network Monitoring Management Information Base*. Request for Comments (RFC), Internet Engineering Task Force (IETF), Mai 2000, `ftp://ftp.isi.edu/in-notes/rfc2819.txt`.

[RFC 2865] RIGNEY, C., S. WILLENS, A. RUBENS und W. SIMPSON: *RFC 2865: Remote Authentication Dial In User Service (RADIUS)*. Request for Comments (RFC), Internet Engineering Task Force (IETF), Juni 2000, `ftp://ftp.isi.edu/in-notes/rfc2865.txt`.

[RFC 2866] RIGNEY, C.: *RFC 2866: RADIUS Accounting*. Request for Comments (RFC), Internet Engineering Task Force (IETF), Juni 2000, `ftp://ftp.isi.edu/in-notes/rfc2866.txt`.

[RFC 2924] BROWNLEE, N. und A. BLOUNT: *RFC 2924: Accounting Attributes and Record Formats*. Request for Comments (RFC), Internet Engineering Task Force (IETF), September 2000, `ftp://ftp.isi.edu/in-notes/rfc2924.txt`.

[RFC 2975] ABOBA, B., J. ARKKO und D. HARRINGTON: *RFC 2975: Introduction to Accounting Management*. Request for Comments (RFC), Internet Engineering Task Force (IETF), Oktober 2000, `ftp://ftp.isi.edu/in-notes/rfc2975.txt`.

 Als Buch erhältlich im Verlag Dr. Hut, München, www.dr.hut-verlag.de (ISBN 3-934767-97-4)

[RFC 2977] GLASS, S., T. HILLER, S. JACOBS und C. PERKINS: *RFC 2977: Mobile IP Authentication, Authorization, and Accounting Requirements*. Request for Comments (RFC), Internet Engineering Task Force (IETF), Oktober 2000, `ftp://ftp.isi.edu/in-notes/rfc2977.txt`.

[RFC 3060] MOORE, B., E. ELLESSON, J. STRASSNER und A. WESTERINEN: *RFC 3060: Policy Core Information Model – Version 1 Specification*. Request for Comments (RFC), Internet Engineering Task Force (IETF), Februar 2001, `ftp://ftp.isi.edu/in-notes/rfc3060.txt`.

[RFC 3084] CHAN, K., J. SELIGSON, D. DURHAM, S. GAI, K. MCCLOGHRIE, S. HERZOG, F. REICHMEYER, R. YAVATKAR und A. SMITH: *RFC 3084: COPS Usage for Policy Provisioning (COPS-PR)*. Request for Comments (RFC), Internet Engineering Task Force (IETF), März 2001, `ftp://ftp.isi.edu/in-notes/rfc3084.txt`.

[RFC 3334] ZSEBY, T., S. ZANDER und C. CARLE: *RFC 3334: Policy-Based Accounting*. Request for Comments (RFC), Internet Engineering Task Force (IETF), Oktober 2002, `ftp://ftp.isi.edu/in-notes/rfc3334.txt`.

[RFS 99] REICHL, P., G. FANKHAUSER und B. STILLER: *Auction Models for Multi-Provider Internet Connections*. In: BAUM, D. und N. MÜLLER (Herausgeber): *10. GI/ITG Fachtagung Messung, Modellierung und Bewertung von Rechen- und Kommunikationssystemen (MMB 99)*, Trier, Germany, September 1999. VDE Verlag GmbH, Berlin, Offenbach.

[RJB 98] RUMBAUGH, J., I. JACOBSON und G. BOOCH: *Unified Modeling Language — Reference Manual*. Addison–Wesley, 1998.

[Roel 99] RÖLLE, H.: *Authentisierung und Autorisierung für das Java/CORBA–Agentensystem MASA*. Diplomarbeit, Technische Universität München, August 1999, `http://www.nm.informatik.uni-muenchen.de/common/Literatur/MNMPub/Diplomarbeiten/roel99/roel99.shtml`.

[Roel 99a] RÖLLE, H.: *Prototypische Implementierung des CORBA Topology Service*. Fortgeschrittenenpraktikum, Technische Universität München, Februar 1999.

[Saca 02] SACA, V.: *Aufbau eines Testbeds für das IP Accounting und dessen Einbindung in die Mobile Agent System Architecture (MASA)*. Systementwicklungsprojekt, Technische Universität München, April 2002, `http://www.nm.informatik.uni-muenchen.de/common/Literatur/MNMPub/Fopras/saca02/saca02.shtml`.

[SAWW 01] SEKKAKI, A., L. ALVAREZ, W. WATANABE und C. WESTPHALL: *Development of a Prototype based on TINA Accounting Management Architecture*. In: PAVLOU, G., N. ANEROUSIS und A. LIOTTA (Herausgeber): *Proceedings of the 7th International IEEE/IFIP Symposium on Integrated Network Management (IM 2001)*, Seiten 123–136, Seattle, Washington USA, Mai 2001. IEEE Publishing.

[Schm 01] SCHMIDT, H.: *Entwurf von Service Level Agreements auf der Basis von Dienstprozessen*. Dissertation, Ludwig-Maximilians-Universität München, Juli 2001.

[Schm 02a] SCHMITZ, D.: *Konzeption und prototypische Implementierung eines Werkzeugs zur Unterstützung der MNM Dienstmodellierungsmethodik*. Diplomarbeit, Ludwig-Maximilians-Universität München, September 2002.

[Schw 97] SCHWERDTNER, S.: *Eine Methodik zur Analyse und Spezifikation des Abrechnungsmanagements von Dienstleistungen eines Rechnernetzes*. Dissertation, Technische Universität München, August 1997.

[SGFR 01] STILLER, B., J. GERKE, P. REICHL und P. FLURY: *Management of Differentiated Services Usage by the Cumulus Pricing Scheme and a Generic Internet Charging System*. In: PAVLOU, G., N. ANEROUSIS und A. LIOTTA (Herausgeber): *Proceedings of the 7th International IEEE/IFIP Symposium on Integrated Network Management (IM 2001)*, Seiten 93–106, Seattle, Washington USA, Mai 2001. IEEE Publishing.

[Sido 98] SIDOR, D. J.: *TMN Standards: Satisfying Today's Needs While Preparing for Tomorrow*. IEEE Communications Magazine, 36(3):54–64, März 1998.

[Song 99] SONGHURST, D.J. (Herausgeber): *Charging Communication Networks – From Theory to Practice*. Elsevier Science B.V., Erste Auflage, 1999, `http://www.elsevier.nl/locate/isbn/0-444-50275-0`.

[SRGF 01] STILLER, B., P. REICHL, J. GERKE und P. FLURY: *A Generic and Modular Internet Charging System for the Cumulus Pricing Scheme*. Journal of Network and Systems Management, 9(3):293–325, September 2001.

[SRL 01] STILLER, B., P. REICHL und S. LEINEN: *A Practical Review of Pricing and Cost Recovery for Internet Services*. In: *Netnomics – Economic Research and Electronic Networking*, Band 3. Baltzer, The Netherlands, März 2001.

[TINA 94] FUENTE, L. A. DE LA und T. WALLES: *Management Architecture*. Version 2.0, Telecommunications Information Networking Architecture Consortium, Dezember 1994.

[TINA 96] HAMADA, T.: *Accounting Management Architecture*. Version 1.3 (Draft), TINA Consortium, Februar 1996.

[TINA 97] CHRISTIANSEN, L.: *Service Architecure*. Version 5.0, TINA Consortium, Juni 1997.

[TR 101 734] *Internet Protocol (IP) based networks; Parameters and mechanisms for charging*. ETSI Technical Report TR 101 734 V1.1.1, European Telecommunications Standards Institute, September 1999.

[UML 1.4] *OMG Unified Modeling Language Specification, Version 1.4*. Technischer Bericht formal/2001-09-67, Object Management Group, September 2001, `http://www.omg.org/cgi-bin/doc?formal/01-09-67`.

[Wald 02] WALDBUSSER, STEVEN: *Application Performance Measurement MIB*. Internet Draft, Internet Engineering Task Force (IETF), April 2002, `http://www.normos.org/ietf/draft/draft-ietf-rmonmib-apm-mib-07.txt`.

[Webe 00] WEBER, R.: *Accounting and Payment Concepts for Fee-Based Scientific Digital Libraries*. Dissertation, Technische Universität München, November 2000.

[Wies 94] WIES, R.: *Policies in Network and Systems Management — Formal Definition and Architecture*. Journal of Network and Systems Management, 2(1):63 – 83, 1994. M. Malek (edt.), Plenum Publishing Corp., New York.

[Wies 95] WIES, R.: *Policies in Integrated Network and Systems Management: Methodologies for the Definition, Transformation, and Application of Management Policies*. Dissertation, Ludwig-Maximilians-Universität München, Juni 1995.

[XMLS-0] FALLSIDE, D. C. (EDITOR): *XML Schema Part 0: Primer*. W3C Recommendation REC-xmlschema-0-20010502, World Wide Web Consortium (W3C), Mai 2001, `http://www.w3.org/TR/xmlschema-0/`.

[XMLS-1] THOMPSON, H. S., D. BEECH, M. MALONEY und N. (EDS.) MEHDELSOHN: *XML Schema Part 1: Structures*. W3C Recommendation REC-xmlschema-1-20010502, World Wide Web Consortium (W3C), Mai 2001, `http://www.w3.org/TR/xmlschema-1/`.

[XMLS-2] BIRON, P. V. und A. (EDS.) MALHOTRA: *XML Schema Part 2: Datatypes*. W3C Recommendation REC-xmlschema-2-20010502, World Wide Web Consortium (W3C), Mai 2001, `http://www.w3.org/TR/xmlschema-2/`.

 Als Buch erhältlich im Verlag Dr. Hut, München, www.dr.hut-verlag.de (ISBN 3-934767-97-4)

INDEX

 Als Buch erhältlich im Verlag Dr. Hut, München, www.dr.hut-verlag.de (ISBN 3-934767-97-4)