MIAS: Management Infrastruktur für agentenbasierte Systeme

vorgelegt von
Diplom-Informatiker
Jan Keiser

Von der Fakultät IV – Elektrotechnik und Informatik
der Technischen Universität Berlin
zur Erlangung des akademischen Grades

Doktor der Ingenieurwissenschaften
– Dr.-Ing. –

genehmigte Dissertation

Promotionsausschuss:

Vorsitzender:	Prof. Dr. Klaus-Robert Müller
Berichter:	Prof. Dr.-Ing. habil. Sahin Albayrak
Berichter:	Prof. Dr. rer. nat. habil. Dr. h. c. Alexander Schill

Tag der wissenschaftlichen Aussprache: 25. September 2008

Berlin 2008

D 83

Vorwort

Diese Arbeit entstand während meiner Zeit als wissenschaftlicher Mitarbeiter am DAI-Labor. Dies gab mir die Gelegenheit mich mit vielen Kollegen über das Thema meiner Arbeit auszutauschen. Bedanken möchte ich mich insbesondere bei Prof. Dr. Albayrak, der mir die Gelegenheit zur Anfertigung der Dissertation gab und mich bei meiner Arbeit unterstützte. Ebenfalls bedanken möchte ich mich bei Dr. Benjamin Hirsch, der mich in der letzten Phase der Arbeit nochmals motiviert und vorangetrieben hat sowie für inhaltliche Diskussion jederzeit Bereitschaft zeigte, und bei meiner Familie, die mir immer wieder Kraft und Gelegenheit gab mich auf die Arbeit zu konzentrieren. Schließlich geht mein Dank auch an alle Mitglieder des Kompetenzzentrums Agententechnologie des DAI-Labors für das gemeinsame Interesse an dieser Technologie und an alle übrigen Mitarbeiter für die gute Zusammenarbeit.

Jan Keiser

Berlin, Februar 2008

Kurzfassung

Unter dem Begriff „Managementinfrastruktur“ wird das Zusammenspiel verschiedener Mechanismen verstanden, die zur Verwaltung und Administration von Komponenten, Diensten sowie deren Anbieter und Nutzer innerhalb von verteilten und offenen Systemen notwendig sind. Diese Arbeit konzentriert sich vor allem auf das Management von Multiagentensystemen, die einige Besonderheiten gegenüber klassischen verteilten Systemen aufweisen. Ausgehend aus der durch ein Szenario verdeutlichten Problemstellung und dem aktuellen Stand der Entwicklung wird ein Konzept einer Managementinfrastruktur für agentenbasierte Systeme (MIAS) vorgestellt und am Beispiel der Agentenentwicklungsumgebung JIAC implementiert. MIAS besteht sowohl aus wiederverwendbaren Basismechanismen als auch aus höherwertigen Managementfunktionen beispielsweise aus den Bereichen Abrechnungs- und Konfigurationsmanagement, die ebenfalls im Rahmen dieser Arbeit entwickelt wurden. Abschließend werden weitere Trends im Bereich des Managements offener und verteilter Diensteumgebungen aufgezeigt.

Abstract

A management infrastructure is a combination of different mechanisms for monitoring, control and administration of components, services as well as their provider and user within open distributed systems. This work concentrates on the management of multi agent systems, which have some special characteristics compared with traditional distributed systems. Based on a scenario-specific problem description and the current state of the art of management and agent technologies, a concept of a management infrastructure for agent-based systems (MIAS) is given and implemented with and for the agent framework JIAC. MIAS consists of reusable basic management mechanisms and higher-value management functions for example the accounting and configuration management, which were also developed. Finally, an outlook about future work on management of open distributed service environments is given.

Inhaltsverzeichnis

Abbildungsverzeichnis

Tabellenverzeichnis

Teil I

Einleitung

Kapitel 1
Einleitung und Problemstellung

Um die Akzeptanz agentenbasierter Systeme in der Industrie zu erhöhen, ist eine Managementinfrastruktur für solche Systeme dringend erforderlich. Diese Aussage wird nun etwas genauer erläutert bevor anhand eines Beispielszenarios aus dem Bereich der Verkehrstelematik die Anforderungen an das Management verdeutlicht werden. Aus den genannten Anforderungen lassen sich anschließend auch die Ziele dieser Arbeit ableiten.

1.1 Motivation

Die Agententechnologie ist noch weitgehend im Forschungsstadium wird aber laut AgentLink Roadmap [LMSW05] in naher Zukunft enorm an Bedeutung gewinnen. Ein wesentlicher Grund liegt in der Flexibilität, Kooperationsfähigkeit und Autonomie der Agenten, mit denen Systeme auf einer höheren Abstraktionsebene und somit schneller und effizienter entwickelt werden können. Dadurch ist die hohe Komplexität und Verteiltheit zukünftiger Anwendungen besser in den Griff zu bekommen. Nachteil ist allerdings, dass solch große und hoch-dynamische Systeme schwer zu testen sind und noch keine formalen Methoden und automatisierten Verifikationstechniken in ausreichendem Maße existieren.

Auch wenn Agenten ihre Ziele selbständig und autonom verfolgen können, ist es nicht zuletzt für kritische Prozesse notwendig, dass der Betreiber des Systems in der Lage ist rechtzeitig zu erkennen, wenn das Agentensystem oder Teile davon nicht mehr in seinem Sinne handeln. In solchen Fällen

müssen geeignete Maßnahmen eingeleitet werden können, um größeren Schaden zu verhindern. Eine Verselbständigung von Agentensystemen in Form einer Anarchie oder Anomie ist nicht wünschenswert und sollte verhindert werden können. D.h. Agenten müssen entweder eine Schnittstelle zur direkten Einflussnahme bereitstellen oder zur Einhaltung von Regeln und Normen gezwungen bzw. durch entsprechende Sanktionen dazu angehalten werden können. Bei Letzterem handelt es sich um ein eigenständiges und äußerst komplexes Forschungsgebiet, das in dieser Arbeit nicht weiter verfolgt wird. Für den industriellen und kommerziellen Einsatz sind also umfangreiche Managementfunktionen notwendig, die es ermöglichen das Agentensystem sowohl manuell als auch automatisiert zu überwachen und an geänderte Bedürfnisse anzupassen. Für das Management selbst ist ebenfalls eine höhere Abstraktionsebene vorzusehen, die komplexe Managementabläufe in die elementaren Aktionen auf Agentenebene herunterbricht. Insbesondere in einer offenen und heterogenen Umgebung muss von den konkreten Agentenarchitekturen abstrahiert werden können.

Soll mit dem Agentensystem zudem die Bildung von elektronischen Dienstemärkten unterstützt werden, ist u.a. eine Bedarfsanalyse und individuelle Abrechnung von Diensten unter Einbeziehung weiterer Anbieter wichtig. Auch die Bereitstellungszeit und Qualitätssicherheit von Diensten sind wesentliche Kriterien für die Akzeptanz solcher Umgebungen. Ein Markt kann hierbei als Ort des Austausches von Gütern und Leistungen verstanden werden, an dem sich durch Aufeinandertreffen von Angebot und Nachfrage die Preise bilden. Die mit den ökonomischen Prozessen verbundenen Aktivitäten sind beispielsweise im sogenannten Markttransaktionsmodell zusammengefasst [SL97], das eine Informations-, Verhandlungs- und Abwicklungsphase beschreibt. Diese Phasen sollten einen möglichst hohen Automatisierungsgrad aufweisen.

An einem Dienstemarkt sind zudem nicht nur einzelne Anbieter sondern unterschiedliche Rollen und Akteure vorhanden, die an der Bereitstellung und Nutzung von Diensten gemäß einer Wertschöpfungskette bzw. eines Geschäftsmodells beteiligt sind. Hierzu gehören u.a. die Kunden bzw. Endnutzer, die Dienstanbieter bzw. Händler, die Vermittler, die Anbieter von Dienstinhalten und bei elektronischen Märkten zudem Plattformanbieter und Netzbetreiber. Für alle Beteiligten sind entsprechende Managementfunktionalitäten bereitzustellen, mit deren Hilfe die Dienste administriert werden können. Beispielsweise muss der Kunde nach verfügbaren Diensten suchen und über diese erfahren können, ob ein anonymer Zugriff erlaubt bzw. ein Subscriben notwendig ist und wie der Dienst konfiguriert werden kann. Der Status und die Performanz von Dienstnutzungen sollten kontrollierbar und Dienstaufrufe auch widerrufbar sein. Anbieter von Diensten hingegen benöti-

gen Unterstützung für die reibungslose Inbetriebnahme, den fehlerfreien Betrieb, die schnelle und flexible Erweiterbarkeit sowie für die kostentransparente und qualitätssichernde Erbringung von Diensten.

1.2 Beispielszenario

Mit Hilfe eines Szenarios[1] aus dem Bereich der agentenbasierten Verkehrstelematik [ABK+98] werden nun die Notwendigkeit einer Managementinfrastruktur sowie einige Anforderungen an das Management verdeutlicht. Dieses Szenario ist eine Anwendung bestehend aus mehreren Agenten, die auf folgende Plattformen[2] verteilt sind:

- *Globale Plattformen*: Auf einer globalen Plattform werden verschiedene Verkehrstelematikdienste (z.B. Notruf, Pannenhilfe, Fahrzeugdiagnose, Zielführung, Tankstellensuche und Auskunftsdienste) angeboten, die zur Nutzung in ein Fahrzeug geladen werden können. Dort halten sich ebenfalls global agierende Anbieter auf, die entsprechende Inhalte oder Leistungen für diese Dienste liefern.

- *Regionale Plattformen*: Auf einer regionalen Plattform befinden sich Anbieter von Dienstinhalten oder -leistungen, die auf eine bestimmte Region beschränkt sind (z.B. die Berliner Feuerwehr).

- *Plattform in den Fahrzeugen*: Die in einem Fahrzeug vorhandene Plattform ermöglicht die Suche, das Laden und die Ausführung beliebiger Verkehrstelematikdienste. Diese Plattform bietet auch den Zugriff auf die vorhandenen Fahrzeugsensoren (z.B. Tankfüllung, Ölstand, Geschwindigkeit, Airbagsensoren und GPS-Empfänger).

- *Plattformen in den Leitstellen*: Die an den Notruf- und Pannenhilfediensten beteiligten Anbieter betreiben für die Verwaltung ihrer Leitstellen und ihres Fuhrparks sowie die Abarbeitung von Aufträgen eigene Plattformen.

- *Sonstige Plattformen*: Auf den sonstigen Plattformen werden weitere Leistungen angeboten, die aus technischen oder organisatorischen

[1] Dieses Szenario stammt ursprünglich aus dem Projekt „IA-Verkehr“ am DAI-Labor
[2] Eine Agentenplattform ist eine Laufzeitumgebung für Agenten.

Gründen nicht auf den globalen oder regionalen Plattformen laufen (z.B. eine Verkehrsmanagementzentrale).

Die zu entwickelnde Managementschnittstelle muss von konkreten Agentenarchitekturen abstrahieren können, da beispielsweise Agenten auf der Plattform der Leitstellen nicht auf der gleichen Architektur aufbauen müssen wie die zugehörigen Anbieteragenten auf der globalen oder regionalen Plattform. Ein Pannenhilfeanbieter sollte aber alle seine Agenten über eine einheitliche Schnittstelle administrieren können. Außerdem sollten vorwiegend höherwertige Managementfunktionen verfügbar sein, um die Komplexität, den Aufwand und die Fehleranfälligkeit für den Administrator zu reduzieren.

Zu den gewünschten höherwertigen Managementfunktionen gehört u.a. die Abrechnung von Diensten. So möchte beispielsweise der Verkehrstelematikanbieter für die Deckung seiner Kosten für die Bereitstellung seiner innovativen Dienste und den damit verbundenen globalen und regionalen Plattformen eventuell ein Nutzungsentgeld verlangen. Dabei müssen möglichst beliebige Geschäftsmodelle und Tarifmodelle abgebildet werden können, die nicht nur auf der Häufigkeit oder Dauer der Nutzung basieren.

Eine weitere wichtige Managementfunktion ist die Vermeidung bzw. Behandlung von Fehlern und Qualitätseinbußen. Ein Notrufanbieter muss beispielsweise sicherstellen können, dass eine geeignete Hilfe innerhalb einer vorgegebenen Zeit am Unfallort eintrifft. Ist dies nicht möglich sind entsprechende Maßnahmen (z.B. das Zurückgreifen auf Fremdanbieter) einzuleiten. Dies hat wiederum Auswirkungen auf die Abrechnung der Dienste, indem auf eine nicht oder nur teilweise erbrachte Leistung kein oder weniger Entgeld verlangt werden kann.

Für die Änderung von Tarifmodellen und die Einleitung von Maßnahmen zur Einhaltung der Dienstqualität bedarf es einer Möglichkeit zur einfachen Konfiguration des Systems. So müssen beispielsweise neue Ressourcen hinzugefügt und fehlerhafte Ressourcen ersetzt werden können. Das Gesamtsystem sollte dabei offen gestaltet sein, so dass auch neue Anbieter jederzeit einfach und schnell integrierbar sind.

In einer kommerziellen oder kritischen Umgebung sind Sicherheitsmechanismen notwendig, die es erlauben die Konfigurationsmöglichkeiten des Systems oder den Zugriff auf Dienste einzuschränken. Beispielsweise sollten Verkehrstelematikdienste nur von den Fahrzeugführern geladen werden können, deren Identität eindeutig festgestellt wurde und die aufgrund eines abgeschlossenen Vertrages dazu berechtigt sind.

Aus Sicht der Dienstnutzer ist es wichtig ihr Profil bzw. ihre Präferenzen definieren zu können. So muss beispielsweise bei der Auswahl und Aushandlung des geeigneten Pannenhilfeanbieters berücksichtigt werden, ob eine möglichst schnelle oder preiswerte Hilfe gewünscht ist. Auch der Typ des verwendeten Fahrzeuges und eine möglicherweise vorhandene Diagnose spielt dabei eine gewisse Rolle.

Das Szenario zeigt, dass eine Managementinfrastruktur eine unabdingbare Voraussetzung für einen reibungslosen Ablauf einer agentenbasierten Anwendung ist. Die Infrastruktur stellt eine der Anwendung übergeordnete Instanz dar, die immer wieder in das Geschehen eingreift, um es dem Ziel näherzubringen und dieses Geschehen durch Speichern von Prozessdaten etc. besser überschaubar und nachvollziehbar zu machen.

1.3 Zielsetzung

Ziel dieser Arbeit ist es eine Managementinfrastruktur zu entwickeln mit der agentenbasierte Systeme möglichst flexibel und auf hoher Abstraktionsebene überwacht und gesteuert werden können. Dafür notwendig sind einerseits Basismechanismen, die auf entsprechende Funktionen der verwendeten Agentenarchitektur zugreifen, und andererseits höherwertige Managementfunktionen, die auf diesen Basismechanismen basieren. Die Managementfunktionen sollten nicht auf eine bestimmte Agentenarchitektur ausgerichtet sondern so generisch gehalten sein, dass sie für möglichst verschiedene Architekturen einsetzbar sind oder zumindest leicht angepasst werden können. Hierbei gilt es vorhandene Standards mit zu berücksichtigen.

Die Praktikabilität der zu entwickelnden Basismechanismen wird exemplarisch durch die Bereitstellung von höherwertigen Managementfunktionen aus den Bereichen Abrechnungs- und Konfigurationsmanagement aufgezeigt. Eine spätere Erweiterbarkeit um zusätzliche Managementbereiche soll selbstverständlich möglich sein. Mit dem Abrechnungsmanagement soll eine flexible und zugleich einfache Abrechnung agentenbasierter Dienste möglich sein, mit dem beliebige Geschäftsmodelle abgedeckt werden können. Das Konfigurationsmanagement soll eine Anpassung des Systems unter Berücksichtigung vorhandener Abhängigkeiten und notwendiger Bedingungen erlauben. Zu den allgemeinen Anforderungen an alle Managementbereiche gehören sowohl die Skalierbarkeit der Lösung als auch die Berücksichtigung unterschiedlicher administrativer Domänen sowie Benutzerklassen und Rollen.

Für die Realisierung eines den genannten Anforderungen Rechnung tragendes Rahmenwerk für das Management agentenbasierter Systeme mit ihren Plattformen, Agenten und Diensten wird neben dem theoretischen Entwurf der Managementinfrastruktur auch die prototypische Implementierung am Beispiel einer auszuwählenden Agentenarchitektur erfolgen. Als Laufzeitumgebung für die Managementdienste bieten sich wie auch für die Anwendungsdienste die Agentenplattformen an. Gleichfalls ist durch die Übernahme dieses Konzeptes auch eine Vermittlung sowie Vermarktung der Managementdienste möglich.

1.4 Aufbau der Arbeit

Der Rest dieser Arbeit besteht im Wesentlichen aus zwei großen Teilen.

Im Teil „Technologien“ wird der aktuelle Stand aus den Bereichen Management und Agententechnologie vorgestellt. Aus dem Bereich Management werden vor allem Basistechnologien als auch Technologien zum Abrechnungsmanagement untersucht. Zur Agententechnologie werden erst eine mögliche Definition für Agenten gegeben sowie vorhandene Standards und verschiedene Agentenmodelle beschrieben bevor einige konkrete Agentenentwicklungsumgebungen vorgestellt und die Grenzen der Agententechnologie diskutiert werden. Anschließend erfolgt eine kurze Bewertung der existierenden Ansätze in Bezug auf die in dieser Arbeit gesetzten Ziele.

Der Teil „Konzeption“ verfeinert die bereits in [FBK+01] sehr oberflächlich dargestellte Managementinfrastruktur MIAS und beschreibt zuerst das Grobkonzept bestehend aus Anwendungsfällen, einer Architektur und Ontologien. Anschließend werden die Basismechanismen der Infrastruktur (z.B. Introspektion und Manipulation) genauer betrachtet bevor im Detail auf die Realisierung der höherwertigen Managementfunktionen am Beispiel des Abrechnungs- und Konfigurationsmanagements eingegangen wird. Abschließend erfolgt eine Zusammenfassung der geleisteten Arbeit und eine Bewertung der entwickelten Lösung gegenüber anderen Managementansätzen und den zu Beginn der Arbeit gesetzten Zielen.

Teil II

Technologien

Kapitel 2
Managementtechnologien

Unter Management wird im weitesten Sinne die Betreuung oder Leitung eines Systems über dessen gesamten Lebenszyklus hinweg verstanden. Meist werden aber die Entwicklungsphasen (Anforderungsdefinition, Entwurf, Implementierung) vernachlässigt und sich mehr auf die nachfolgenden Phasen (Auslieferung, Betrieb, Optimierung, Ablösung) konzentriert. Ein Überblick über Konzepte und Techniken zum Management von IT-Infrastrukturen wird in [Bon02] gegeben. Das wohl bekannteste Werk ist die Information Technology Infrastructure Library (ITIL), die auf vorhergehenden Arbeiten von IBM [Sch85] beruht. Der Kern des in dieser Arbeit betrachteten Managementbegriffs umfasst im Wesentlichen die Überwachung und Steuerung des Systems zur Laufzeit, mit dem Ziel einen möglichst optimalen Betrieb zu gewährleisten.

Bei der Untersuchung des Standes der Technik bezüglich des Managements verteilter Systeme werden zuerst in Abschnitt 2.1 die Grundlagentechnologien betrachtet. Dabei werden exemplarisch die drei wichtigsten Standardisierungen aus dem Bereich des Netzwerkmanagements beschrieben, auf denen konkrete problemspezifische Managementlösungen aufbauen können.

Da im Rahmen von MIAS auch höherwertige Managementfunktionen am Beispiel des Abrechnungs- und Konfigurationsmanagements realisiert werden, erfolgt zusätzlich eine Betrachtung des Standes der Technik zum Abrechnungsmanagement, das neben dem Gesamtmodell und den Basismechanismen einen Schwerpunkt dieser Arbeit darstellt. In Abschnitt 2.2 werden die Spezifikationen und Standardisierungsbemühungen von acht ausgewählten Organisationen untersucht.

2.1 Grundlagen

Wesentlich für das Management verteilter Systeme sind die zur Zeit verfügbaren Standards im Bereich des Netzwerkmanagements. Dabei handelt es sich im wesentlichen um die ISO/OSI Standards, die ITU Empfehlungen zum TMN sowie die RFCs der IETF zum Management von Netzwerkkomponenten. Alle drei Ansätze werden nun genauer untersucht (vgl. [STGB98]).

2.1.1 OSI-Management

Das von der International Organization for Standardization (ISO)[1] entwickelte OSI (Open Systems Interconnection)-Basis-Referenzmodell (BRM) ist eine Grundlage für die Entwicklung von Standards für die Verbindung offener Systeme [IT94]. Aufbauend auf diesem Modell wird im folgenden ein Überblick über das OSI-Management gegeben.

Das OSI-Management beschreibt wie Benutzer und Anbieter von Kommunikationsdiensten die entsprechenden Aktivitäten innerhalb der offenen Systeme planen, überwachen und kontrollieren können. Hierfür werden Mechanismen bereitgestellt, die die Koordination, Verteilung und Zuweisung der in den Kommunikationsnetzen verwendeten Ressourcen ermöglicht. Es handelt sich dabei um ein Rahmenwerk für die Entwicklung von Managementdiensten in einer OSI-Umgebung, das basierend auf dem OSI-BRM die Struktur von verteilten Managementsystemen sowie deren Interaktion definiert. Hierbei sind vor allem ein objektorientierter Ansatz zur Strukturdefinition sowie entsprechende Dienste und Protokolle für den Austausch von Managementinformationen zu nennen. Die systeminterne Verarbeitung der Managementinformationen ist nicht Bestandteil des OSI-Managements.

[1] http://www.iso.org/

Standardisierungsbereiche

Das OSI-Management unterscheidet folgende drei Bereiche (vgl. [KHA07]):

- Das *Systemmanagement* umfasst die Überwachung, Kontrolle und Koordination eines Systems über alle sieben OSI-Schichten[2] hinweg und ist auf der Anwendungsebene angesiedelt.
- *(N)-Layer-Management* definiert Funktionen, Dienste und Protokolle für eine bestimmte Schicht.
- *(N)-Layer-Operation* beschreibt Mechanismen zur Überwachung und Kontrolle von Verbindungen innerhalb einer bestimmten Schicht. Die notwendige Kommunikation zwischen den steuernden Instanzen erfolgt über bereits definierte Protokolle.

Die für OSI-Management zuständige Arbeitsgruppe der ISO definiert nur Managementstandards zum Systemmanagement [IT98c], dem sich wiederum folgende Standardisierungsbereiche zuordnen lassen:

- *Funktionsmodell*: Es werden verschiedene Managementprozeduren spezifiziert, die jeweils Bezug auf die verschiedenen Specific Management Functional Areas (SMFAs) nehmen (siehe auch FCAPS).
- *Informationsmodell*: Die Syntax und Semantik der Managementinformationen ist in den Standards unter Structure of Management Information (SMI) [IT93, IT92a, IT92b] beschrieben.
- *Organisationsmodell*: Der verteilte Charakter des Managements ergibt sich aus den Beziehungen zwischen dem verwalteten System, den verschiedenen Bereichen, sowie den Manager- und Agentenprozessen.
- *Kommunikationsmodell*: Dienste und Protokolle zur Übermittlung von Managementinformationen sind in Common Management Information Service (CMIS) [IT98b] und Common Management Information Protocol (CMIP) [IT98a] definiert.

Bevor diese vier Teilmodelle genauer erläutert werden, erfolgt anhand von Abbildung 2.1 eine kurze Darstellung der OSI-Standarddokumente und deren Zusammenhang.

[2]Die sieben OSI-Schichten sind: Anwendungs-, Präsentations-, Kommunikationssteuerungs-, Netzwerk, Verbindungs- und die physikalische Schicht.

Als vierter Teil des OSI-Referenzmodells bildet das Managementrahmenwerk [IT89] die Grundlage des OSI-Managements, das in System Management Overview [IT98c] detailliert beschrieben ist. Diese Dokumente setzen das Managementinformationsmodell [IT93] und verschiedene Systemmanagementfunktionen voraus.

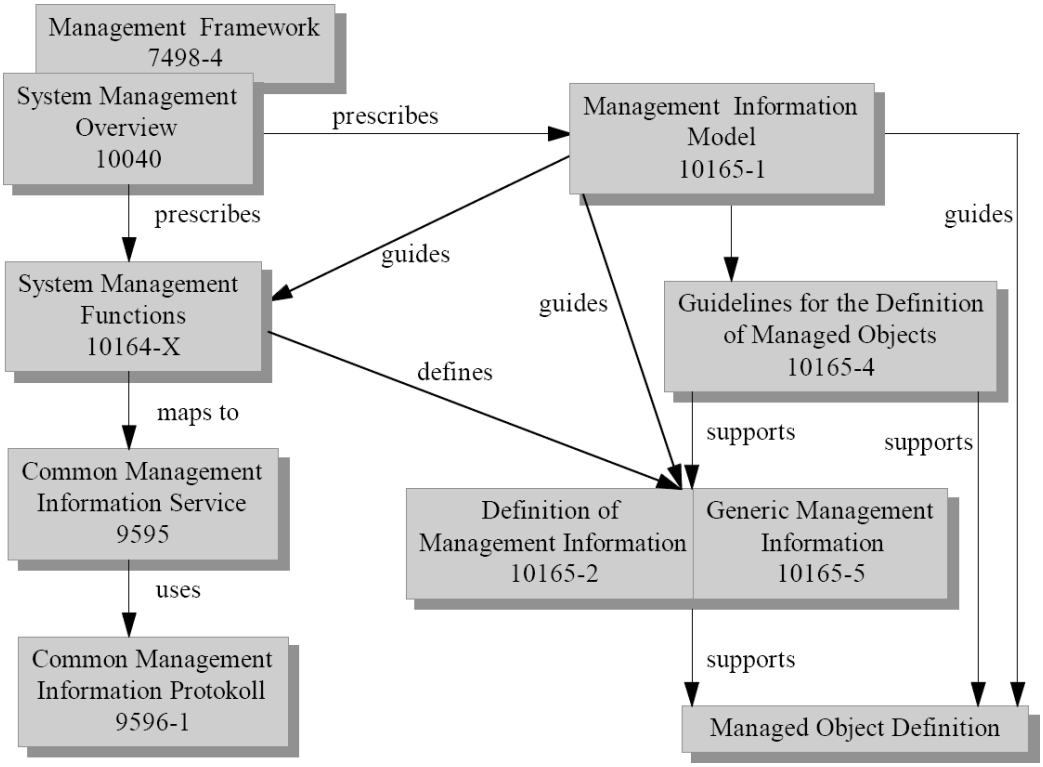

Abbildung 2.1: Standards zum OSI-Management [STGB98]

Alle Managementinformationen eines OSI-Systems werden in einer objektorientierten Management Information Base (MIB) [IT93] abgelegt. Die Repräsentation der zu verwaltenden Ressourcen erfolgt dabei über Managed Objects (MOs) [IT92a], die jeweils durch eine Klasse und eine Reihe von Attributen definiert sind [IT92b]. System Management Application Entities (SMAEs) regeln den Zugriff auf die MIB. Die Managementdienste und der zugehörige Datenaustausch basieren auf CMIS und CMIP. Die Managementfunktionen selbst sind in [IT] beschrieben.

Sämtliche OSI-Managementstandards wurden von Comité Consultatif International Téléphonique et Télégraphique (CCITT) angepasst und in die X.700 Reihe übernommen, die die Grundlage für das Telecommunication Management Network (TMN) (siehe Abschnitt 2.1.2) bilden.

Funktionsmodell

Das OSI-Systemmanagement unterscheidet fünf Bereiche denen Managementfunktionen zugeordnet sein können. Diese SMFAs, die auch unter dem Begriff FCAPS bekannt sind, werden nun kurz erläutert:

- *Fault Management (Fehlermanagement)*: Dieser Bereich umfasst alle Aktivitäten, um Störungen rechtzeitig zu erkennen, geeignet darauf zu reagieren und die Ursachen zu bekämpfen. Dazu gehören beispielsweise die Definition von Fehlerklassen, die Überwachung des Systemverhaltens und die Generierung und Weiterleitung von Fehlermeldungen. Durch Erhöhung der Fehlertoleranz kann die Ausfallsicherheit des Systems gesteigert werden.
- *Configuration Management (Konfigurationsmanagement)*: Für die Initialisierung, Kontrolle und Anpassung der Konfiguration des Systems müssen entsprechende Informationen identifiziert, gesammelt und bereitgestellt werden.
- *Accounting Management (Abrechnungsmanagement)*: Zu diesem Bereich gehören sowohl die Bewertung von Diensten und Ressourcen einer Anwendung in Form von Berechnungsvorschriften für deren einzelne oder kombinierte Nutzung als auch die Aufzeichnung von Daten und Verwaltung von Abläufen, die für die Abrechnung nötig sind (z.B. Kunden- bzw. Benutzerverwaltung, Ressourcenverbrauch und Rechnungen). Das Accounting stellt somit eine Voraussetzung für die Umsetzung eines Billingkonzeptes dar. Auch mögliche Aushandlungen von Tarifen, Verbrauchsgrenzen und Abrechnungsvorschriften fallen in diesen Bereich.
- *Performance Management (Qualitätsmanagement)*: Dieser Bereich umfasst vorwiegend die Überwachung von Leistungsparametern des Systems. Dazu gehören beispielsweise die Dauer, Geschwindigkeit oder Verzögerung von Datenübertragungen, die Dauer bestimmter Aktionen und die Auslastung begrenzter Ressourcen. Aber auch die Berechnung geeigneter Anpassungen zur Wahrung der Leistung des Systems ist Bestandteil dieses Bereiches.
- *Security Management (Sicherheitsmanagement)*: Zu diesem Bereich gehören alle Maßnahmen wie Authentifikation und Autorisation, die zur Absicherung der Kommunikation und des Zugriffs auf Ressourcen notwendig sind. Das Erkennen und die geeignete Abwehr von Angriffen auf das System fällt ebenfalls in diesen Bereich.

Diese FCAPS-Definitionen bieten keine klare Abgrenzung bei der Bereitstellung von Managementleistungen, weil für die vollständige Erbringung bestimmter Managementfunktionalitäten eine bereichsübergreifende Betrachtungsweise notwendig ist. Beispielsweise gehört bei einer Fehlerbehebungsmaßnahme die Fehlererkennung zum Fehlermanagement, aber für die eigentliche Fehlerkorrektur ist das Konfigurationsmanagement gefragt, wenn sich die Ursache nur durch eine Rekonfiguration beheben lässt. Deshalb wird basierend auf CMIS/CMIP eine Menge von Managementfunktionen definiert, die verschiedenen Funktionsblöcken zugeordnet sein können. Diese Funktionen werden als Managementdienste in Funktionsbibliotheken zusammengetragen [IT]. Folgende Funktionen sind bisher dem Standardisierungsprozess unterzogen worden: „Object Management“, „State Management“, „Relationship Management“, „Alarm Reporting“, „Event Report Management“, „Log Control“, „Security Alarm Reporting“, „Security Audit Trail“, „Access Control“, „Usage Metering“, „Metric Management“, „Test Management“, „Summarization“, „Confidence and Diagnostic Test“, „Scheduling“, „Management Knowledge Management“, „Changeover“, „Software Management“, „Management Domain and Management Policy Management“, „Time Management“, „Command Sequencer“ und „Response Time Monitoring“.

Informationsmodell

Das Informationsmodell definiert die Struktur von Managementinformationen, wofür folgende Entscheidungen getroffen werden mussten:

- der Modellierungsansatz (z.B. Objektorientierung oder Entity-Relationship-Modelle).
- die Syntax zur Beschreibung der Informationen.
- die Auswahl und Beschreibung der zu verwaltenden Ressourcen (Hard- und Software) des Systems mit der Möglichkeit anwendungsspezifische Erweiterung vornehmen zu können.
- die Gewährleistung einer eindeutigen Abbildung zwischen den Ressourcen und den zugehörigen Beschreibungen.

Das verwendete Konzept zur Modellierung relevanter Ressourcen entspricht einem objektorientierten Ansatz basierend auf Datenabstraktion und Vererbungsmechanismen. Die Managementinformationen werden in Form von

MOs in einer MIB abgelegt. Eine MO-Instanz ist dabei eine konkrete Ausprägung einer Objektklasse, der verschiedene Attribute, Operationen und Benachrichtigungen zugeordnet sein können. Als Notation wird die Abstract Syntax Notation One (ASN.1) verwendet.

Mit Hilfe relativer und absoluter Namen werden MOs eindeutig identifiziert. Relative Distinguish Name (RDN) ist der MO-Name relativ zur Position im Hierarchiebaum der MOs, während sich der absolute MO-Name aus dem absolutem Pfadnamen und dem RDN zusammensetzt. Der absolute Pfadname stellt den Weg von der Wurzel bis zur Position des MO im Baum dar. Für die Erzeugung systemweit eindeutiger RDNs werden Verzeichnisdienste (z.B. X.500) verwendet. Solch ein Verzeichnisdienst kann dabei als ein verteilter Dienst aufgefasst werden, der aus mehreren über das Directory System Protocol (DSP) kommunizierenden Directory System Agenten (DSAs) besteht (siehe Abbildung 2.2).

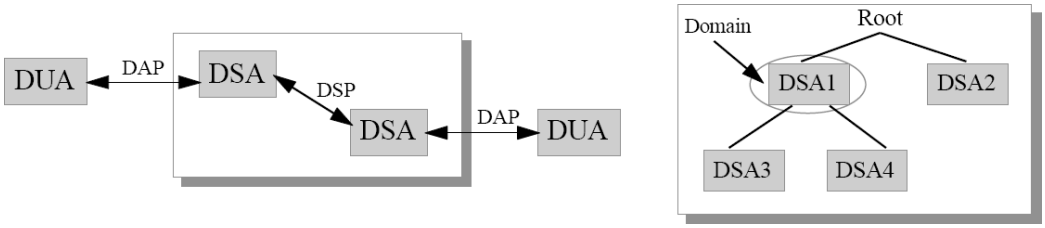

Abbildung 2.2: Verzeichnisdienst [STGB98]

Ein Directory Information Tree (DIT) wird aus einer Menge von DSAs gebildet, die jeweils die für ihre Domäne relevanten Informationen verwalten und diese über das Directory Access Protocol (DAP) beliebigen Directory User Agenten (DUAs) zur Verfügung stellen. Ein DSA soll Anfragen an den zuständigen DSA weiterleiten, wenn das angeforderte Objekt nicht in seiner Domäne enthalten ist. Ist das Objekt im gesamten DIT nicht vorhanden, so erhält der DUA eine entsprechende Fehlermeldung.

Organisationsmodell

Das Oganisationsmodell geht grundsätzlich von einem verteilten kooperativen Management in einem Netz von offenen Systemen aus. Es werden Rollen (Manager, Agent) unterschieden, wobei ein System verschiedene Rollen in Hinblick auf bestimmte Ressourcen einnehmen kann. Ein Manager ruft Managementaktionen auf den zu verwaltenden Systemen (auch Managed Open System genannt) auf, die die Agentenrolle einnehmen. Innerhalb des

Systems werden die Aktionen auf den MOs durch sogenannte Agenten ausgeführt.

Diese auf einer Manager- oder Agentenrolle basierenden Aktivitäten werden als System Management Processes (SMPs) behandelt. Ein zusätzliches Domänenkonzept erlaubt die organisatorische Gruppierung von Managementobjekten.

Kommunikationsmodell

Für den Austausch von Managementinformationen zwischen verschiedenen Systemen sind entsprechende Dienste und Protokolle nötig. Abbildung 2.3 zeigt beispielhaft die Kommunikationsbeziehung zwischen drei Systemen, bei dem System B sowohl die Agenten- als auch die Managerrolle einnimmt.

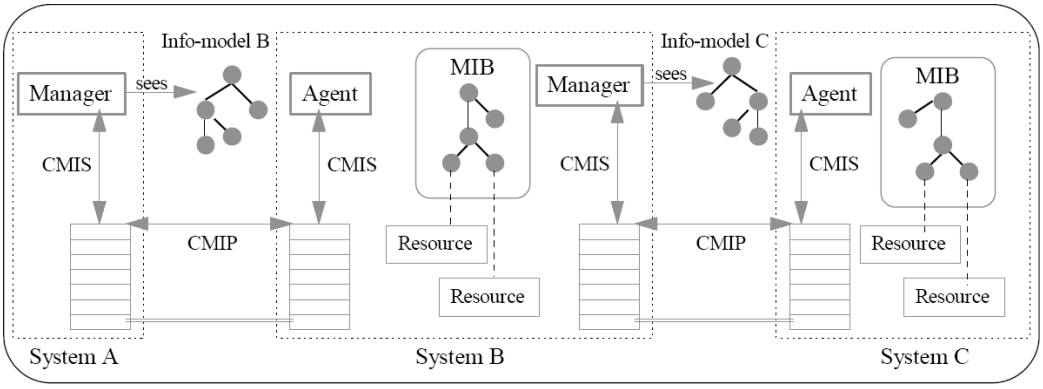

Abbildung 2.3: Managementkommunikation zwischen OSI-Systemen [STGB98]

Die verschiedenen Managementfunktionalitäten werden von den SMAEs bereitgestellt, die aus mehreren Anwendungsdienstelementen (ASEs) bestehen. CMIS hingegen stellt nur Dienste für den Austausch von Managementinformationen zur Verfügung.

Die Common Management Information Service Elemente (CMISE) regeln den Auf- und Abbau von Managementverbindungen zwischen OSI-Systemen sowie die Abwicklung von Managementaufträgen von anderen Systemen. Es existieren zwei Kategorien von Dienstelementen in CMIS:

1. Dienstprimitive zur Benachrichtigung über Ereignisse (Management Notification Service).

- *M-Event-Report* (Anforderungs-, Anzeige-, Antwort- und Bestätigungsprimitive) informiert das Managementsystem über Ereignisse.

2. Dienstprimitive für den Aufruf von Aktionen auf MOs (Management Operation Service):

   - *M-Create* für die Erzeugung von MOs
   - *M-Delete* zum Löschen von MOs
   - *M-Action* zum Ausführen von Aktionen, die für ein MO definiert sind
   - *M-Set* für die Änderung von MO-Attributen
   - *M-Get* für die Abfrage von MO-Attributen
   - *M-Cancel-Get* zum Beenden einer gestarteten M-Get-Operation.

2.1.2 Telecommunication Management Network (TMN)

Die International Telecommunication Union (ITU)[3], die aus CCITT hervorgegangen ist, hat im Jahre 1985 mit der Standardisierung von TMN begonnen und ab 1992 die M-Reihe (Blue Books) veröffentlicht, deren Zusammenhang in Abbildung 2.4 dargestellt ist.

TMN bietet eine Plattform für das integrierte Management von beliebigen Telekommunikationsnetzwerken und Diensten an, die sich stark am OSI-Management (GDMO, CMIS/CMIP, etc.) orientiert. Es wird zwar für die Kommunikation der Managementinformationen ein eigenes Netzwerk vorausgesetzt, das über entsprechende Schnittstellen mit dem zu verwaltenden Telekommunikationsnetzwerk verbunden ist, prinzipiell kann aber auch das Telekommunikationsnetz selbst zur Datenübertragung verwendet werden.

[3] http://www.itu.int/

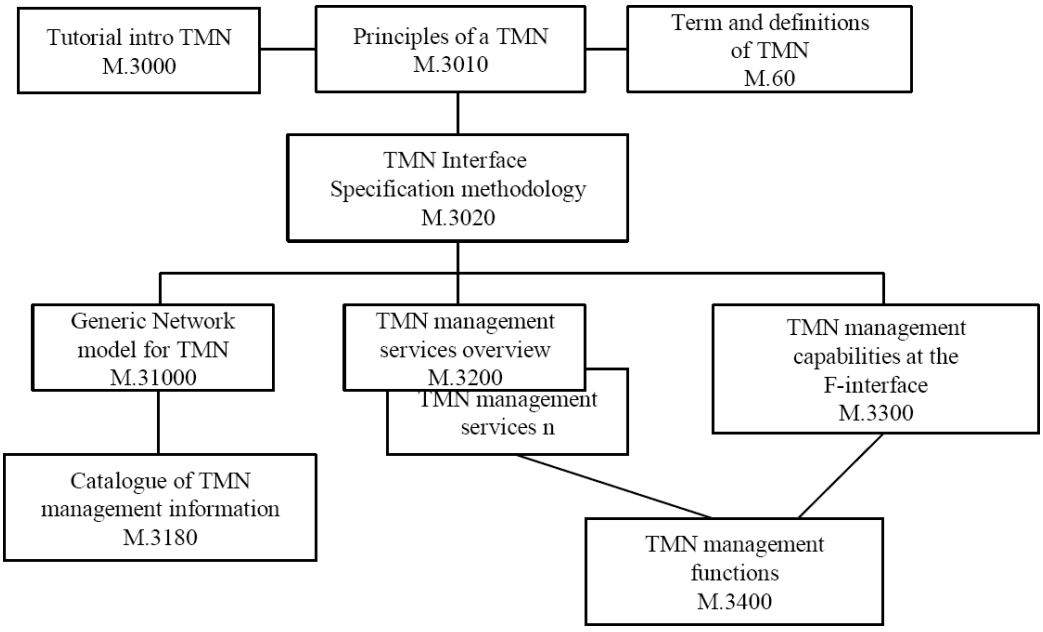

Abbildung 2.4: ITU-Empfehlungen zum TMN [STGB98]

Die TMN-Architektur besteht aus den folgenden drei Ebenen:

- Informationsarchitektur
- Funktionale Architektur
- Physikalische Architektur

Informationsarchitektur

Wie beim OSI-Management wird auch beim TMN basierend auf den X.700 Empfehlungen der objektorientierte Ansatz verwendet. MOs mit entsprechenden Attributen repräsentieren dabei die physikalischen und logischen Netzwerkkomponenten mit ihren Eigenschaften sowie ihre Beziehungen untereinander (siehe Abbildung 2.5). Es ist sogar möglich eine Menge von Ressourcen mit den zugehörigen MOs auf ein MO abzubilden. Mit Hilfe von Managementdomänen kann auch eine Gruppierung von Komponenten vorgenommen werden, der Austausch von Informationen erfolgt aber stets in Form von MOs.

Die Empfehlung Structure of Management (SMI) [IT93] definiert zwar nicht den Inhalt der MOs, aber den funktionalen Aufbau an deren Schnittstelle.

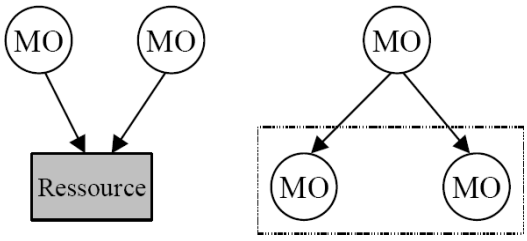

Abbildung 2.5: Beziehung zwischen MOs und Ressourcen [STGB98]

Es ist also festgelegt wie auf die Attribute, die Managementoperationen, das Verhalten und die Benachrichtigungen der MOs zugegriffen werden kann. Die Guidelines for the Definition of Management Objects (GDMO) [IT92b] spezifizieren basierend auf ASN.1 die Anforderungen bezüglich der Syntax. Ähnlich zum OSI-Management erfolgt der Zugriff auf MOs gemäß dem in Abbildung 2.6 dargestellten Client-Server-Ansatz, bei dem ein Manager als Client eine Managementaktion bei einem Agenten als Server anstößt. Nur Benachrichtigungen werden unaufgefordert vom Agenten an den Manager geschickt. CMIP und CMIS definieren auch hier das entsprechende Zugriffsprotokoll und den zugehörigen Dienst. Die ausführbaren Operationen sind bereits durch CMISE des OSI-Managements beschrieben.

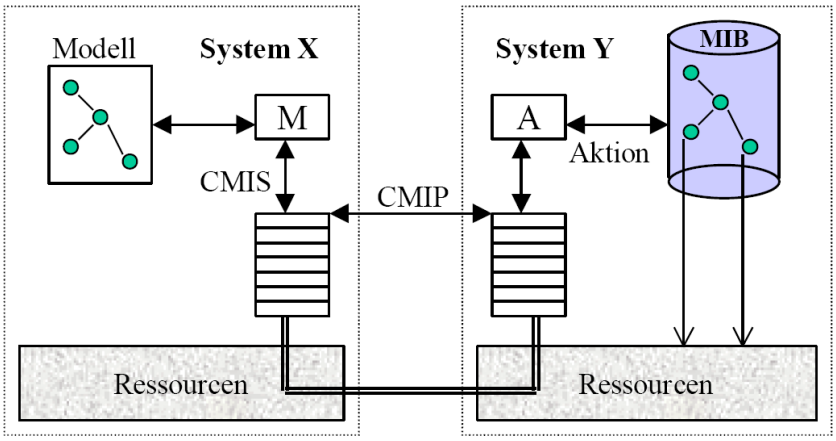

Abbildung 2.6: Informationsstruktur des TMN [STGB98]

Wie bereits erwähnt können MOs beliebigen Managementdomänen zugeordnet werden. Diese Domänen können disjunkt zueinander sein oder gemeinsame Teilmengen von MOs besitzen. Eine Aggregationsbeziehung ist ebenfalls möglich.

Funktionale Architektur

Die funktionale Architektur beschreibt die Verteilung der Managementfunktionalität innerhalb eines TMN. Dabei werden fünf Funktionsblöcke und fünf Referenzpunkte spezifiziert (siehe Abbildung 2.7). Ein Referenzpunkt definiert einen Zugangspunkt zu Funktionsblöcken und eine zugehörige Schnittstelle.

Die Operation Systems Function (OSF) ist für die Ausführung der Managementfunktionen, d.h. die Beobachtung, Koordination und Kontrolle des Netzwerkes und der Dienste zuständig. Der q3-Referenzpunkt erlaubt dabei Aktionen einer Network Element Function (NEF), Q-Adapter Function (QAF) oder Mediation Function (MF) aufzurufen und Benachrichtigungen von den NEFs und MFs zu erhalten. Eine OSF kann wiederum aus mehreren OSFs bestehen. Als Zugangspunkt zu anderen TMN-Systemen steht der von der ITU nicht näher spezifizierte x-Referenzpunkt zur Verfügung.

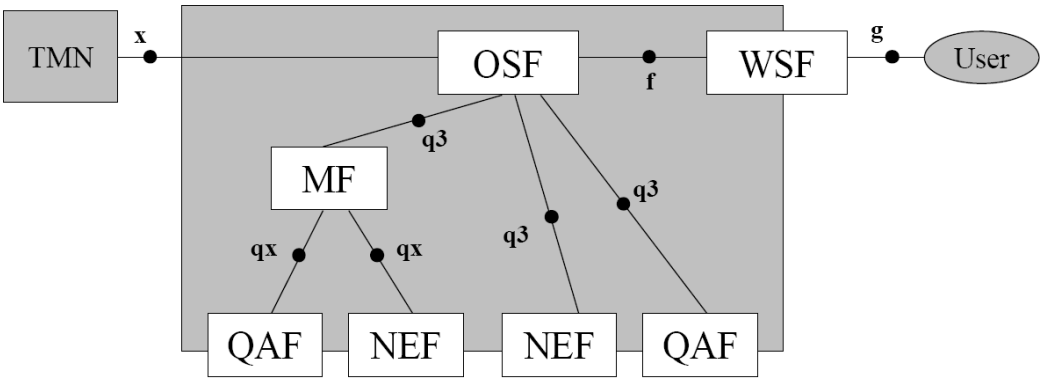

Abbildung 2.7: Funktionale Blöcke des TMN [STGB98]

Eine NEF besteht aus den zu verwaltenden Ressourcen und aus einer Agentenrolle, die Funktionalitäten zur Kommunikation mit der OSF bereitstellt. Eine QAF verbindet das TMN mit anderen Systemen, indem eine Übersetzung zwischen den unterschiedlichen Schnittstellen vorgenommen wird. Eine MF übernimmt die Anpassung, Umformung und Zustellung von Managementinformationen zwischen NEFs/QAFs und einer OSF. Für die Kommunikation mit den NEFs und QAFs steht der qx-Referenzpunkt zur Verfügung.

Operatoren bzw. Administratoren können das TMN nutzen, indem sie auf die Workstation Function (WSF) des Systems über den g-Referenzpunkt zugreifen, der eine von der ITU nicht näher spezifizierte Benutzerschnittstelle darstellt. Eine WSF übernimmt dabei die Kommunikation mit der OSF oder

einer MF über den f-Referenzpunkt und ist in der Lage die Managementinformationen für den Administrator zu interpretieren.

Physikalische Architektur

Die physikalische Architektur definiert unter anderem wie die Funktionsblöcke durch konkrete TMN-Bausteine des Telekommunikationsnetzes realisiert werden können. Die Übertragung der Managementdaten zwischen den einzelnen Funktionsblöcken erfolgt über ein eigenes Data Communication Network (DCN). Nur zwischen der MF und den NEFs/QAFs kann ein Local Communication Network (LCN) in Form eines Subnetzes verwendet werden.

2.1.3 Simple Network Management Protocol (SNMP)

Das SNMP ist eine Weiterentwicklung des Simple Gateway Monitoring Protocol (SGMP) und des anschließenden Simple Management Protocol (SMP), mit dem Ende der achtziger Jahre die Möglichkeit geschaffen werden sollte, das enorm gewachsene Internet besser in den Griff zu bekommen. Aufgrund der laufenden Standardisierungen der ISO sollte es durch ein einheitliches OSI-Managementkonzept ersetzt werden und orientierte sich deshalb wie das TMN ebenfalls an den OSI-Konzepten. Statt der Ersetzung kam es allerdings zur stetigen Weiterentwicklung des SNMP und mit dem SNMPv2 und SNMPv3 wurden somit neue Versionen herausgebracht.

Die Standardisierung des SNMP erfolgt durch die Internet Engeneering Task Force (IETF)[4] in Form von Request for Comments (RFCs). Für das SNMP relevant sind dabei RFC 1155 [RM90] und RFC 1157 [CFSD90], die die Vorgehensweise bei der Modellierung von Managementinformationen definieren, und RFC 1213 [MR91], das für den Austausch dieser Daten einen in Abbildung 2.8 beispielhaft dargestellten Stapel an Kommunikationsprotokollen spezifiziert. Das SNMP besteht konzeptionell aus folgenden Elementen:

- Management Station
- Management Agent
- Management Information Base
- Network Management Protocol

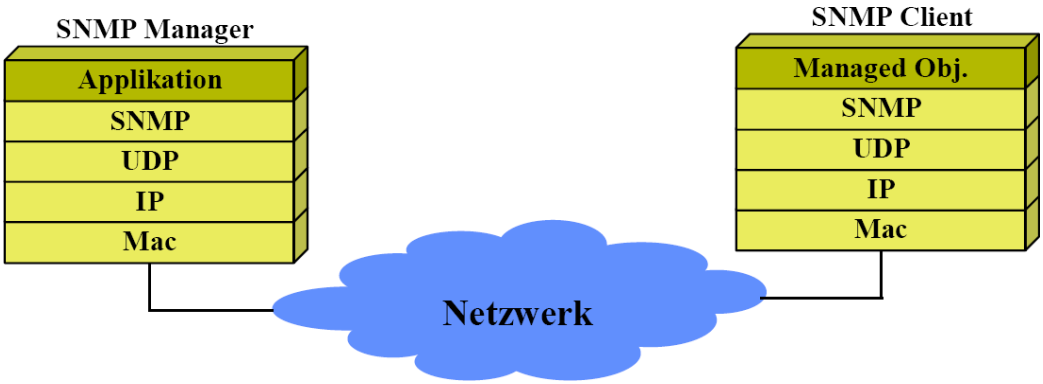

Abbildung 2.8: SNMP Netzwerk [Sta93]

Über die Management Station, auch SNMP-Manager genannt, stehen dem Administrator Managementfunktionalitäten wie Datenanalyse, Fehlerbehebung und Netzüberwachung zur Verfügung. Alle notwendigen Aktionen werden dabei auf den Netzwerkelementen (z.B. Router, Bridges, Workstations) von einem SNMP-Agenten ausgeführt. Die Eigenschaften dieser Elemente werden durch Variablen modelliert, auf die mit folgenden atomaren Aktionen zugegriffen werden kann:

- Get
- Set
- Trap

Die Get-Aktion erlaubt Variablenwerte zu lesen, während mit der Set-Aktion die Werte entsprechend verändert werden. Mit der Trap-Aktion kann ein SNMP-Agent im Fehlerfall unaufgefordert Mitteilungen an den zuständigen SNMP-Manager senden. Dieser Mechanismus wird auch beim trap-directed Polling verwendet, um den Manager von periodischen Anfragen zu entlasten und Netzbandbreite einzusparen.

Das SNMP basiert standardmäßig auf UDP, um eine möglichst schnelle Datenübertragung zu gewährleisten. Die Verwendung von TCP ist aber prinzipiell möglich. Das eigentliche Management des Netzwerkes ist aber auf der Anwendungsebene angesiedelt. Dabei werden die zu verwaltenden Ressourcen durch Objekte mit entsprechenden Attributen modelliert, die in der MIB des

[4]http://www.ietf.org/

jeweiligen SNMP-Agenten gesammelt werden, um den administrativen Zugriff auf die zugehörigen Ressourcen durch den SNMP-Manager zu ermöglichen.

Die Verbindung von SNMP mit proprietären Managementsystemen erfolgt über geeignete Proxies, die als Gateway fungieren.

SNMP Version 2

Aufgrund der geringen Netzbelastung und der leichten Realisierbarkeit wurde bereits die erste Version von SNMP von nahezu allen Netzwerkelementen unterstützt. Mit der Erweiterung des Standards und der damit verbundenen Beseitigung einiger Unzulänglichkeiten entstand eine neue Version, die SNMP Version 2 (SNMPv2).

Die zusätzlich definierte Protokollprimitive „Get-Bulk-Request“ ermöglicht es größere Datenmengen (z.B. komplette Routingtabellen) in einem Schritt abfragen zu können, anstatt eine ganze Reihe von Get-Aktionen ausführen zu müssen. Durch die Get-Aktion „non-atomic“ können bei Angabe mehrerer Variablen auch eine Untermenge und deren Werte übertragen werden.

Für die Verwaltung sehr großer Netzwerke wurden sogenannte Top-level Management Stations eingeführt, denen gesamte Bereiche an Managementagenten zugeordnet sind. Diese Managementagenten werden entweder direkt oder unter Zuhilfenahme von Intermediate-Managern verwaltet, die entsprechend den Vorgaben der Management Station selbständig ihre Managementagenten beobachten und kontrollieren. Zusätzliche Protokollprimitive (z.B. „Inform“) und eine gemeinsam zugängliche Manager-to-Manager MIB unterstützen die Kooperation zwischen den Top-level Management Stations und den Intermediate-Managern.

SNMP Version 3

Von 1996 bis 1998 wurde an der Version SNMPv3 gearbeitet, die verschiedene Sicherheitsmechanismen enthält. In diesem Zusammenhang wurde SNMP in drei Module aufgeteilt:

- Das *Message-Processing and Control Module* ist für die Erzeugung und das Parsen von Managementnachrichten zuständig.
- Das *Local Processing Module* verarbeitet die eintreffenden Informationen und Ereignisse.

- Das *Security Module* stellt Funktionen für Authentifikation und Verschlüsselung bereit und kontrolliert das zeitliche Eintreffen der Nachrichten.

SNMPv3 bietet Authentifikation, Geheimhaltung und Autorisierung als Sicherheitsmechanismen an. Bei der Authentifikation wird festgestellt, ob die empfangene Nachricht tatsächlich von dem angegebenen Manager abgeschickt und zwischenzeitlich nicht verändert wurde. Unter Geheimhaltung verbirgt sich die Verschlüsselung und Entschlüsselung von Nachrichten vor bzw. nach deren Übertragung, um eine Ausspähung der Managementaktionen weitgehend ausschließen zu können. Mit der Autorisierung ist es möglich für jeden Manager individuelle Zugriffsrechte auf die Aktionen der Agenten zu definieren.

2.2 Abrechnungsmanagement

Mit dem Abrechnungsmanagement, auch Accounting Management genannt, wurde sich bisher vorwiegend im Zusammenhang mit dem Schlagwort AAA (Authentifizierung, Autorisierung und Accounting) beschäftigt.

Das Accounting Management umfasst die Sammlung von Daten über Ressourcennutzungen für Kapazitäts- und Trendanalysen, Kostenzuweisungen, Rechnungsprüfungen und Gebührenerfassungen [AAH00]. Hierfür ist neben der Messung, Bewertung und Zuordnung der Ressourcennutzungen auch die Kommunikation der Daten zwischen den entsprechenden Parteien erforderlich. Das Ziel von Kapazitäts- und Trendanalysen sind typischerweise Vorhersagen über zukünftige Nutzungen, die aufgrund der Verwendung von statistischen Verfahren meist sehr ungenau sind. Eine Kostenzuweisung erfolgt gewöhnlich im Bereich der Telekommunikation zwischen Unternehmenspartnern oder Abteilungen eines Unternehmens. Hierfür werden Modelle (z.B. traditionelle und leistungsbezogene Mechanismen) verwendet, die typischerweise auf einer genauen Analyse der Nutzungsdaten basieren. Bei der Rechnungsprüfung wird die Korrektheit von ausgestellten Rechnungen sowie die Erfüllung von Nutzungspolitiken, Dienstgütevereinbarungen und Sicherheitsrichtlinien überprüft. Die Gebührenerfassung kann nutzungsabhängig oder nutzungsunabhängig gestaltet sein, wobei die von Nutzungsinformationen abhängige Gebührenberechnung ein externes Rechnungswesen erfordert.

Zum Thema Accounting Management existieren bereits eine Reihe von Vorschlägen, Spezifikationen und Lösungen, die sich aber zumeist mit dem Accounting auf Kommunikationsebene (Übertragungsdienste bzw. Netzzugriffe) beschäftigen oder nicht alle Aspekte des Accountings abdecken. Die wesentlichen Entwicklungen in diesem Bereich werden nun vorgestellt.

2.2.1 Internet Engineering Task Force (IETF)

Die Internet Engineering Task Force (IETF)[5] gibt einen guten Einstieg in die Thematik des Accountings [MHR91, AAH00] und definiert gleichzeitig eine einfache Accounting Management Architektur (siehe Abbildung 2.9), in der auch die Anforderungen an die Protokolle zwischen Netzwerkgeräten, Accounting- und Billingserver verschiedener Organisationen beschrieben sind. Die bisher am häufigsten verwendeten Protokolle sind Diameter [CLG+03], der Remote Authentication Dial In User Service (RADIUS) [RRSW00], Cisco's proprietäres Terminal Access Controller Access-Control System Plus (TACACS+) [CG97] und das SNMP (siehe Abschnitt 2.1.3).

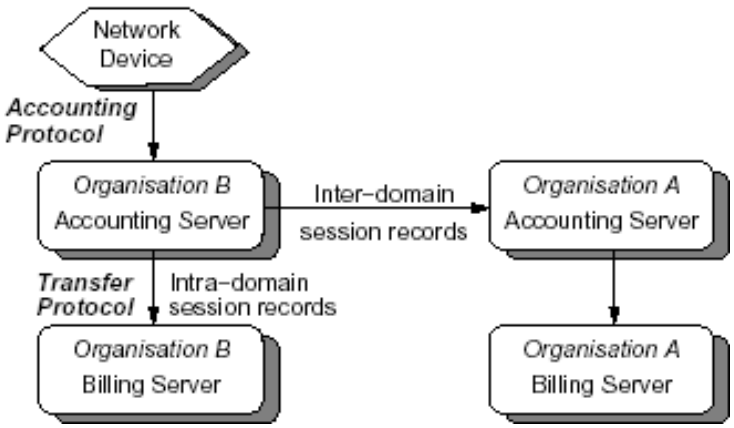

Abbildung 2.9: Accounting Management Architektur der IETF [Rad03]

RADIUS ist ein Protokoll für AAA, das vorwiegend von Netzwerkzugangsservern verwendet wird, um eine Authentifizierung von Benutzern durchzuführen. Dazu wird beim Einloggen des Benutzers eine Anfrage in Form eines UDP-Paketes an den RADIUS-Server gesendet, der im einfachen Fall entweder mit einer Zusage oder einer Ablehnung antwortet. Die Accounting-Erweiterung von RADIUS [Rig00] definiert zusätzlich Pakete, die sowohl zum

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

anschließenden Starten der Abrechnung als auch beim Ausloggen des Benutzers zum Beenden der Abrechnung verwendet werden können. Dabei werden auch Informationen über die Sitzung (z.B. Dauer und übertragene Datenmenge) mitgeteilt.

Als Nachfolger von RADIUS wurde Diameter ebenfalls mit einer Accounting-Erweiterung [ACZ01] entwickelt, das einige Unzulänglichkeiten beseitigt, die aufgrund des starken Wachstums des Internets und der Einführung neuer Zugangstechnologien entstanden sind. Die Vorteile gegenüber RADIUS sind unter anderem (vgl. [ING99]):

- eine deutliche Ausweitung der maximalen Größe eines Paketes,
- die Erkennung von mehrfach übertragenen Paketen über einen deutlich längeren Zeitraum hinweg,
- die Möglichkeit über TCP zu kommunizieren und den Fluss von UDP Paketen zu regulieren,
- eine Empfangsbestätigung durch den Server,
- eine Fehlermeldung, wenn Pakete falsche Informationen enthalten,
- eine Mitteilung, wenn der Server heruntergefahren wird,
- eine Verschlüsselung und Authentifizierung über Proxies hinweg,
- die Möglichkeit zur Definition von benutzerspezifischen Kommandos und
- ein Format für eine effizientere Verarbeitung der Pakete.

Um Nutzungsdaten zwischen den Organisationen austauschen zu können, sind neben Übertragungsprotokollen auch eine Menge von Accountingattributen sowie Datenformate definiert worden [BB00]. Das Accounting Data Interchange Format (ADIF) beschreibt ein erweiterbares und menschenlesbares Format für Accountingdaten [AL00]. Basierend auf MIME und unter Verwendung von Attribut-Wert-Paaren bzw. Variablenbindungen erlaubt es eine kompakte und protokollunabhängige Repräsentation von Accountingdaten. ADIF definiert zwar keine eigenen Attribute, gibt aber Beispiele an, in denen Accountingattribute von RADIUS genutzt werden.

Die IETF beschäftigt sich vorwiegend mit der Nutzungserfassung insbesondere der Messung des Datenverkehrs. Grundlage bildet hierbei die von OSI

entwickelte Accountingarchitektur (siehe Abschnitt 2.2.2). Zu den anderen Aspekten des Accountings wurden ausschließlich Anforderungen im Zusammenhang mit Einwahlzugängen definiert.

2.2.2 International Organization for Standardization (ISO)

In Abschnitt 2.1 wurden bereits die FCAPS-Funktionsbereiche des OSI-Managements erwähnt.

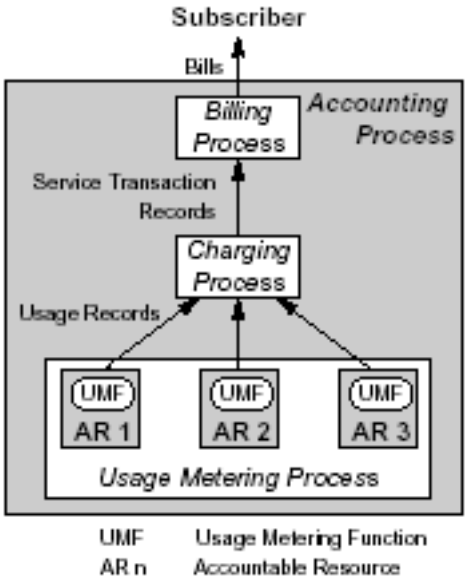

Abbildung 2.10: Architektur des OSI Accounting Managements [Rad03]

Wie in Abbildung 2.10 dargestellt, werden innerhalb des Accounting Managements die drei Teilprozesse Usage Metering Process (Nutzungserfassung), Charging Process (Gebührenberechnung) und Billing Process (Rechnungsstellung) unterschieden [IT95]. Allerdings wurde nur die Nutzungserfassung näher untersucht und die beiden anderen Prozesse blieben bisher unbeachtet.

2.2.3 International Telecommunication Union (ITU)

Das vom Telecommunication Standardization Sector der International Telecommunication Union (ITU-T)[6] entwickelte Telecommunications Management Network (TMN) basiert auf den Spezifikationen des OSI-Managements (siehe Abschnitt 2.2.2). Als Erweiterung der OSI System Management Functions wurden folgende vier Function Set Groups für das Accounting definiert [IT97]:

- Usage Measurement: Nutzungserfassung.
- Tariffing/Pricing: Tarifierung.
- Collections and Finance: Rechnungsstellung sowie auch betriebswirtschaftlich administrative Aspekte (z.B. Personalverwaltung, Rentenverwaltung und Steuererklärung).
- Enterprise Control: Betriebswirtschaftliche, fiskale Aspekte (z.B. Kostenreduzierungsmethoden und Versicherungen).

Die ITU-T beschreibt somit eine umfassende Sicht auf das Accounting Management und beschränkt sich nicht auf den Prozess der Nutzungserfassung. Die Ausführungen bleiben allerdings sehr oberflächlich und spezifizieren keine Prozesse und Funktionen.

2.2.4 IPDR Organization

Die IPDR Organization[7] ist ein Industriegremium, das sich mit der Spezifikation eines offenen und erweiterbaren Datenformats, dem sogenannten Internet Protocol Detail Record (IPDR), sowie einem sicheren, robusten und flexiblen Übertragungsprotokoll zum Austausch von Dienstnutzungsdaten beschäftigt [IPD01], das einen Teil des Prozesses Network Data Management (NDM) aus der Telecom Operations Map (TOM) des TM Forums (siehe Abschnitt 2.2.6) ausmacht.

Das Datenformat wird durch Verwendung von XML-Schema festgelegt. Die baumartige Struktur des in Abbildung 2.11 dargestellten dienstunabhängigen Anteils (Master IPDR Schema) besteht aus folgenden Elementen:

[6] http://www.itu.int/ITU-T/
[7] http://www.ipdr.org/

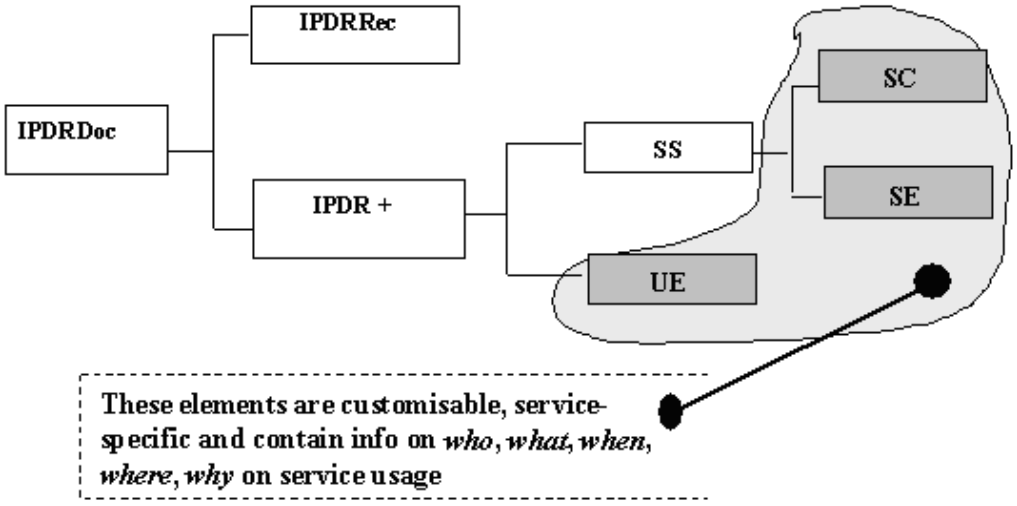

Abbildung 2.11: Master IPDR Schema [FOK02]

- IPDRDoc enthält als oberstes Element eine Menge von IPDR Elementen.
- IPDRRec beschreibt das für die Erzeugung des IPDRDoc verantwortliche Dienst- oder Netzelement.
- IPDR enthält Service Session und Usage Entry zur aktuellen Dienstnutzung.
- Service Session (SS) gruppiert Service Consumer und Service Element Informationen.
- Service Consumer (SC) beschreibt den Dienstnutzer (z.B. Subscriber-ID, IP-Adresse, etc.).
- Service Element (SE) beschreibt das genutzte Dienstelement (z.B. VoIP Serveradresse, Adresse des Anwendungsservers, etc.).
- Usage Entry (UE) beschreibt die Nutzungsdaten (z.B. Startzeit, Endzeit, Dauer, Ergebnisstatus, Name des Video, etc.).

Diese Spezifikationen scheinen erfolgsversprechender als die Bemühungen der IETF zu sein, da sich der IPDR Organization alle relevanten Hersteller angeschlossen haben. Die Steuerung bzw. Konfiguration der an der Abrechnung beteiligten Komponenten wird allerdings nicht berücksichtigt.

2.2.5 Object Management Group (OMG)

Die Object Management Group (OMG)[8] ist ein im April 1989 gegründetes Industriekonsortium, das sich innerhalb zahlreicher Working Groups mit der Entwicklung von Standards im Bereich verteilter Softwareobjekte beschäftigt. Die Telecommunications Domain Task Force definierte die für die Abrechnung von Dienstnutzungen relevante Federated Charging and Rating Facility (FCR) [FOK02]. Ziel ist es dabei, geeignete Schnittstellen für die Kooperation der an der Abrechnung beteiligten Komponenten (z.B. Billing- und Tarifsysteme) einer Multiprovider-Umgebung bereitzustellen, wobei explizit das von der IPDR Organization standardisierte Format zum Datenaustausch unterstützt wird (siehe Abschnitt 2.2.4).

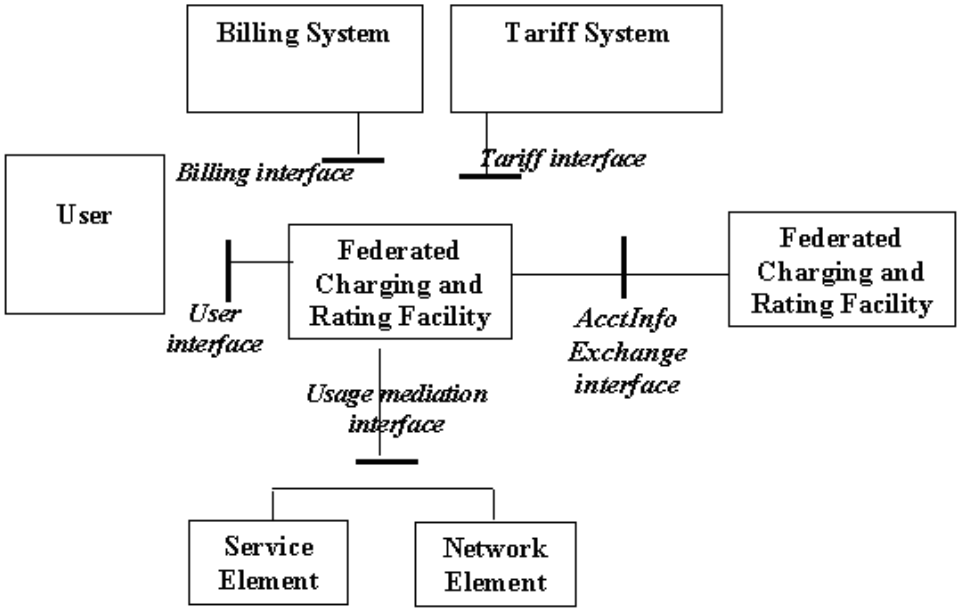

Abbildung 2.12: Die FCR-Architektur aus [FOK02]

Wie in der Abbildung 2.12 dargestellten Architektur zu sehen, stehen für die Sammlung von Nutzungsdaten zwei Arten von Quellen zur Verfügung. Dies sind zum einen die Systeme der Anbieter von Anwendungsdiensten (Service Element) und zum anderen die Systeme der Netzanbieter (Network Element), die beide ihre Daten über die gleiche Schnittstelle aktiv (Push) bereitstellen. Die Distributed Accounting Facility (DAF) berechnet aus den Nutzungsdaten eines Dienstes die Gebühren unter Verwendung des momentan gültigen

[8] http://www.omg.org/

Tarifs. Der gültige Tarif kann über eine definierte Schnittstelle bei einem zugehörigen Tarifsystem abgefragt werden, wobei die Tarifstruktur letztendlich von der konkreten Implementierung der FCR abhängt und somit nur beispielhaft angegeben ist. Registrierte Benutzer des Systems haben über eine spezielle Schnittstelle Zugriff auf vorkalkulierte Gebühreninformationen. Für eine Rechnungsstellung werden die Gebühreninformationen an ein Billingsystem weitergeleitet. Für ein gemeinsames Dienstangebot oder gegenseitige Ressourcennutzung können mehrere Anbieter untereinander Vertragsbeziehungen basierend auf einer Dienstgütevereinbarung eingehen. Diese werden dann zum Austausch von Accountinginformationen zwischen den DAFs der einzelnen Anbieter verwendet.

Ob diese für den Abrechnungsprozess spezifizierten Schnittstellen eine weite Verbreitung finden, ist noch unklar. Für die Steuerung der an der Abrechnung beteiligten Komponenten wurden keine Schnittstellen definiert.

2.2.6 TeleManagement Forum (TM Forum)

Das TeleManagement Forum (TM Forum)[9] ist ein Industriekonsortium, das sich mit dem Management und Betrieb von Informations- und Kommunikationsdiensten beschäftigt, wobei eine wesentliche Initiative in der Erstellung von Spezifikationen zu New Generation Operations Systems and Software (NGOSS) besteht, das die Interoperabilität zwischen verschiedenen Operational Support Systems (OSS) bzw. Business Support Systems (BSS) ermöglicht. In diesem Zusammenhang ist die Telecom Operations Map (TOM) entstanden, die alle für einen Dienstanbieter erforderlichen Prozesse strukturiert beschreibt [TMF00].

Das in Abbildung 2.13 dargestellte TOM Business Process Framework teilt die Prozesse in vier Ebenen und drei Geschäftsprozessbereiche ein. Für das Accounting sind besonders die Prozesse aus dem Bereich Billing relevant. Zu jedem identifizierten Prozess existiert ein Prozessgraph in Form eines Informationsflussdiagramms, das jeweils die Ein- und Ausgabedaten visualisiert und die Bearbeitungsschritte auflistet. Allerdings erfolgt keine genaue Spezifikation der ausgetauschten Daten sowie keine Beschreibung der internen Prozessabläufe. Mechanismen zur Steuerung des Accountings bleiben gänzlich unbeachtet.

[9] http://www.tmforum.org/

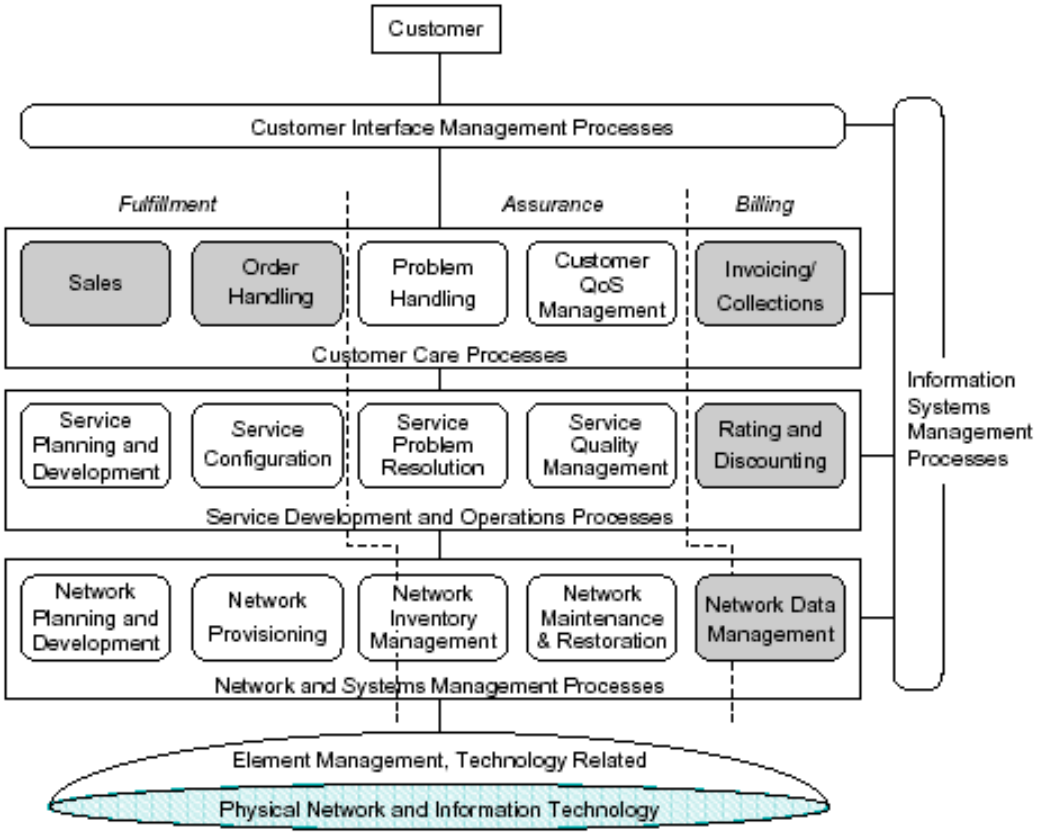

Abbildung 2.13: TOM Business Process Framework [TMF00]

2.2.7 TINA Consortium (TINA-C)

Das Telecommunications Information Networking Architecture Consortium (TINA-C)[10] hat im Rahmen einer Accounting Management Architektur[TC96] das Konzept des Accounting Management Kontext entwickelt, das sowohl qualitative als auch quantitative Details für das Accounting Management spezifiziert. Es ist Teil der Setup- und Wrapup-Phase einer Diensttransaktion, die in der Dienstarchitektur [TC97] beschrieben ist. In der Setup-Phase präsentiert der Benutzer das gewünschte Accountingschema, das mit dem Schema des Anbieters analysiert und beschlossen ist. Der Kontext wird dann entsprechend der Dienstspezifikation und der Accountingumgebung des Anwendungsbereiches (z.B. Tarif, Billing) interpretiert.

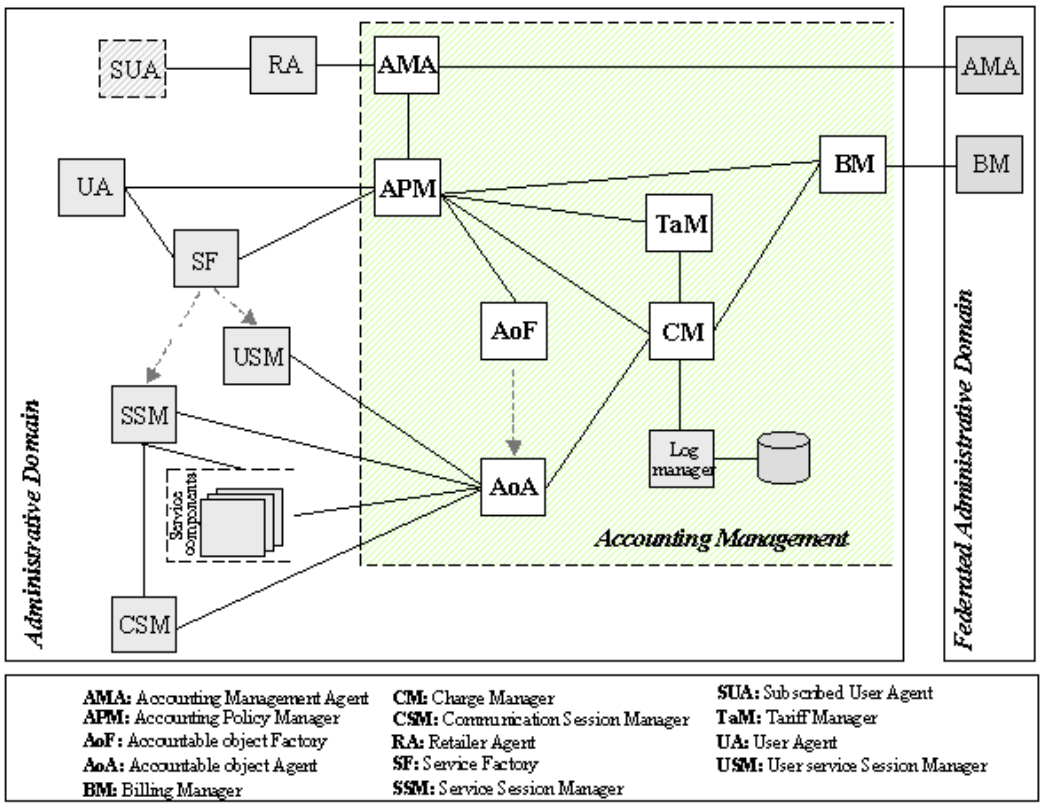

Abbildung 2.14: Accounting Management Modell des Projektes MONTAGE [ZPK+99]

Innerhalb dieses Ansatzes fehlt ein Konzept zur Integration bereits existierender Abrechnungskomponenten in die neue Architektur. Insbesondere wird

[10] http://www.tinac.com/

nicht beschrieben, wie die durch existierende TINA-konforme Plattformen bereitgestellten Daten in entsprechende Abrechnungsdaten gemäß der neu definierten Datenstrukturen überführt werden können. Als Konsequenz ist die Spezifikation zur Accounting Management Architektur mittlerweile nicht mehr relevant. Dennoch gab es Projekte, die diesen Ansatz weiter verfolgt haben, wovon zwei der bekanntesten Arbeiten nachfolgend kurz beschrieben werden.

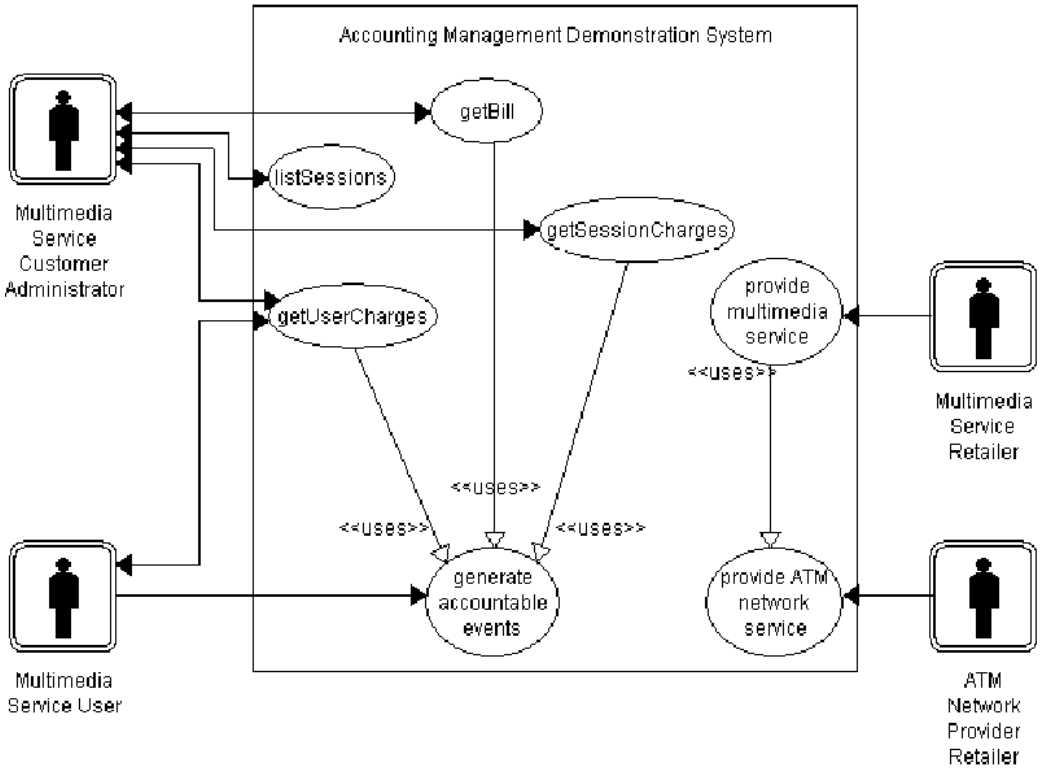

Abbildung 2.15: UseCase-Diagramm zum Accounting in FlowThru [HRDL99]

Im Rahmen des ACTS[11]-Projektes MONTAGE (MObile iNTelligent AGEnts in Accounting, Charging and Personal Mobility Support)[12] wurde ein agentenbasiertes Accounting und Charging für mobilitätsunterstützende Dienste realisiert [ZPK+99]. Das in Abbildung 2.14 dargestellte Accounting Management Modell basiert auf der Annahme, dass nur die Ressourcen für das Accounting Management relevant sind, deren Nutzungen messbar und einem Nutzer zuordenbar sind. Unabhängig davon werden spezielle Chargingfunk-

[11] http://www.cordis.lu/infowin/
[12] http://montage.ccrle.nec.de/

tionen, Tarife und Preispolitiken sowie der Einsatz agentenbasierter Fähigkeiten für Dienstaushandlungen diskutiert.

Ein weiteres ACTS-Projekt FlowThru entwickelte ein Managementsystem, das den Fluss von Managementinformationen zwischen Organisationen und technologischen Bereichen unterstützt. Die Entwicklung eines Accountingsystems [HRDL99] ist ein Teil dieses Projektes, das in die drei in TOM definierten Geschäftsprozessbereiche aufgeteilt ist (siehe Abschnitt 2.2.6). Das Accountingsystem besteht aus einer TINA-basierten Implementierung des Zugangs zu und Kontrolle von Multimediadiensten auf ATM-Netzen. Accountingkomponenten wurden sowohl auf Dienst- als auch auf Netzwerkebene realisiert, wobei die entsprechenden gebührenrelevanten Ereignisse zusammengefasst und dem Dienstkunden in einer einzigen Rechnung präsentiert werden. Abbildung 2.15 gibt einen Überblick über die verschiedenen Anwendungsfälle des Accountings.

2.2.8 M3I Consortium

Das Market Managed Multiservice Internet Consortium (M3I Consortium)[13] beschäftigt sich mit der Realisierung eines auf Marktmechanismen basierenden Systems zum Management von Internetressourcen, speziell durch die Unterstützung von verschiedensten Abrechnungsarten für unterschiedliche Dienste. Ein Teil des M3I-Projektes befasste sich mit der Realisierung eines generischen und modularen Charging und Accounting Systems (CAS) [MC00, MC01], das einen erweiterten und flexiblen Ansatz für z.B. die Implementierung von statischen und dynamischen Preisschemata bietet [SRGF01]. Das Framework unterstützt verschiedene Ebenen von Internetdiensten, wobei hier die höherwertigen Dienste von besonderer Bedeutung sind. Abbildung 2.16 zeigt die komplette Architektur des Internet Charging System (ICS) und den internen Aufbau des darin enthaltenen Internet Charge Calculation und Accounting Systems (ICCAS).

Dieser Ansatz definiert neben dem klassischen Abrechnungsvorgang auch Komponenten für die Einhaltung der Vertragsbedingungen (Customer Support, User Support), für eine Kostenkontrolle bzw. Rechteverwaltung (Host/ Gateway Agent), für die Konkretisierung von dienstspezifischen QoS-Anforderungen (Service Directory) und für das Management aller beteiligten Komponenten (Enterprise Policy Control). Es existiert allerdings zu jeder identifizierten Komponente nur eine kurze Beschreibung und eine stichpunktartige

[13] http://www.m3i.org/

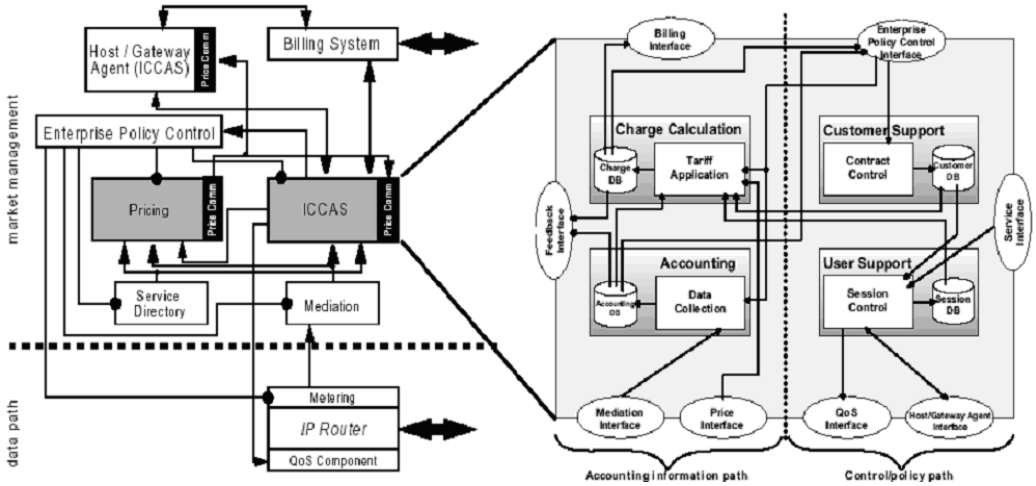

Abbildung 2.16: a) Architektur des Chargingsystems b) interne ICCAS-Architektur [SRGF01]

Auflistung der Funktionen sowie Implementierungsvarianten. Innerhalb der Schnittstellenbeschreibungen sind zwar die zu übertragenden Daten aufgelistet, aber deren Inhalt bzw. Struktur nicht weiter spezifiziert.

Kapitel 3
Agententechnologie

Nachdem in Kapitel 2 verschiedene Konzepte und Lösungen für das Management verteilter Systeme vorgestellt wurden, erfolgt in diesem Kapitel eine Einführung in das Thema der Agententechnologie. Agentenbasierte Systeme sind verteilte Systeme mit besonderen Eigenschaften, auf die in Abschnitt 3.1 genauer eingegangen wird, bevor in Abschnitt 3.2 existierende Standards aus diesem Bereich vorgestellt werden. Anschließend gibt Abschnitt 3.3 einen Überblick über verschiedene Agentenmodelle, die einige der genannten Eigenschaften berücksichtigen, und Abschnitt 3.4 stellt existierende Entwicklungsumgebungen vor, die zum Teil auf diesen Modellen basieren. Abschließend werden in diesem Kapitel die Grenzen der Agententechnologie aufgezeigt.

3.1 Definitionen

Bisher gibt es keine allgemeingültige Definition von Agenten. Für gewöhnlich werden Agenten anhand ihrer Eigenschaften definiert, die sich allerdings von Definition zu Definition unterscheiden. Häufig werden jedoch Systeme (meist Softwareeinheiten) dann als Agenten bezeichnet, wenn sie [WJ95, Fer99]:

- *autonom* agieren,
- *soziales Verhalten* aufweisen,
- sich *reaktiv* und
- *proaktiv* verhalten.

Neben diesen häufig genannten Eigenschaften werden teilweise auch noch weitere Merkmale erwähnt [Alb98]:

- Adaptivität
- Mobilität

In den folgenden Unterabschnitten werden nun die genannten Begriffe etwas genauer erläutert.

3.1.1 Autonomie

Agenten handeln autonom, wenn sie ohne direkten Eingriff von außen (durch Menschen oder andere Agenten) Aktionen selbständig ausführen können und dabei Kontrolle über ihre Handlungen und internen Zustände besitzen. Im Gegensatz zum objektorientierten Ansatz können Agenten auch Anfragen ablehnen bzw. von anderen Faktoren (z.B. Berechtigungen, eigene Ziele, etc.) abhängig machen.

3.1.2 Soziale Fähigkeiten

Basis für jegliches soziales Verhalten ist die Kommunikation mit anderen Entitäten. Die Kommunikation zwischen Agenten erfolgt über den Austausch von Sprechakten [Sea69] unter Verwendung einer gemeinsamen Kommunikationssprache (Agent Communication Language). Beispiele hierfür sind FIPA-ACL (siehe Abschnitt 3.2.1) und KQML (Knowledge Query and Manipulation Language).

Basierend auf der Kommunikation können weitere soziale Fähigkeiten wie Kooperation und Koordination vorhanden sein. Bei der Kooperation handelt es sich um das gemeinsame Erreichen höherwertiger Ziele. Diese Ziele werden in Teilziele zerlegt, die von den einzelnen Beteiligten erbracht werden können. Die Einhaltung von notwendigen Rahmenbedingungen erfordert unter Umständen die Koordination von einander abhängigen Zielen und Aufgaben.

Um die Interoperabilität insbesondere zwischen verschiedenen Systemen zu erhöhen und Agenten unterschiedlicher Anwendungen und Organisationen

miteinander kooperieren zu lassen, ist für die Kommunikation neben der Verwendung von Standards auch die Definition der Semantik in Form von Ontologien notwendig. Ontologien beschreiben Dinge eines bestimmten Anwendungsbereiches und ihre Beziehungen zueinander. Sie können beispielsweise für die Beschreibung von Diensten und den Austausch von Informationen verwendet werden. Für die Repräsentation von Ontologien stehen eine Vielzahl von Ontologiesprachen (z.B. OWL [MvH03], JADL [KHA06]) zur Verfügung, die jeweils unterschiedliche Konzepte bereitstellen.

3.1.3 Reaktivität

Agenten können ihre Umgebung wahrnehmen und auf Änderungen geeignet und zeitgerecht reagieren. Reaktivität wird meist dadurch erreicht, dass die Umgebung des Agenten permanent überwacht wird und eine Menge von Regeln vorhanden sind, die bestimmte Zustände der Umgebung auf Aktionen abbilden.

3.1.4 Proaktivität

Agenten agieren nicht nur auf Anfragen von außen, sondern können auch zielgerichtetes Verhalten aufweisen und eigenständig die Initiative ergreifen. Es können verschiedene Arten von Zielen unterschieden werden[WPHT02]. Deklarative Ziele beschreiben einen Zustand der erreicht werden möchte. Prozedurale Ziele hingegen geben Handlungen an, die ausgeführt werden sollen.

3.1.5 Adaptivität

Ein Agent verhält sich adaptiv, wenn er sich an seine Umgebung anpassen kann. Dazu sollte er Änderungen in der Umgebung erkennen und aus den veränderten Auswirkungen seiner Aktionen lernen können. Voraussetzung ist, dass der Agent die Fähigkeit besitzt seine Handlungen zu beobachten.

3.1.6 Mobilität

Ein Agent ist mobil, wenn er sich eigenständig durch ein Computernetzwerk bewegen kann. Die Übertragung der Agenten erfolgt dabei durch die jeweilige Laufzeitumgebung für Agenten auf den beteiligten Rechnern.

Dabei kann zwischen schwacher und starker Migration unterschieden werden. Bei der schwachen Migration werden nur die Fähigkeiten (i.a. der Code) des Agenten übertragen. Nach erfolgreicher Migration startet der Agent seine Handlungen von Beginn an oder führt sie an vorgegebener Stelle fort. Bei der starken Migration wird zusätzlich der gesamte Zustand des Agenten übertragen, so dass der Agent seine Handlungen genau an dem Punkt fortsetzen kann, an dem er vor der Migration stehen geblieben war.

3.2 Standards

In den folgenden Abschnitten 3.2.1 und 3.2.2 werden die beiden bekanntesten Standards *FIPA* und *MASIF* aus dem Bereich der agentenbasierten Systeme vorgestellt. Da bei den meisten Agentenmodellen die Kooperation zwischen Agenten über gegenseitige Dienstnutzungen abläuft, lohnt es sich auch einen Blick auf die Standards der *Web Services* und des *Semantic Web* (siehe Abschnitte 3.2.3 und 3.2.4) zu werfen.

3.2.1 Foundation for Intelligent Physical Agents (FIPA)

Die Foundation for Intelligent Physical Agents (FIPA)[1] ist eine 1996 gegründete gemeinnützige Organisation, die Standards für die Interoperabilität von heterogenen Agentensystemen definiert. Seit 2005 ist FIPA das elfte Standardisierungskomitee der IEEE Computer Society und beschäftigt sich zusätzlich mit der Interoperabilität der Agententechnologie mit anderen Technologien. Zu den vollwertigen Mitgliedern gehörten im Jahre 2006 weltweit 33 Firmen und Organisationen (z.B. Boeing, CACI, Nippon Telegraph & Telephone, Rockwell Automation, RWTH Aachen, Siemens, Toshiba, Whitestein Technologies, etc.).

FIPA definiert eine Abstract Architecture, die auf einer abstrakten Ebene diejenigen Elemente identifiziert, die grundlegende Bausteine eines Agenten-

[1] http://www.fipa.org/

systems darstellen [FIP02a]. Dieses Agentensystem stellt ein ideales System dar, welches durch konkretere Spezifikationen verfeinert wird. Diese konkreteren Spezifikationen, beispielsweise die Kommunikations- oder Managementspezifikationen, können dann als Referenz verwendet werden, um letztendlich Agentensysteme zu implementieren. Die Art und Weise der Implementierung, d.h. auf welchem System (Betriebssystem, Programmiersprache) dieses Agentensystem beruht, ist weitgehend freigestellt. FIPA identifiziert lediglich wichtige Komponenten, die autonome Ausführung und semantische Interoperabilität ermöglichen.

Dabei greift FIPA auf die Analyse von verschiedenen Varianten zurück wie Software strukturiert werden kann, um die Besonderheiten eines Multiagentensystems zu implementieren. Ein solches Multiagentensystem kann als autonom ausführbares Areal gesehen werden, in welchem Einheiten (Agenten) intentionale und zielgerichtete Aktionen durchführen. Dabei sind diese Einheiten in der Lage, untereinander auf einer hohen semantischen Ebene miteinander zu kommunizieren. Dadurch wird eine „Semantische Interoperabilität“ geschaffen, die es ermöglicht, offene Systeme zu schaffen, die gemeinsam Probleme lösen können.

Konkrete Implementierungen für die FIPA Spezifikationen existieren in Form von Agentenentwicklungsumgebungen. Dabei existieren nur wenige Vertreter, die diese Spezifikationen so implementiert haben, dass sie untereinander interoperabel sind. Hierzu gehören JADE (Java Agent Development Framework) und FIPA-OS, wobei JADE am weitesten verbreitet ist.

FIPA definiert teilweise ausführlich die Agenteninfrastruktur inklusive der Ausführung und Registrierung von Agenten sowie deren Kommunikation untereinander. Alle anderen Managementbereiche wie das Fehler-, Konfigurations-, Abrechnungs- und Leistungsmanagement werden allerdings noch nicht berücksichtigt.

FIPA-ACL

Um Interoperabilität auf der Ebene der Kommunikation zwischen Agenten zu gewährleisten, sind sowohl die Struktur [FIP02b] als auch mögliche Repräsentationen (z.B. Bit-Efficient, String, XML) der ausgetauschten Nachrichten spezifiziert.

Eine Nachricht enthält eine Menge von Parametern, die sowohl durch FIPA (siehe Tabelle 3.1) als auch selbstdefiniert sein können. Einziger notwendiger

Parameter	Beschreibung
performative	Typ des Sprechaktes
sender	Identität (Agentenname) des Senders
receiver	Identität des beabsichtigten Empfängers
reply-to	Name des Agenten, der die Antworten erhalten soll
content	Eigentlicher Inhalt (Aktion) der Nachricht
language	Sprache, in der der Inhalt formuliert ist
encoding	Bestimmte Kodierung des Inhaltes
ontology	Ontologie(n), die dem Inhalt eine Bedeutung geben
protocol	Interaktionsprotokoll, das der Sender verwendet
conversation-id	Ordnet Sprechakte einer Konversation zu
reply-with	Ausdruck, auf den die Antworten verweisen sollen
in-reply-to	Verweis der Antwort auf eine Anfrage
reply-by	Spätester Zeitpunkt zu dem eine Antwort erwartet wird

Tabelle 3.1: FIPA-ACL Nachrichtenparameter

Parameter ist „performative“, wobei meist auch die Angabe von „sender“, „receiver“ und „content“ erwartet wird.

Agent Management Reference Model

Um einen Rahmen für diese Interaktionen und die Möglichkeit für die Existenz von Agenten zu schaffen, wurde in FIPA auch auf den Management Aspekt Rücksicht genommen. Dort wird das Starten, Ausführen und Beenden von Agenten beschrieben. Ausserdem wird beschrieben wie die Agenten ihre gegenseitigen Fähigkeiten erfahren können und wie Directory Services eingegliedert werden können.

FIPA spezifiziert in der Agent Management Specification diejenigen Komponenten die für die Management-Aspekte eines Multiagentensystemes wünschenswert oder notwendig sind [FIP02c]. Dabei werden die in der Abstract Architechture identifizierten Komponenten verfeinert. Für die Kommunikation mit diesen Komponenten ist die Sprache „fipa-sl0“ [FIP02e] als Sprache für den Inhalt der Nachrichten vorgeschrieben.

Die Agent Management Specification beschreibt ein Agent Management Reference Model welches bestimmte Entitäten vorsieht, die auf einer Agentenplattform vorhanden sein müssen oder können (siehe Abbildung 3.1). Die folgenden Entitäten wurden in der Spezifikation identifiziert:

- *Agent*: Der Agent ist eine grundlegende autonome Einheit in einem Agentensystem. Diese führt funktionelle, zielgerichtete Aktionen durch, die es dem Agentensystem ermöglicht dessen Intentionen gemäß der Ziele zu erreichen. Agenten kommunizieren mit anderen Agenten durch eine Agent Communication Language, die ebenfalls von FIPA definiert wird. In der Beschreibung eines Agenten können dessen Eigentümer, die angebotenen Dienste, die unterstützten Protokolle sowie die bekannten Ontologien und Sprachen angegeben sein.

- *Agent Platform (AP)*: Dieser Agent befindet sich mit anderen Agenten auf einer Agentenplattform, welche die Umgebung darstellt in der Agenten ausgeführt werden. Hier werden Agenten gestartet, gestoppt und ihre Lebenszykluszustände überwacht.

- *Message Transport Service (MTS)*: Die Kommunikation der Agenten findet über den Message Transport Service [FIP02d] statt, der dadurch eine zentrale Rolle innehat. Die Fähigkeiten (unterstützte Transportprotokolle, z.B. IIOP oder HTTP) des MTS können in der Plattformbeschreibung angegeben werden.

- *Agent Management System (AMS)*: Das AMS überwacht die Agent Platform und reguliert die Laufzeit der Agenten und den Zugriff auf die Agentenplattform. Auf jeder Plattform muss ein AMS vorhanden sein. Er bietet zusätzlich noch einen White Pages Dienst an, bei dem sich Agenten auf der Plattform registrieren müssen. Jeder Agent erhält durch den AMS einen Agent Identifier (AID), der den Agenten mittels eines eindeutigen Namens (z.B. *name*@*hap_name*[2]) und evtl. bekannter Kommunikationsadressen identifiziert. Das AMS hat den Namen ams@*hap_name*.

- *Directory Facilitator (DF)*: Der DF bietet einen Gelbe Seiten Dienst an, bei dem Dienstanbieter ihre Dienstbeschreibungen registrieren können. Diese Registrierungen können von anderen Agenten durchsucht werden, damit diese einen geeigneten Dienstanbieter finden können. Auf einer Agentenplattform können mehrere DFs existieren, die durch gegenseitige Registrierung miteinander bekannt sein können. Der standardmäßige DF hat den Namen df@*hap_name*.

- *Software*: Dies stellt eine Anwendung dar, die nicht durch Agenten betrieben wird. Auf diese wird von den Agenten der Plattform zugegrif-

[2]Der Name der Heimatagentenplattform (HAP) ist in der Beschreibung der HAP eindeutig angegeben.

fen, um beispielsweise Informationen zu erhalten oder zur Verfügung zu stellen.

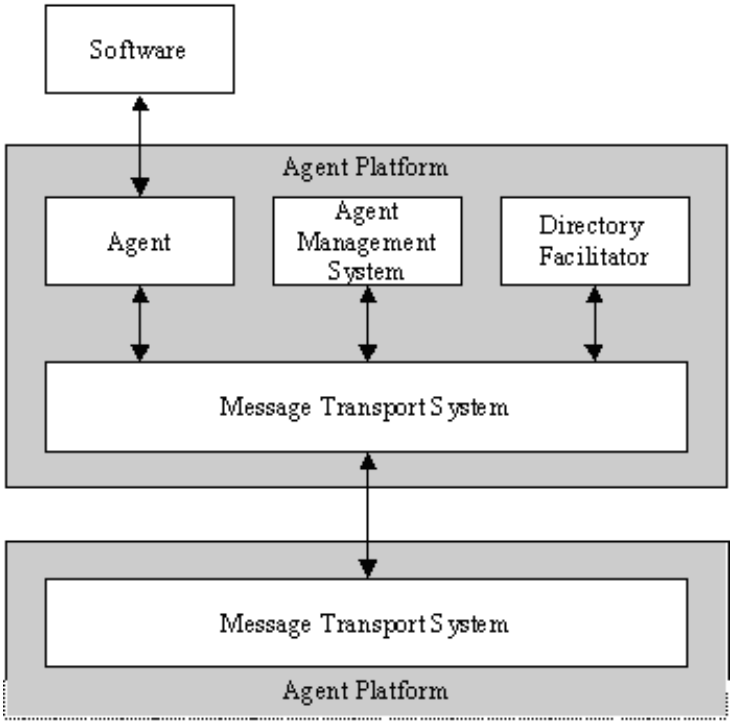

Abbildung 3.1: FIPA Agent Management Reference Model [FIP02c]

Zwei neuere noch nicht zum Standard erklärte Spezifikationen schlagen einen Agent Discovery Service (ADS) für die plattformübergreifende Bekanntmachung bzw. Registrierung von Agenten und Diensten auf entfernten Endgeräten in sogenannten Ad-Hoc-Netzwerken vor [FIP03a, FIP03b]. Letztere Spezifikation beschäftigt sich dabei speziell mit der JXTA-Technologie. Der ADS erlaubt zusätzlich das Abonnieren [FIP02f] von bestimmten Informationen, das einer Realisierung des Push-Prinzips entspricht.

Ontologie zum Agentenmanagement

Grundlage für alle zuvor genannten Dienste des Agentenmanagements ist die Ontologie „fipa-agent-management“ [FIP02c], die die notwendigen Datenstrukturen und ihre Beziehungen untereinander definiert. Diese Ontologie gibt zum Beispiel an, wie APs, Agenten und Dienste bei Verwendung eines DF bzw. AMS beschrieben sein müssen (siehe Abbildung 3.2).

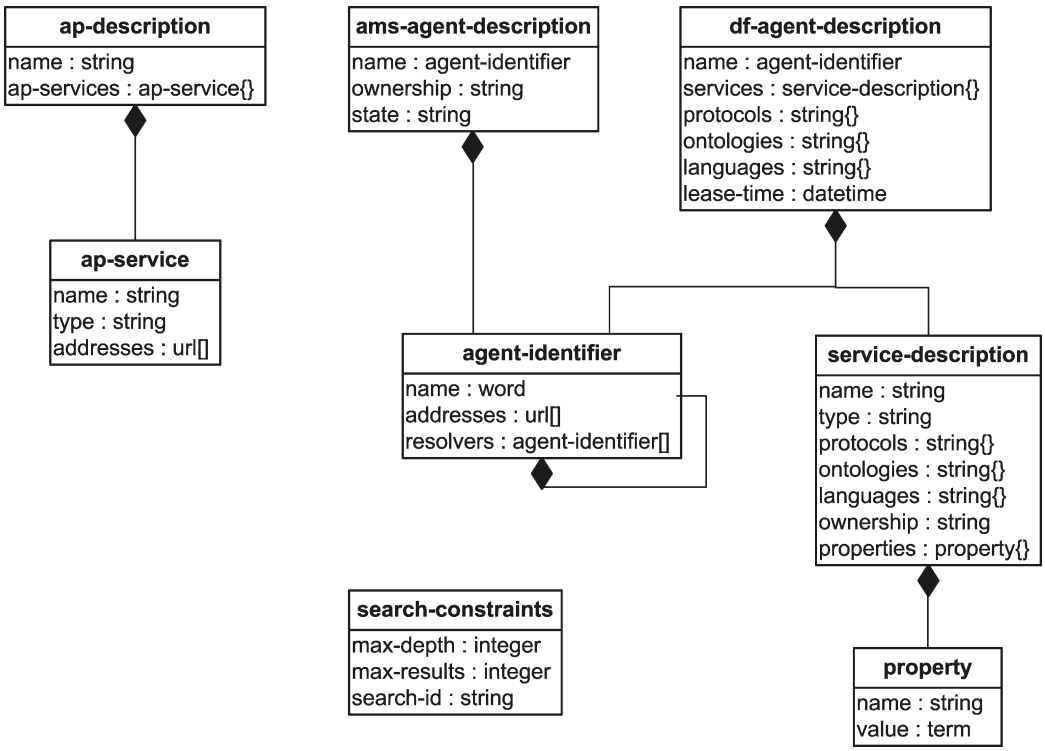

Abbildung 3.2: FIPA-Ontologie zum Agentenmanagement

Eine AP wird durch einen Namen und eine Menge angebotener Dienste beschrieben, die wiederum einen Namen, einen Typ (z.B. „fipa.mtp.*“) und eine Adressliste enthalten. Ein AMS kann zu jedem Agenten die Identifizierung (eindeutiger Name, Kommunikationsadressen und mögliche Adressverwalter), den Eigentümer und den aktuellen Zustand verwalten. Ein DF kann neben der Identifizierung auch die angebotenen Dienste, unterstützten Protokolle sowie bekannten Ontologien und Sprachen eines Agenten verwalten. Zu jedem Dienst kann ein Name, der Typ, eine Menge von unterstützten Protokollen, Ontologien und Sprachen sowie der Eigentümer und eine Menge von Properties angegeben werden. Für die Suche nach Agenten kann eine maxi-

male Suchtiefe, eine maximale Anzahl von Ergebnissen und eine ID definiert werden.

3.2.2 Mobile Agent System Interoperability Facility (MASIF)

Bereits vor den Aktivitäten der FIPA entwickelte die Object Management Group (OMG), der u.a. die GMD Fokus, International Business Machines Corporation, Crystaliz, General Magic und Open Group angehörten, einen Standard für die Interoperabilität von Agentensystemen. In der MASIF-Spezifikation wird ein Common Conceptual Model, einige CORBA-Dienste, die Mobile Agent Facility IDL (Interface Definition Language) und ein Szenario definiert [OMG00].

Das in Abbildung 3.3 dargestellte Common Conceptual Model soll das Management, die Migration und die Verfolgung von Agenten zwischen unterschiedlichen Agentensystemen ermöglichen. Zu den Basiskonzepten gehören stationäre und mobile Agenten, die einen bestimmten Ausführungszustand besitzen, und eine Infrastruktur für die Kommunikation zwischen den Agenten. Agenten befinden sich auf Plätzen, die Agentensystemen[3] und diese wiederum Regionen zugeordnet sind. Agentensysteme ermöglichen die Erzeugung und eine starke Migration von Agenten, die Generierung global eindeutiger Namen für Systeme, Plätze und Agenten, das Auffinden von mobilen Agenten und die Schaffung einer sicheren Umgebung für Agentenoperationen. Durch die Bereitstellung der Infrastruktur kann ein Agent jederzeit andere Agenten auf einem entfernten System erzeugen, die Agenten zu einem anderen System des gleichen Typs migrieren lassen und Methoden eines anderen Agenten aufrufen.

Die Infrastruktur greift dabei auf verschiedene CORBA-Dienste zurück. Folgende Dienste sind innerhalb der Spezifikation definiert worden:

- *Naming Service*: Eindeutige Abbildung von Namen auf CORBA-Objekte
- *Lifecycle Service*: Erzeugen, Löschen, Kopieren und Übertragen von CORBA-Objekten
- *Externalization Service*: Serialisierung und Deserialisierung der Zustände von CORBA-Objekten

[3] Von den Agentensystemen (bei FIPA auch Agentenplattformen genannt) existiert meist genau eine Instanz pro Rechner.

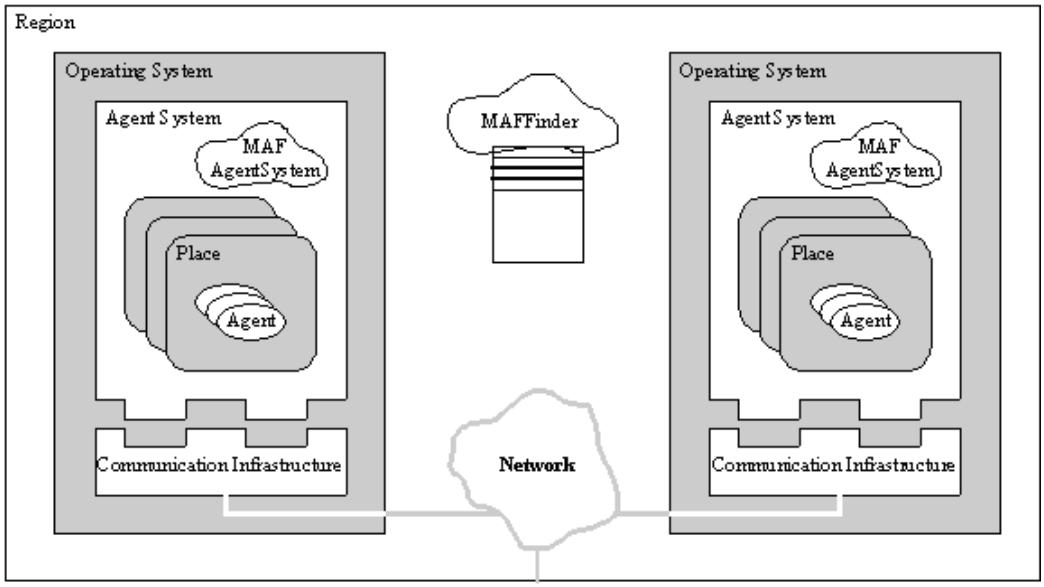

Abbildung 3.3: MASIF Common Conceptual Model

- *Security Service*: Sicherheitsanforderungen für Agenten und Agentensysteme in CORBA (Namensgebung, Integrität, Vertraulichkeit, Aufdeckung von Wiedereinspielungen, Authentifizierung, Sicherheitspolitiken)

Der global eindeutige Name von Agenten und Systemen besteht aus dem Namen der Person bzw. Organisation, der innerhalb eines Agentensystemtyps eindeutigen Identität und dem Typ des Agentensystems. Der Ort wird als URI (Universal Resource Identifier) [BL94] oder URL (Uniform Resource Locator) [BLMM94] der Form „mafiiop://[*host*:*port*/]*agentsystem*[/*place*]*“ angegeben. Jedem Agentensystem sind ein Name, ein Typ, Serialisierungsmethoden für die unterstützten Programmiersprachen, eine Systembeschreibung mit Versionsspanne und verschiedene Properties zugeordnet. Das MAFAgentSystem (ähnlich zum AMS in FIPA) bietet Operationen zur Terminierung des Agentensystems, zur Erzeugung, Terminierung, Unterbrechung und zum Aufwecken von Agenten, zum Empfang und zur Instanziierung von Agenten, zum Laden von Klassen, zur Suche nach dem nächsten Agentensystem mit der Fähigkeit zur Ausführung eines bestimmten Agenten und zur Abfrage verschiedener Informationen wie der Zustand eines Agenten, die Systeminformation, Authentifizierung eines Agenten, die MAFFinder-Referenz sowie die Liste aller im Agentensystem registrierten Agenten einer bestimmten Person oder Organisation und aller Plätze innerhalb des Agentensystems. Der

MAFFinder (ähnlich zum DF in FIPA) bietet Operationen zur Registrierung, Deregistrierung und Lokalisierung von Agenten, Plätzen und Agentensystemen innerhalb einer Region an.

3.2.3 Web Services

Ein Web Service [BHM+04] ist eine Software, der ein eindeutiger Uniform Resource Identifier (URI) [BLFM98] zugeordnet ist und deren Schnittstellen in XML [BPSM+04] definiert und beschrieben sind. Web Services können von anderen Softwaresystemen über einen Verzeichnisdienst gefunden und durch Austausch von XML-basierten Nachrichten über internetbasierte Protokolle genutzt werden. Web Services erlangen im praktischen Einsatz eine stark zunehmende Bedeutung.

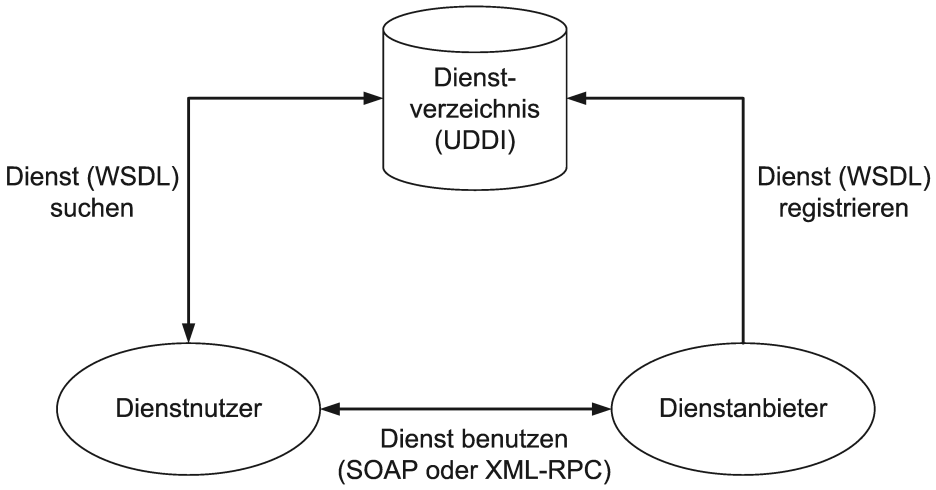

Abbildung 3.4: Architektur der Web Services

Die Grundlage der Web Services bilden drei Standards, die jeweils auf XML basieren und in den nächsten Abschnitten näher beschrieben werden:

- *UDDI*: Dieser Standard ist ein Verzeichnisdienst zur Registrierung und Suche von Web Services.
- *WSDL*: Diese Sprache dient zur Beschreibung der angebotenen Methoden eines Web Service und deren Parameter.
- *SOAP (oder XML-RPC)*: Diese Protokolle regeln die Kommunikation. Hier wird der eigentliche Aufruf gestartet.

UDDI

Universal Description, Discovery and Integration (UDDI) [CHvRR04] bezeichnet einen Verzeichnisdienst für Web Services und andere Dienste. Dieser Verzeichnisdienst ist selbst ein Web Service, der Informationen über Dienste und deren Anbieter verwaltet. Dienstanbieter können mit UDDI ihre Dienste bekannt geben. Dienstnutzer können entsprechend ihren Anforderungen nach passenden Diensten suchen und sich Informationen über die Nutzungsbedingungen geben lassen. UDDI unterscheidet dabei zwischen drei Arten von Informationen.

- *White Pages*: Dienstanbieter mit Kontaktinformationen und weiteren Detailangaben
- *Yellow Pages*: Dienste mit Branchenangabe, Anbieterliste und andere Informationen
- *Green Pages*: Technische Details zu den angebotenen Diensten

WSDL

Web Services Description Language (WSDL) [CMRW06] ist ein XML-Standard zur Beschreibung von Web Services. Die Operationen und Nachrichten werden unabhängig vom Netzwerkprotokoll und Nachrichtenformat definiert. Dabei werden im Wesentlichen die Funktionen definiert, die von außen zugänglich sind, sowie die Parameter und Rückgabewerte dieser Operationen.

SOAP

SOAP [GHM+03] ist ein Protokoll zum Austausch von Informationen zwischen Systemen und zur Durchführung von entfernten Methodenaufrufen unter Verwendung von XML für die Repräsentation der Daten und Internetprotokollen für die Übertragung der Daten. Am häufigsten wird HTTP und TCP verwendet. Ursprünglich war SOAP die Abkürzung für Simple Object Access Protocol, seit Version 1.2 ist SOAP jedoch offiziell keine Abkürzung mehr, da es nicht nur dem Zugriff auf Objekte dient.

Erweiterungen zu Web Services

Zur Technologie der Web Services existieren bereits eine Vielzahl von Erweiterungen, so beispielsweise WS-Security, WS-Policy, WS-Notification, WS-Transaction, WS-BPEL, WS-Federation und WS-Eventing.

Bezüglich der Managementaspekte ist das Web Service Distributed Management (WSDM) von besonderem Interesse. Hierzu gehören die beiden Spezifikationen Management Using Web Services (MUWS) [BV06] und Management Of Web Services (MOWS) [WS06]. MUWS beschreibt eine auf der Web Service Technologie basierende Schnittstelle für das Management beliebiger Ressourcen. Die Verwaltung von Ressourcen erfolgt dabei über entsprechende Zugangspunkte. Jeder dieser Zugangspunkte definiert eine eindeutige Identität, eine Menge zu verwaltender Eigenschaften (z.B. eine Beschreibung, den Zustand, die Verfügbarkeit, verschiedene Metriken oder die Konfiguration) der Ressource und mögliche Beziehungen zu anderen Zugangspunkten. MOWS definiert eine Spezialisierung dieser Schnittstelle für den Fall, dass es sich bei den zu verwaltenden Ressourcen um Web Services handelt. Hier sind unter anderem zusätzliche Metriken (z.B. Anzahl erfolgreicher und gescheiterter Aufrufe oder maximale und letzte Antwortzeit) und die Nutzung des Web Service Lifecycle (WSLC) [HPS04] als Zustandsmodell für Ressourcen und Aufrufverarbeitung definiert.

Somit stellt das WSDM eine erste Basis für das Management von Ressourcen (z.B. Web Services) unter Verwendung der gleichen Technologie dar, jedoch werden keine weiterführenden Funktionen wie beispielsweise Fehler-, Konfigurations- oder Abrechnungsmanagement spezifiziert. Da WSDM auf der Web Service Technologie basiert, sind diese Spezifikationen für das Management agentenbasierter Systeme erst bei Bereitstellung eines Web Service Adapters relevant.

3.2.4 Semantic Web

Das Semantic Web [BLHL01] ist eine Erweiterung des World Wide Web um maschinenlesbare Informationen mit einer wohldefinierten Bedeutung. Es erlaubt somit Daten zwischen verschiedenen Anwendungen und Organisationen auszutauschen bzw. zu verarbeiten und ermöglicht eine bessere Zusammenarbeit zwischen Rechnern und Menschen. Die Repräsentation der Informationen basiert auf dem Resource Description Framework (RDF) [LS99] oder der darauf aufbauenden Web Ontology Language (OWL) [MvH03], die wiederum XML für die Syntax und URI für die Identifizierung verwendet.

Die Aktivitäten rund um das Semantic Web werden vom World Wide Web Consortium (W3C)[4] koordiniert, an denen sich auch zahlreiche Forscher und Industriepartner beteiligen. Die Semantic Web Services Interest Group des W3C bietet bereits ein Forum, in dem die Integration der Semantic Web Technologie in die Web Service Architektur diskutiert wird. Wie beispielsweise eine kombinierte Nutzung von OWL-S und WSDL aussehen könnte wird bereits in [MBL+04] beschrieben.

RDF

Beim Resource Description Framework (RDF) [LS99] handelt es sich um Spezifikationen des W3C, die ein generisches Modell für Metadaten (Beschreibungen der im WWW vorhandenen Informationen) definieren und somit den Austausch und die Verarbeitung von Informationen im WWW verbessern. Für die Repräsentation der RDF-basierten Informationen wird meist die Syntax von XML und die Identifizierung durch URI verwendet.

Das RDF Basismodell besteht aus folgenden drei Typen:

- *Resources*: Zu den RDF-Ressourcen gehören alle Dinge, die durch einen RDF-Ausdruck beschrieben sind (z.B. einzelne Webseiten, Teile von Webseiten oder eine Sammlung von Webseiten). Sie werden durch Verwendung von URI eindeutig benannt.
- *Properties*: RDF-Properties sind spezielle Merkmale, Attribute, Eigenschaften oder Beziehungen, die eine Ressource genauer beschreiben. Properties besitzen eine bestimmte Bedeutung und definieren ihren Wertebereich, die Art der beschreibbaren Ressourcen sowie ihre Beziehungen zu anderen Properties.
- *Statements*: Ein RDF-Statement ist ein Triple bestehend aus einer Ressource (Subjekt), einer Property (Prädikat) und einem Wert für diese Property (Objekt). Der Wert einer Property kann eine Ressource, ein URI, eine einfache Zeichenkette oder ein primitiver Datentyp sein.

[4] http://www.w3.org/

OWL

Die OWL Web Ontology Language [MvH03] wurde vom W3C entwickelt und ist ein Nachfolger der DAML+OIL Web Ontology Language [MFHS02]. Beide Sprachen sind aus den Aktivitäten innerhalb des Forschungsschwerpunktes „Semantic Web“ entstanden und gehören mittlerweile zu den verbreitetsten Ontologiesprachen.

OWL basiert auf RDF und erlaubt das Erstellen, das Publizieren und die Verteilung von Ontologien, die Dinge eines Problembereiches und deren Beziehungen untereinander formal beschreiben. OWL ermöglicht durch zusätzliche Sprachkonstrukte und einer formalen Semantik eine bessere maschinelle Interpretation und Verarbeitung von Informationen als XML und RDF. Zum zusätzlichen Vokabular für die Beschreibung von Klassen und Merkmalen gehören unter anderem Klassenbeziehungen (z.B. Disjunktion), Kardinalität (z.B. „exactly one“), Gleichheit, umfangreichere Typisierung von Merkmalen, Eigenschaften von Merkmalen (z.B. Symmetrie) und Aufzählungstypen.

OWL bietet drei Teilsprachen, die aufeinander aufbauen und somit unterschiedliche Ausdruckskraft besitzen:

- *OWL Lite*: Sie unterstützt im Wesentlichen Klassifizierungshierarchien und einfache Randbedingungen und besitzt somit eine geringere Komplexität als OWL DL.
- *OWL DL*: Diese Sprache (benannt nach „description logics“) enthält als eine äquivalente Untermenge der Prädikatenlogik erster Stufe alle Konstrukte von OWL, die aber nur mit bestimmten Einschränkungen benutzt werden können, um vollständige Berechenbarkeit zu gewährleisten. Dies ermöglicht aus vorhandenem Wissen neues Wissen zu schlussfolgern.
- *OWL Full*: Sie besitzt die maximale Ausdruckskraft mit der syntaktischen Freiheit von RDF. Eine vollständige Berechenbarkeit wird damit allerdings nicht garantiert.

OWL-S

OWL-S [MBH+04] ist eine OWL-basierte Ontologie für die Beschreibung der Merkmale und Fähigkeiten von Diensten des Semantic Web in einer eindeutigen und maschinenlesbaren Form. Diese Ontologie ermöglicht die Automatisierung verschiedener Aufgaben wie die Ermittlung, Ausführung, Zusammensetzung und Zusammenarbeit von Diensten. Frühere Versionen von OWL-S basierten auf der Ontologiesprache DAML+OIL (dem Vorgänger von OWL) und waren unter dem Namen DAML-S [ABH+01] bekannt.

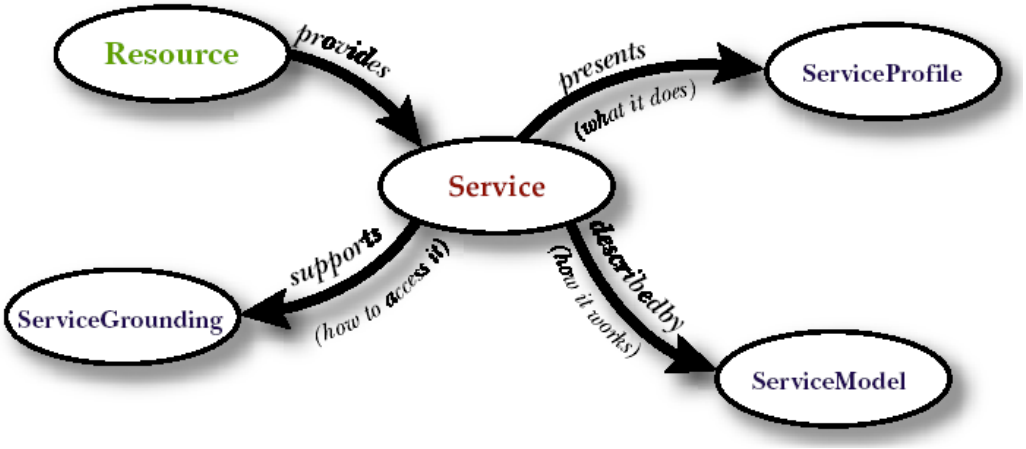

Abbildung 3.5: Grobstruktur der Ontologie OWL-S [ABH+01]

Auf der obersten Ebene von OWL-S werden Eigenschaften definiert, die für alle Arten von Diensten des Semantic Web gelten (siehe Abbildung 3.5), wobei Erweiterungen um anwendungsspezifische Details möglich sind. Jeder Dienst besitzt ein Profil (ServiceProfile), ist durch ein Modell (ServiceModel) beschrieben und unterstützt eine definierte Schnittstelle (ServiceGrounding) für die Dienstnutzung. Es ist keine minimale und maximale Anzahl dieser Elemente für ein Dienst vorgeschrieben. Einzige Bedingung ist, dass zu jedem definierten Dienstmodell eine entsprechende Dienstschnittstelle angegeben werden muss.

Das Dienstprofil beschreibt Informationen über die Funktionalität des Dienstes, die für dienstsuchende Agenten relevant sind. Hierzu gehören eine menschenlesbare Beschreibung des Dienstes, Ein- und Ausgabeparameter, Vor- und Nachbedingung sowie weitere funktionale Attribute (z.B. Qualitätsmerkmale). Das Dienstmodell beschreibt, wie der Dienst abläuft. Hierfür wird üblicherweise ein Prozessmodell (ProcessModel) angegeben, das sowohl den eigentlichen Prozess (Process) als auch die Prozesskontrolle (ProcessControl)

beschreibt. Eine Dienstschnittstelle definiert ausführlich, wie ein Agent auf den Dienst zugreifen kann. Hierzu gehören meist ein Kommunikationsprotokoll (z.B. RPC, HTTP, CORBA, SOAP) und dienstspezifische Informationen für den Dienstzugriff (z.B. Portnummern).

3.3 Agentenmodelle

Die zum Teil recht unterschiedlichen Agentensysteme lassen sich verschiedenen Agentenmodellen zuordnen. Diese reichen von rein reaktiven Agenten, deren Handlungen direkt durch Wahrnehmung der Umgebung ausgelöst werden, bis hin zu deliberativen Agenten, die Schlussfolgerungen ziehen und ihre Handlungen bewusst auswählen und planen können.

In [RN95] werden vier Ausprägungen solcher Agenten definiert, die alle ihre Umgebung wahrnehmen und daraus Handlungen ableiten. Das einfachste Modell eines Agenten ist der einfach reaktive Agent, der basierend auf Bedingungs-Aktions-Regeln direkt auf eine Wahrnehmung reagiert. Der modellbasierte reaktive Agent besitzt ein Gedächtnis und somit eine Art internen Zustand über die bereits gesammelten Informationen, das ihm ein Modell seiner Umgebung und die Auswirkungen seiner Handlungen vermittelt, auf denen dann die Regeln angewandt werden. Ein Vertreter für reaktive Agenten ist beispielsweise die Brooks' Subsumption Architecture [Bro91]. Zielbasierte Agenten hingegen verfolgen Ziele und wählen die Aktionen aus, die sie diesen Zielen am nächsten bringen. Können Ziele nicht direkt erreicht werden, ist auch eine Planung mehrerer Aktionen möglich. Zu dieser Kategorie gehören auch die BDI-Architekturen (Belief-Desire-Intention) [RG91], die Wissen über die Umwelt, die erstrebenswerten Zustände und die verfolgten Absichten enthalten. Eine Implementierung, die sich an der BDI-Architektur orientiert, ist beispielsweise IRMA [BIP91]. Nutzenbasierte Agenten definieren eine Nutzenfunktion, die mögliche Zustände bewertet, und wählen die Aktionen aus, die den größten Nutzen versprechen. Dies hilft auch dann Entscheidungen zu treffen, wenn anderenfalls Unsicherheit über die Erreichung der Ziele herrscht oder Zielkonflikte auftreten.

Aufbauend auf diesen Modellen werden noch lernende Agenten beschrieben, die in der Lage sind in zuvor unbekannten Umgebungen zurechtzukommen. Sie enthalten zusätzlich ein lernendes Element, einen Kritiker und einen Problemgenerator. Das lernende Element bekommt Rückmeldungen vom Kritiker darüber, wie gut der Agent handelt, und verändert entsprechend seinen

Entscheidungsprozess, um zukünftig noch besser zu handeln. Der Problemgenerator schlägt basierend auf bestimmten Lernzielen neue Handlungen vor, mit denen neue Erfahrungen gesammelt werden können.

Es existieren auch andere Ansätze, bei denen die Interaktionen zwischen Agenten im Vordergrund stehen [GK94]. Hierbei geht es um den Austausch semantischer Informationen, die von allen beteiligten Agenten gleichermaßen interpretiert werden müssen. Aufbauend auf den kommunikativen Fähigkeiten dienen Organisationsstrukturen [OPF03] und Koordinationsmechanismen dazu, die Interaktionen und somit auch das Verhalten des gesamten Multiagentensystems zu steuern. Ein Vertreter für Modelle interaktiver Agenten ist MAGSY [Fis93].

Zusätzlich zu den hier beschriebenen Modellen existieren auch hybride Ansätze, die sowohl reaktive als auch deliberative und interaktive Anteile enthalten und so versuchen die Vorteile miteinander zu kombinieren [Mül97]. Beispiele hierfür sind JIAC (siehe Abschnitt 3.4.5) und JACK (siehe Abschnitt 3.4.4).

3.4 Agentenentwicklungsumgebungen

Agentenentwicklungsumgebungen bieten meist mit Konzepten wie Zielen, Fähigkeiten und Sprechakten sowie mit deklarativen Spezifikationstechniken einen höheren Abstraktionsgrad bei der Modellierung und Implementierung von Anwendungen als herkömmliche Programmiersprachen. Wesentlicher Bestandteil einer solchen Entwicklungsumgebung ist die Agentenarchitektur, die aus einer Menge von Software-Modulen besteht. Diese Komponenten realisieren grundlegende Fähigkeiten wie Sprechaktkommunikation, lokales Wissensmanagement, die Verarbeitung eingehender Informationen und die Repräsentation, Ausführung und das Management von Fähigkeiten. Die auf diese Weise bereitgestellten generischen Fähigkeiten verringern den Programmieraufwand, schränken aber die Freiheit bei der Verwendung der Architektur ein. Im Folgenden werden fünf ausgewählte Agentenentwicklungsumgebungen (OAA, JADE, Cougaar, JACK und JIAC) in Hinblick auf das Management genauer untersucht.

3.4.1 Open Agent Architecture

Die Open Agent Architecture^{TM} (OAA)[5] ist ein Forschungsprototyp des Artificial Intelligence Center von SRI International, das die Erstellung von verteilten Systemen bestehend aus kooperierenden Agenten vereinfacht. Die Schwerpunkte von OAA liegen auf der Unterstützung flexibler Interaktionen zwischen den Agenten durch Delegation von Aufgaben oder Übermittlung von Informationen und die Bereitstellung von multimodalen Benutzerschnittstellen.

OAA definiert die in Abbildung 3.6 dargestellte Architektur. Für das Anfragen, Anbieten und Delegieren von Diensten wird die proprietäre Interagent Communication Language (ICL) bereitgestellt, die auch Ausdrücke in natürlicher Sprache verstehen soll. Über eine Hierarchie von Facilitator-Agenten[6] findet die Dienst- und Datenvermittlung statt, indem sie komplexe Aufgaben auf verschiedene Agenten verteilen und deren Aktivitäten koordinieren. Die Agenten kommunizieren also nicht direkt miteinander, sondern kooperieren jeweils durch Ablegen und Auslesen von Informationen im Facilitator.

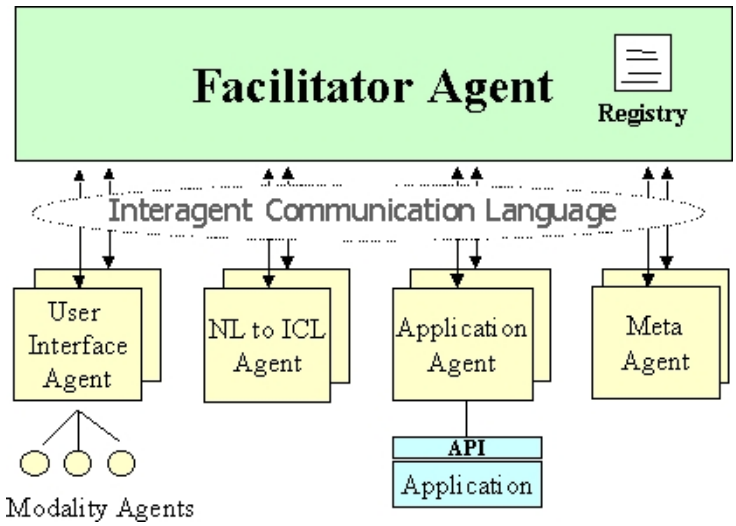

Abbildung 3.6: OAA Architektur [CMM98]

Das Handeln und Kommunizieren der Agenten basiert auf der Programmiersprache Prolog und somit den Prinzipien des Logischen Schlussfolgerns.

[5] http://www.ai.sri.com/ oaa/

[6] Der Facilitator ist ein Blackboard, d.h. ein gemeinsamer Speicher für Agenten. Es besteht eine enge Verwandschaft zu Object Request Brokern.

Die einzelnen Funktionalitäten der Agenten können jedoch in verschiedenen Sprachen (z.B. Java, C++, Prolog) unter Berücksichtigung der entsprechenden Schnittstellen implementiert werden. Obwohl alle Agenten die oben genannten Merkmale aufweisen, ist es durchaus sinnvoll verschiedene Arten von Agenten zu unterscheiden. Hierzu gehören beispielsweise User Interface Agents für multimodale Benutzerinteraktionen[7], Natural Language Agents zur Erzeugung von Anfragen in ICL und Application Agents für die Einbindung anderer Systeme.

Das Agent Development Toolkit fasst mehrere Werkzeuge zusammen. Für das Deployment steht ein Repository zur Verfügung, das wiederverwendbare Agenten verwaltet und eine graphische Zuordnung von Agenten auf Rechner ermöglicht. Der Execution Manager „Start-It“ startet die angegebenen Agenten auf der richtigen Plattform, prüft deren Verbindung zum Facilitator und stellt den Zustand jedes Agenten in einer Oberfläche dar. Funktionen des Managements zur Laufzeit werden nur über einen Monitor-Agenten angeboten. Hierzu gehören das Verfolgen, Debuggen und Darstellen der Kommunikation in einer Agent Community unter Verwendung von Informationen des Facilitator. Für die Wartung eines Systems existiert bisher keine Anwendung. Um ein ausführlicheres Management von Agenten zu betreiben, bedarf es meiner Ansicht nach nur einer Erweiterung des Facilitator um zusätzliche Schnittstellen, da er bereits sämtliche Informationen über die Aktionen und den Zustand der Community enthält.

3.4.2 JADE

Java Agent DEvelopment Framework (JADE)[8] wurde von der CSELT Telecom Italia Group und der Universität Parma entwickelt und ist zurzeit eine der bekanntesten Agentenplattformen. Die Entwicklungsumgebung vereinfacht die Implementierung von Multi-Agentensystemen durch die Bereitstellung einer Middleware. Diese Middleware besteht aus einer Bibliothek von Klassen zur Erzeugung von Agenten, einer verteilten Laufzeitumgebung mit zugehörigen Diensten und Agenten, Werkzeugen zur Unterstützung der Debugging- und Entwicklungsphase und einer Benutzeroberfläche zur Fernwartung der Konfiguration.

Wie in Abbildung 3.7 angedeutet, ist JADE vollständig in Java implementiert

[7] Es handelt sich hierbei um die simultane Nutzung verschiedener Modalitäten. Ein Benutzer zeigt z.B. auf ein Objekt, auf das sich ein gleichzeitig ausgesprochenes Kommando beziehen soll.

[8] http://jade.cselt.it/

und kann daher auf verschiedensten Rechnern laufen. Durch die Berücksichtigung von J2ME werden sogar kleine Endgeräte unterstützt. Die Implementierung der Agenten, d.h. die Definition und Verarbeitung von Wissen sowie die Implementierung des Verhaltens (Behaviour) erfolgt ebenfalls in Java, da hierfür keine zusätzliche Sprache bereitgestellt wird. Für Standardaufgaben wie das Senden und Empfangen von Nachrichten stehen bereits vorgefertigte Fähigkeiten zur Verfügung. Die Modellierung komplexer Aufgaben erfolgt durch die Aggregation mehrerer einfacher Aufgaben.

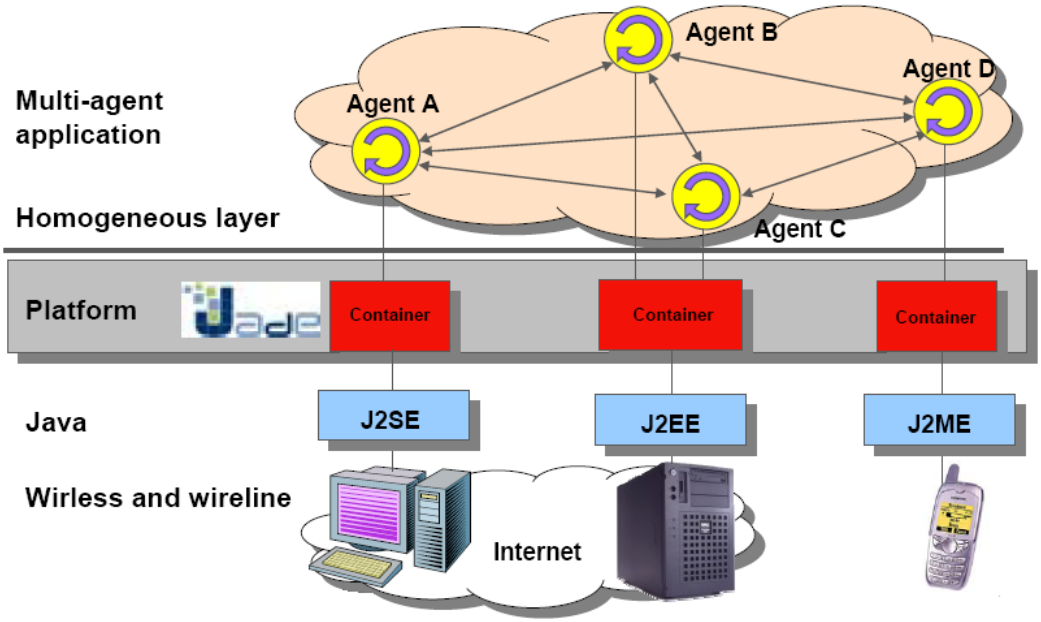

Abbildung 3.7: JADE Architektur [Cai04]

JADE ist konform zum FIPA-Standard (siehe Abschnitt 3.2.1), d.h. alle Agenten bekommen beim Start automatisch eine Global Unique Identification (GUID) und können unter Verwendung der FIPA-ACL miteinander kommunizieren. In JADE besteht eine Agentenplattform[9] aus einem oder mehreren Containern, die jeweils in einer JVM laufen. Einer dieser Container (Main Container) beinhaltet den AMS und den standardmäßigen DF, die automatisch gestartet werden. Jeder Agent befindet sich in genau einem Container und besitzt einen eigenen Thread. Innerhalb einer Agentenplattform werden die Nachrichten (Sprechakte) zwischen Agenten nicht in einer FIPA-konformen Repräsentation sondern als Event-Objekte ohne jegliche Übersetzung ausgetauscht. Befinden sich die Agenten dabei auf unter-

[9] In Abbildung 3.7 ist die Agentenplattform exemplarisch als „Multi-agent application“ dargestellt.

schiedlichen Containern, so erfolgt die Übertragung per Remote Message Invocation (RMI). Alle empfangenen Nachrichten werden in der Warteschlange des Agenten abgelegt.

Basierend auf JADE als reine Middleware zur Erzeugung mobiler, interagierender Agenten existieren bereits einige Erweiterungen. So unterstützt beispielsweise Jadex[10] das in Abschnitt 3.3 beschriebene BDI-Modell. Ein weiteres Beispiel ist das „JessBehaviour“, das die Java Expert System Shell (JESS)[11] in JADE integriert, um die Implementierung von Entscheidungsfindungen und Schlussfolgerungen zu vereinfachen.

Für das Management zur Laufzeit stehen überwiegend Benutzerschnittstellen zur Verfügung. Hierzu gehören der Remote Monitoring Agent (RMA), die DF GUI, der Dummy Agent, der Sniffer Agent und der Introspector Agent. Der RMA liefert die Haupt-Benutzeroberfläche der JADE-Plattform, mit der die Zustände aller auf der Plattform existierenden Agenten verwaltet, beobachtet und kontrolliert sowie auch andere FIPA-konforme Plattformen überwacht werden können. Die DF GUI erlaubt das An- und Abmelden von Agenten sowie die Suche nach und das Ansehen und Ändern der Beschreibungen von Agenten und Diensten. Des Weiteren sind das Zusammenschließen von DFs sowie das Betrachten der anderen DFs möglich. Der Dummy Agent ermöglicht zum Testen von Agenten das Verfassen und Versenden von ACL-Nachrichten an andere Agenten und protokolliert dabei den Nachrichtenaustausch. Die versendeten und erhaltenen Nachrichten können eingesehen, editiert und bei Bedarf gespeichert werden. Der Sniffer Agent beobachtet die Interaktion bzw. den ACL-Nachrichtenaustausch zwischen Agenten. Er erzeugt dynamisch ein Sequenzdiagramm der Nachrichten, mit dem der Benutzer detaillierte Informationen über die einzelnen ACL-Nachrichten abrufen, speichern und für eine spätere Analyse laden kann. Der Introspector Agent erlaubt die Beobachtung, Überwachung und Kontrolle des Lebenszyklus eines laufenden Agenten und seine ausgetauschten ACL-Nachrichten. Er ermöglicht außerdem das Debuggen der Agenten, deren Warteschlange von gesendeten bzw. empfangenen Nachrichten und deren Warteschlange von Behaviours.

Für die Implementierung weiterführender Managementfunktionen existiert beispielsweise der Dienst „EventNotification“. Es können aber auch weitere Dienste erstellt werden.

[10] http://sourceforge.net/projects/jadex
[11] http://herzberg.ca.sandia.gov/jess

3.4.3 Cougaar

Cougaar (Cognitive Agent Architecture)[12] wurde seit 1996 im Rahmen der DARPA (Defense Advanced Research Projects Agency) finanzierten Projekte ALP (The Advanced Logistics Project) und Ultra∗Log entwickelt.

Wie in Abbildung 3.8 zu sehen besteht ein Agent in Cougaar unter anderem aus verschiedenen Plug-Ins, die unabhängig arbeiten und nichts voneinander wissen. Sie bestimmen das Verhalten eines Agenten. Ein Agent stellt den Plug-Ins eine Infrastruktur zur Verfügung, welche die Plug-Ins für die Kommunikation untereinander und mit der Außenwelt benutzen müssen. Es existieren bereits verschiedene Plug-Ins (z.B. für Transport und Visualisierung von Nachrichten, für das Logging und für die Planung von Aufgaben, Workflows, Ressourcen und Ablaufplänen).

Das wichtigste Element eines Agenten ist das Blackboard. Es dient dem Austausch von Informationen zwischen Plug-Ins und dem Agenten nach der Subscribe/Notify Semantik, d.h. ein Agent oder Plug-In kann bestimmte Objekte auf dem Blackboard abonnieren. Vom Plug-In Scheduler aufgerufen kann auf ein Delta-Objekt zugegriffen und somit abgelesen werden, welche Objekte hinzugekommen sind, welche gelöscht und welche geändert wurden. Es werden vier Typen von Objekten unterschieden, die auf dem Blackboard untergebracht werden können: Task (Aufgabe), Asset (Ressource), OrganizationalAsset (Proxy zum anderen Agenten) und PlanElement (Anordnung von Aufgaben).

Cougaar besitzt einen Agent Naming Service der eindeutige Agentennamen erzeugt, die das Zustellen von Nachrichten ermöglichen. Um Agenten zu erreichen, stellt Cougaar einen White-Pages-Dienst bereit, der die Adressen zu den Agentennamen herausfindet. Um die Fähigkeiten von Agenten zu ermitteln, ist ein Yellow-Pages-Dienst vorhanden, der eine gezielte Suche nach Eigenschaften ermöglicht, die durch Attribute beschrieben werden. Der White-Pages und der Yellow-Pages-Dienst unterstützt Hierarchien von Servern.

In der Cougaar Architektur wird ein Persistenz-Mechanismus eingesetzt, der es ermöglicht, dass ein Agent nach einem Fehler, der zu dessen Beendigung führte, wiederhergestellt werden kann. Das bedeutet konkret, dass der Zustand eines jeden Agenten und der Objekte, die der Agent öffentlich gemacht hat, gespeichert werden. Im Fehlerfall lässt sich dadurch der Zustand vor dem Fehler wiederherstellen. Basierend auf der Serialisierung von Agenten wird auch deren Migration unterstützt.

[12] http://cougaar.org/

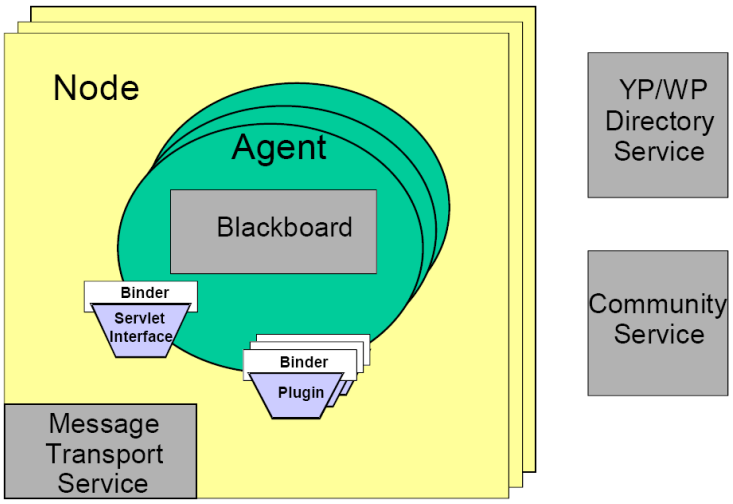

Abbildung 3.8: Cougaar Architektur [HTW04]

Cougaar unterstützt keinen Standard zur Repräsentation von Wissen (weder KQML noch ACL). Das Format der Nachrichten wird durch Plugins bestimmt und hängt von der Anwendung ab. Für den Transport zu anderen Agenten stehen beispielsweise Plug-Ins für RMI, IIOP, SMTP und UDP zur Verfügung. Die Kodierung der Nachrichten erfolgt mittels der Objektserialisierung von Java, da Cougaar selbst komplett in Java geschrieben ist. Zusätzlich zur Kommunikation zwischen Agenten existiert auch eine Unterstützung für Web Services.

Es gibt eine Mobile Edition von Cougaar. Sie wurde mit dem Ziel entwickelt, auch auf kleinen Geräten mit wenig Speicher und Prozessorleistung zu laufen. Als Basis wird J2ME verwendet, was auch die Anzahl der verfügbaren Bibliotheken einschränkt. Die Subscription/Notify-Semantik von Cougaar wurde zwar prinzipiell beibehalten, aber auf verschiedene Aspekte (z.B. Persistenz) wurde verzichtet.

CSMART ist ein Werkzeug zum Erzeugen, Überwachen und Analysieren von Cougaar-Agenten. Es enthält eine graphische Oberfläche zur Konfiguration sowie zum Starten, Überwachen und Anhalten von Agenten. Die Verbindungen zwischen verschiedenen Agenten und Agentengruppen können visualisiert werden. Für das Debugging von Agenten existieren auch schon einige GUIs, die über Servlets realisiert sind.

3.4.4 JACK

JACK^{TM} ist ein kommerzielles Produkt der Agent Oriented Software Group[13]. Es ist eine Entwicklungs- und Laufzeitumgebung für MAS, die über mehrere Rechner verteilt sein können. JACK Agenten basieren auf dem BDI-Ansatz und können sowohl zielorientiert als auch ereignisgesteuert handeln. Das erweiterte JACK Teams Model ermöglicht die Realisierung von teamfähigem Verhalten.

JACK ist komplett in Java implementiert und die Jack Agent Language (JAL) ist eine Programmiersprache, die Java um agentenorientierte Konzepte erweitert. Hierzu gehören Agenten, Fähigkeiten, Ereignisse, Pläne, Wissensbasen sowie ein Ressourcen- und Nebenläufigkeitsmanagement. Der Kern von JACK läuft auch auf kleinen Endgeräten (z.B. PDA), und die komponentenbasierte Architektur ermöglicht eine einfache Integration anderer Software. Mit der Erweiterung JACK WebBot können auch Web-Anwendungen erstellt werden.

Das DCI Netzwerk erlaubt die direkte Kommunikation zwischen zwei Agenten unabhängig vom darunterliegenden Transportmechanismus. Standardmäßig verwendet JACK das Transportprotokoll UDP mit einer darüber liegenden Schicht für die verlässliche Zustellung der Nachrichten. Ein Standard wie beispielsweise FIPA wird nicht unterstützt.

Die JACK Development Environment (JDE) integriert eine Reihe graphischer Editoren, mit denen agentenbasierte Anwendungen entworfen, konfiguriert, gewartet und angepasst sowie in konventionelle Software integriert werden können.

3.4.5 JIAC

Java-based Intelligent Agent Componentware (JIAC)[14] ist ein am DAI-Labor der Technischen Universität Berlin entwickeltes und auf Agententechnologie beruhendes Serviceware-Framework.

Den Kern von JIAC bildet eine skalierbare Komponentenarchitektur, die ein Austausch von Komponenten auch zur Laufzeit erlaubt. Aufbauend auf dieser Komponentenarchitektur existiert eine Kontrollarchitektur in Form von Basiskomponenten, die die grundlegenden Fähigkeiten autonomer, intelligenter,

[13] http://www.agent-software.com
[14] http://www.jiac.de/

kommunizierender und mobiler Agenten integriert. Je nach Bedarf können optional noch weitere Funktionalitäten (z.B. Sicherheits-, Management- oder anwendungsspezifische Komponenten) in die Architektur eingebunden werden (siehe Abbildung 3.9).

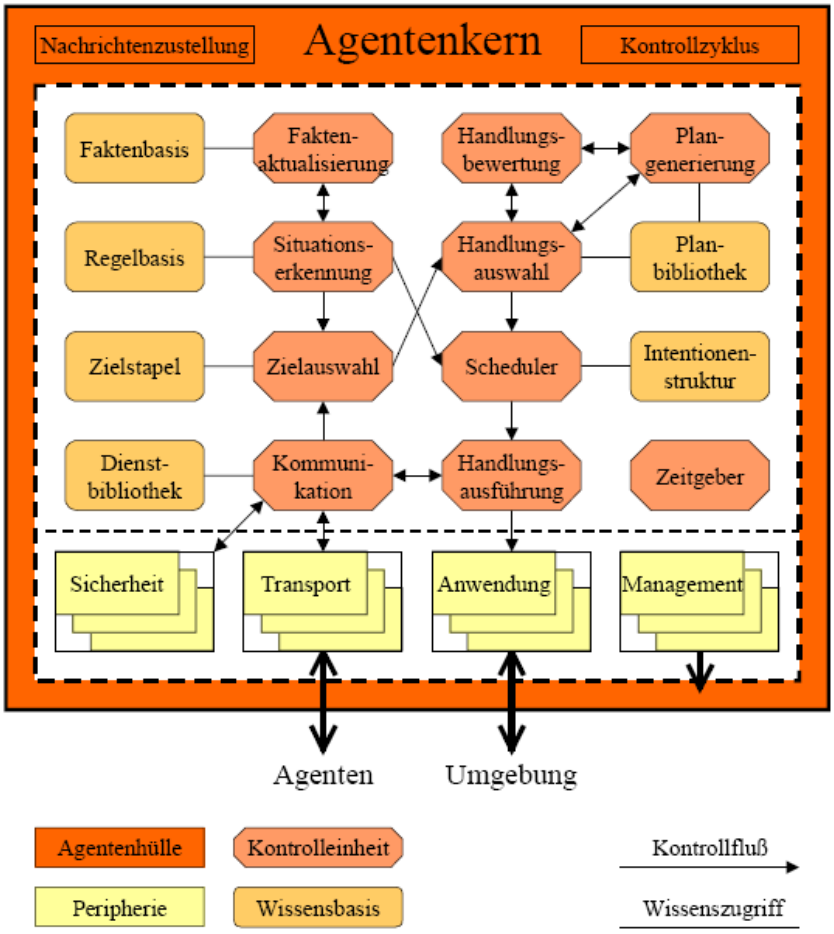

Abbildung 3.9: JIAC Standardarchitektur [Ses02]

Zu den Basiskomponenten gehören beispielsweise die Wissensverarbeitung, Zielverfolgung, Planausführung und Kommunikation, die das reaktive, zielgerichtete und interaktive Verhalten der Agenten gemäß dem in Abschnitt 3.3 beschriebenen BDI-Ansatz umsetzen. Sämtliche Aktivitäten werden dabei durch Ziele ausgelöst und kontrolliert. Ein Ziel beschreibt durch Verwendung von Ontologien einen Zustand, den der Agent entweder durch Nutzung lokaler Aktionen oder durch Starten einer Dienstnutzung versucht zu erreichen. Alle Dienstnutzungen werden über ein generisches Metaprotokoll abgehandelt, das zusätzlich eine Aushandlungsphase enthält und spezifische Verhandlungs-

und Dienstnutzungsprotokolle ausführt. Die Kommunikation basiert auf der Agent Communication Language (ACL) von FIPA (siehe Abschnitt 3.2.1).

Eine Agentenplattform fasst mehrere Agenten in einer Java Virtual Machine (JVM) zusammen. Agentenplattformen werden jeweils durch einen speziellen Manageragenten verwaltet, der mindestens die Funktion des AMS übernimmt.

Für den Service-Engineer steht eine komplette Entwicklungsumgebung bestehend aus Spezifikationssprachen und Werkzeugen in einer umkleidenden Software-Engineering-Methode bereit, mit der sich Domänenwissen, Dienste und Agenten spezifizieren lassen. In der Analyse-Phase werden Agententypen und ihre Rollen und Abhängigkeiten untereinander zunächst abstrakt beschrieben und diese Beschreibung dann beim Design formal konkretisiert, indem Ontologien, Fakten, Ziele, Regeln, Dienste und Protokolle in der JIAC Agent Description Language (JADL) spezifiziert werden. Für die Einbindung existierender Java-Bibliotheken können anwendungsspezifische Komponenten programmiert werden. In der Deployment-Phase erfolgt das Zusammenstellen und Konfigurieren der Agenten aus den benötigten Komponenten und Wissenselementen sowie die Zuordnung der Agenten zu Agentenplattformen. Schließlich wird das laufende System zur Evaluierung getestet.

3.4.6 Zusammenfassung

Agentenentwicklungsumgebungen heben die Agentenprogrammierung durch Bereitstellung von Agentenarchitektur, Agentenspezifikationssprache und integrierenden Entwicklungswerkzeugen auf eine höhere Ebene.

Mehrere an der BDI-Agentenarchitektur orientierte Ansätze existieren, so zum Beispiel Jadex, JACK und JIAC. In Systemen wie JACK und JIAC erfolgt die Programmierung in High-Level-Spezifikationssprachen. Diese Sprachen unterstützen, zusammen mit den auf ihnen operierenden mächtigen Architekturkomponenten, charakteristische Eigenschaften wie zielgerichtetes Verhalten, Autonomie und Kooperation.

Nicht behandelt wurden an dieser Stelle die sogenannten kognitiven Architekturen, die auf Wissensrepräsentation und Reasoning fokussieren. Derartige Architekturen zielen auf die Erreichung intelligenten Verhaltens durch den Einsatz von Methoden der Künstlichen Intelligenz ab, vernachlässigen aber die Aspekte der Agenteninteraktion und werden ausschließlich in Forschungsinstituten eingesetzt.

In Bezug auf die Kommunikation bieten JADE und JIAC die mächtigsten Mechanismen, da neben der Sprechaktkommunikation auch Interaktionsprotokolle bereitgestellt werden. Beides führt in die Kommunikation eine semantische Ebene ein, die Fehlinterpretationen vermeiden hilft.

In OAA, JADE und JIAC ist ein wesentlicher Aspekt die Bereitstellung und Erbringung von Diensten. Sie führen Agenten als Dienstanbieter und Dienstnutzer ein. JADE und JIAC orientieren sich dabei an den Standards der FIPA.

Die meisten Entwicklungsumgebungen beinhalten grafische Werkzeuge zur Erstellung, Kontrolle und/oder Wartung von Agentensystemen. Die kommerziellen Produkte wie JACK bieten gewöhnlich eine umfangreichere Auswahl an Werkzeugen an. Im Bereich des Managements stehen neben einzelnen Werkzeugen oftmals auch APIs für die Überwachung und Administration der Agenten zur Verfügung. Diese APIs sind von System zu System sehr verschieden und unterliegen keiner Standardisierung.

3.5 Grenzen der Agententechnologie

Die Agententechnologie ist kein Allheilmittel für sämtliche Softwareentwicklungsaufgaben. Stattdessen stellt sie ein neues Programmierparadigma für komplexe und verteilte Systeme dar, die es erlaubt das System auf einer höheren Abstraktionsebene zu beschreiben. Durch eine entsprechende Methodologie, geeignete Sprachkonzepte und eine Laufzeitumgebung, die die verschiedenen Konzepte in zugehörige Verhaltensmuster umsetzt, ist die Komplexität leichter in den Griff zu bekommen. Als Ergebnis dessen kann im Durchschnitt die Entwicklungszeit reduziert und die Anzahl der Fehler verringert werden.

Dieser Ansatz hat allerdings auch seinen Preis. Wie schon bei der Verwendung höherer Programmiersprachen führen auch die besonderen Eigenschaften agentenbasierter Systeme (siehe Abschnitt 3.1) zu einem erhöhten Rechenaufwand während der Laufzeit, da die ausgeführten Aktionen im Detail nur begrenzt auf die konkrete Problemstellung optimiert werden können. Aus diesem Grund eignet sich die Agententechnologie nur für ausreichend komplexe Probleme, bei denen das Zeitverhalten nicht im Vordergrund steht. Hierzu gehören beispielsweise der Bereich der Telekommunikation oder des Supply Chain Management, jedoch nicht unbedingt Reaktionsspiele oder Geräte-

treiber. In [WJ95] zählt der Autor einige Fallstricke auf, die beim Einsatz agentenbasierter Systeme beachtet werden sollten.

Die Herausforderung beim Management agentenbasierter Systeme liegt demzufolge auch nicht in dessen Schnelligkeit, sondern darin die Komplexität bei der Überwachung und Steuerung dieser Systeme in den Griff zu bekommen.

Kapitel 4
Bewertung

In dieser Arbeit wurde der Stand der Technik von zwei großen Bereichen untersucht. Einerseits wurden Technologien für das Management verteilter Systeme betrachtet, um herauszufinden inwiefern Teile dieser Technologien auch für das Management agentenbasierter Systeme geeignet sind. Andererseits wurde der Bereich der Agententechnologie genauer untersucht, um sowohl existierende Vorarbeiten für das Management dieser Systeme aufzudecken, als auch Besonderheiten dieser Technologie aufzuzeigen, die es beim Management zu berücksichtigen gilt.

Wesentliche Standardisierungen für das Management verteilter Systeme betreffen das Netzwerkmanagement. Hierzu können die ISO/OSI Standards, die ITU Empfehlungen zum TMN und die verschiedenen RFCs der IETF genannt werden, wobei sich TMN und das SNMP der IETF stark am OSI-Management orientieren. Das Problem bei diesen Spezifikationen ist, dass die höherwertigen Managementfunktionen entweder gar nicht, nicht detailliert oder nicht weit genug spezifiziert oder zu stark auf die Anforderungen der Netzwerkdienste zugeschnitten sind und sich somit nicht ohne Weiteres auf agentenbasierte Systeme anwenden lassen. So werden im Funktionsmodell des OSI-Managements zwar verschiedene Funktionsbereiche (siehe FCAPS) definiert, jedoch nicht vollständig abgedeckt. Beispielsweise werden für das Abrechnungsmanagement vorerst nur die Messung der Ressourcennutzung und beim Leistungsmanagement nur die Überwachung der Antwortzeiten betrachtet. Das TMN beschreibt zwar eine umfassendere Sicht auf das Abrechnungsmanagement, bleibt aber wie auch das TM Forum in den Ausführungen sehr oberflächlich. Die IETF ist stark auf die Abrechnung von Netzwerkdiensten ausgerichtet. Ein interessanterer Ansatz schien meiner Meinung nach die FCR der OMG zu sein, jedoch konnten sich die dort definierten Schnittstellen

genauso wie TINA oder das CAS des M3I Konsortiums nicht so recht durchsetzen. Das Informationsmodell des OSI-Managements nutzt einen objektorientierten Ansatz und die ASN.1-Syntax zur Modellierung von Ressourcen. Speziell für den Austausch von Dienstnutzungsdaten wurde das auf MIME basierende ADIF und das auf XML-Schema basierende IPDR definiert, wobei Letzteres erfolgsversprechender zu sein scheint, da nicht zuletzt alle relevanten Hersteller daran beteiligt sind. Alle drei Formate betrachten leider keine neuartigen Ontologiekonzepte, die semantische Zuordnungen und Schlussfolgerungen ermöglichen. Bei der Auswahl der zu verwaltenden Ressourcen oder Dienstnutzungsereignisse spielten die Besonderheiten agentenbasierter Systeme bisher keine Rolle. Die häufigste Verwendung dieser Modelle bezieht sich auf die Beschreibung von klassischen Netzwerkkomponenten und deren Nutzung. Das Kommunikationsmodell des OSI-Managements, dass die Dienste und Protokolle für den Austausch von Managementinformationen zwischen dem überwachenden und überwachten System beschreibt, ist ebenfalls sehr allgemein gehalten.

Gleiches gilt übrigens auch für die unter dem Namen WSDM zusammengefassten Managementspezifikationen aus dem Bereich der Web Services, die erst in den vergangenen Jahren erstellt wurden. Der wesentliche Unterschied zum OSI-Management besteht darin, dass diese Spezifikationen auf ihrer eigenen Kommunikationstechnologie aufbauen und somit die Syntax für die Beschreibung der Ressourcen auf XML basiert, als Protokoll SOAP statt CMIP verwendet wird und keine generischen Aktionen wie Attribut lesen oder ändern angeboten werden, sondern die Fähigkeiten der Ressource explizit beschrieben werden. Wie eine solche Beschreibung im Detail auszusehen hat ist aber weitgehend offen.

Aus dem Bereich der Agententechnologie existieren mit MASIF und FIPA nur Managementspezifikationen, die sich mit dem Lebenszyklus der Agenten und somit mit der Verwaltung auf Ebene der Multiagentensysteme beschäftigen. Spezifikationen, die die Überwachung und Steuerung des Verhaltens von Agenten beschreiben, sucht man vergebens. Dies liegt vor allem daran, dass keine einheitliche Definition und Architektur für Agenten existiert, sondern zur Zeit in der Forschung noch verschiedenste Ansätze aufgrund unterschiedlicher Anwendungsfelder verfolgt werden. Die untersuchten Agentenframeworks bieten meist nur rudimentäre und proprietäre Schnittstellen für das Management ihrer Agenten an.

Trotz der verschiedenen Ansätze lassen sich einige Besonderheiten agentenbasierter Systeme hervorheben. Das wesentlichste Merkmal ist die Autonomie der Agenten, die sie in die Lage versetzt in gewissem Grad selbstständig

(d.h. ohne direkten Eingriff von außen) Entscheidungen zu treffen und Handlungen auszuführen und dabei Kontrolle über sich selbst zu besitzen. Ein häufiges Merkmal der Agenten stellt auch ihre Proaktivität dar, die es ihnen ermöglicht im voraus zu planen und eigenständig die Initiative zu ergreifen. Der häufigste Ansatz zur Umsetzung dieser Eigenschaften stellt das BDI-Modell dar, bei dem aus den durch Sensoren oder Schlussfolgerungen gewonnenen Annahmen oder Wissen über die Umgebung und den eigenen Zustand geeignete Ziele ausgewählt werden, die dann eine Zeit lang verfolgt werden. Zur Erreichung dieser Ziele werden entsprechende Pläne ausgewählt, mit der Absicht durch deren Ausführung den Zielen ein entscheidendes Stück näher zu kommen. Da keiner der untersuchten Managementansätze diese Eigenschaften berücksichtigt, muss ein neuer Ansatz gefunden oder einer der bestehenden Ansätze entsprechend erweitert werden, um die Ursache unerwünschten oder anarchischen Verhaltens erkennen und die Auswirkungen des komplexen Systems an veränderte Bedürfnisse anpassen zu können. Der nächste Teil dieser Arbeit beschäftigt sich mit der Konzeption eines neuen Ansatzes, der aber bestehende Ansätze weitgehend berücksichtigt bzw. integriert.

Teil III

Konzeption

Kapitel 5
Analyse und Modellbildung

Eine Managementinfrastruktur ist ein System zur Verwaltung von Anwendungssystemen und ihrer Bestandteile. Es ist somit immer vom Aufbau und der Funktionsweise des zu verwaltenden Systems abhängig. Beim Management agentenbasierter Systeme müssen also die Besonderheiten der zu betrachtenden Agenten berücksichtigt werden. Aus diesen Besonderheiten ergeben sich zusätzliche Anforderungen an das Management, die zusammen mit den daraus resultierenden Aufgaben im Abschnitt 5.1 spezifiziert werden.

Da die Kooperation zwischen Agenten gewöhnlich über Dienstnutzungen stattfindet, ist für den Entwurf der Managementinfrastruktur neben der Betrachtung der Agententechnologie (siehe Kapitel 3) auch eine Definition des zu verwendenden Dienstmodells erforderlich. Die Managementinfrastruktur MIAS beschränkt sich also auf agentenbasierte Systeme, die das im Abschnitt 5.2 beschriebene Dienstmodell berücksichtigen.

Nach der Analysephase wird in Abschnitt 5.3 das Modell der Managementinfrastruktur MIAS vorgestellt, das aus mehreren Ebenen besteht. Jedes der einzelnen Ebenen definiert eine Menge von Funktionen, die es zu realisieren gilt.

Aufgrund der räumlichen Verteilung der Managementinfrastruktur und den Anforderungen an Robustheit, Skalierbarkeit, Offenheit und Sicherheit bietet es sich an, MIAS selbst als agentenbasiertes System zu entwickeln. Ein erster Schritt für die Entwicklung eines solchen Systems ist der Entwurf von Ontologien als Teil der Konzeptionsphase bei der agentenorientierten Softwareentwicklung (vgl. [Hes]).

Die Basisontologie von MIAS, die die Bestandteile eines agentenbasierten Systems sowie die Dienste und Dienstnutzungen formal beschreibt, kann aus der

informellen Beschreibung des zu verwendenden Dienstmodells abgeleitet werden (siehe Abschnitt 5.4). Diese Ontologie definiert also die grundlegenden Datenstrukturen, die von allen Managementfunktionen in MIAS verwendet werden sollen.

5.1 Aufgaben der Managementinfrastruktur

Management basiert auf der Überwachung und Manipulation der für den operativen Betrieb eines Systems (Hard- und Software) notwendigen Kontrollparameter. Gerade bei agentenbasierten Systemen ergeben sich zusätzliche Anforderungen an das Management. Deutlich werden diese Anforderungen durch das Betrachten einer Auswahl der bereits in Abschnitt 3.1 aufgezählten Eigenschaften der Agenten:

- Autonomie
- Wissen/Intelligenz
- komplexe Kommunikation
- Kooperationseigenschaft
- Mobilität

Die Expertise eines Agenten ergibt sich aus seinem Wissen sowie seinen Fähigkeit dieses zu verarbeiten. Somit kann ein agentenbasiertes System durchaus eine erhebliche Menge an Wissen enthalten. Der Wissensaustausch wie auch die Interaktion zwischen den Agenten erfolgt über Sprechakte mit wohldefinierten Inhalten auf einer semantisch hohen Ebene. Dadurch ist es möglich, die Funktionalität eines agentenbasierten Dienstes durch Kooperation mehrerer Agenten mit unterschiedlichen Autonomiegraden zu erbringen. Diese wichtige Eigenschaft hat allerdings zwei entscheidende Auswirkungen zur Folge: Zum Einen handelt es sich gewöhnlich um Systeme mit einer sehr starken räumlichen Verteilung, deren Überwachung durch die Mobilitätseigenschaft der Agenten und der damit verbundenen Ortsunabhängigkeit von Dienstfunktionalitäten noch zusätzlich erschwert wird. Zum Anderen zeichnen sich agentenbasierte Systeme durch eine hohe Dynamik aus, indem auf

der Basis des vorhandenen Wissens vermehrt Entscheidungen zur Laufzeit getroffen werden, die meist nicht vorhersehbar sind und demzufolge nachvollziehbar sein sollten. Bei Betrachtung dieser Gegebenheiten wird deutlich, dass eine Managementinfrastruktur u.a. folgende Aspekte zu berücksichtigen hat:

- *Nachvollziehbarkeit*: Der Ablauf eines verteilt abgearbeiteten Dienstes auf der Basis autonom agierender Agenten lässt sich in verschiedenen Detaillierungsgraden verfolgen. Notwendig ist hierfür eine Generierung, Sammlung und Aggregation entsprechender Informationen zur Laufzeit. Das Überwachen kann zeitlich wie folgt unterschieden werden:
  - *Introspektion zur Dienstlaufzeit*: Während der Dienstabarbeitung werden Informationen zu den Aktionen und Zuständen verteilter und insbesondere mobiler Agenten gesammelt und ausgewertet. So kann beispielsweise ermittelt werden, ob ein mobiler Agent seine Aufgaben bereits erledigt hat oder noch daran arbeitet. Der Detaillierungsgrad und der Kontext der Informationen lässt sich jeder Zeit an den Fokus der Überwachung anpassen.
  - *Rekonstruktion nach Dienstablauf*: Nach Ablauf eines Dienstes kann das Verhalten der beteiligten Agenten anhand der gesammelten Informationen analysiert und ausgewertet werden. Der Detaillierungsgrad der Informationen lässt sich allerdings nachträglich nicht mehr verändern.
- *Lokalisierbarkeit*: Durch die Mobilitätseigenschaft der Agenten müssen entsprechende Mechanismen zum Auffinden von Agenten und Diensten bereitgestellt werden.
- *Anpassungsfähigkeit*: Für die Anpassung an neue Bedingungen und die Schaffung neuer Eigenschaften des Systems sind Änderungen an den entsprechenden Bestandteilen des Agentensystems nötig. Dazu werden während der Laufzeit des Systems den unter Umständen abstrakt formulierten Anforderungen jeweils eine Menge elementarer Aktionen zugeordnet, die anschließend auszuführen sind.
- *Kooperationsfähigkeit*: Ein Agent muss zum Erreichen seiner Ziele in der Lage sein Dienste bei anderen Agenten nutzen zu können. Insbesondere ist hierbei die Suche nach geeigneten Diensten und Dienstanbietern sowie die Bereitstellung von Informationen zur Gewährleistung der Kommunikation zwischen den Agenten zu beachten.

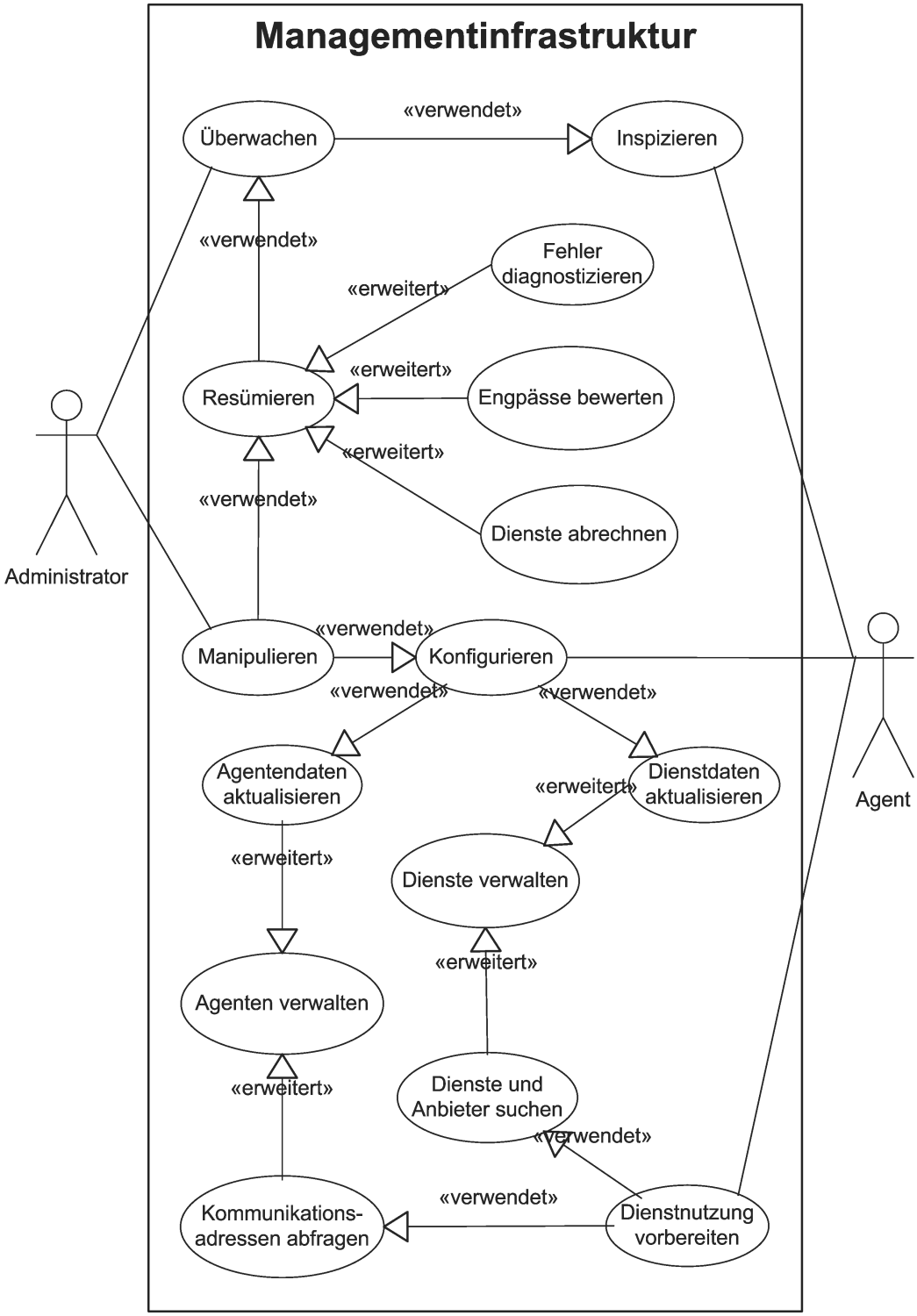

Abbildung 5.1: Anwendungsfalldiagramm zum Management von MAS

In diesem Abschnitt wird nun ein Überblick über die Funktionalitäten gegeben, die für eine Managementinfrastruktur notwendig und nützlich sind. Das in Abbildung 5.1 dargestellte Anwendungsfalldiagramm zeigt die beteiligten Akteure und ihre Schnittstellen zur Managementinfrastruktur, die die in Kapitel 1 genannten Probleme lösen soll.

Der Administrator oder der Eigentümer eines agentenbasierten Systems möchte seine Agenten und Dienste überwachen können, um die Einhaltung der Anforderungen und die Erreichung der Ziele überprüfen zu können. Dazu werden bestimmte Daten zugrundegelegt, die während der Introspektion der entsprechenden Agenten gesammelt worden sind (siehe Abschnitt 6.2). Diese Informationen werden sinnvoll aggregiert, bevor sie über eine Benutzerschnittstelle präsentiert werden oder als Grundlage für eine laufende Auswertung dienen. Diese Auswertung kann verschiedene Aspekte des Managements umfassen. Eine Fehlerdiagnose ermittelt z.B. die Ursache dafür, dass die in der Dienstaushandlung festgelegten Parameter nicht eingehalten wurden oder eine Ressource nicht die gewünschten Ergebnisse liefert. Eine rechtzeitige Erkennung und Bewertung von Engpässen ist Voraussetzung, um weiterhin die ausgehandelten Dienstparameter und definierten Qualitätsanforderungen garantieren zu können. Zur Deckung aller anfallenden Kosten ist eine Abrechnung der Ressourcennutzungen unter Verwendung kundenindividueller und dienstorientierter Tarife unerlässlich (siehe Kapitel 7).

Durch entsprechende Manipulation der am System beteiligten Agenten ist basierend auf den zuvor ausgewerteten Daten eine Behebung von Fehlern, die Überwindung von Engpässen und eine markt- bzw. qualitätsorientierte Anpassung sowie nutzungsspezifische Verwendung von Tarifen möglich. Die Manipulationen werden entweder manuell durch den Administrator über eine Benutzerschnittstelle oder automatisch durch definierte Regeln innerhalb der Managementinfrastruktur angestoßen. Dabei erfolgt eine Aufspaltung der groben Änderungspläne in einzelne Konfigurationsaktionen, die von den entsprechenden Agenten ausgeführt werden (siehe Kapitel 8).

Die während der Konfiguration geänderten Agenten- und Dienstdaten müssen bei den zugehörigen Instanzen, die diese Daten verwalten, aktualisiert werden (siehe Abschnitt 6.4). Vor dem Starten einer Dienstnutzung kann ein Agent über das Dienstverzeichnis nach einem passenden Dienst und Anbieter suchen und sich über das Agentenverzeichnis die Adressen des gefundenen Kommunikationspartners geben lassen.

5.2 Eigenschaften von agentenbasierten Dienstumgebungen

Wie bereits in Abschnitt 3.3 erwähnt spielen Dienste bei der Kooperation zwischen den Bestandteilen eines agentenbasierten Systems eine entscheidende Rolle. Aus diesem Grund wird nun ein allgemeines Modell für die Zusammensetzung, die Bereitstellung sowie die Nutzung von agentenbasierten Diensten vorgestellt, das sich an den bekannten AOT-Ansätzen orientiert und als Grundlage für die später vorgestellten Managementkonzepte dient (vgl. [GSTJ98]).

5.2.1 Zusammensetzung der Dienste

Mit Hilfe von Agenten und ihrer Bestandteile lassen sich auch komplette Dienstplattformen und darauf aufbauende Anwendungen realisieren.

Abbildung 5.2: Ebenen einer agentenbasierten Anwendung

In Abbildung 5.2 sind drei Ebenen solch einer agentenbasierten Anwendung dargestellt. Die unterste Ebene bilden dabei die Agenten mit ihren verschiedenen Bestandteilen und Eigenschaften. Durch Kooperation der Agenten können komplexe Anwendungs- und Managementdienste entstehen. Die Vermittlung dieser Dienste erfolgt durch Agentenplattformen (AP), die wiederum aus verschiedenen spezialisierten Diensten bestehen und eine Infrastruktur für das Management von Dienstnutzungen bereitstellen. Da sich das Management über alle drei Ebenen hinweg erstreckt, werden diese Ebenen nun genauer untersucht.

Ebene der Agenten

Der Aufbau eines Agenten ist von der zugrundeliegenden Agentenarchitektur abhängig. So besitzen Agenten, die beispielsweise auf dem BDI-Modell basieren, eine Menge an Wissen, Zielen und Intentionen (siehe Abschnitt 3.3). Die einzelnen Funktionen eines Agenten können dabei modular aufgebaut und zur Laufzeit austauschbar sein. Unter dem Aspekt des Managements gilt es daher, diese Bestandteile zu kontrollieren und gegebenenfalls zu manipulieren.

Die Aufteilung von einzelnen Funktionen auf mehrere Agenten ist wie beim Entwurf jedes verteilten Systems unter Zuhilfenahme folgender Kriterien jeweils individuell zu entscheiden:

1. Kommunikationsaufwand
2. Skalierbarkeit und Modularität
3. Lastverteilung und Redundanz
4. Autonomie

So ist zu klären, mit welchem zusätzlichen Kommunikationsaufwand die Verteilung von Funktionalitäten auf einzelne Instanzen erkauft wird. Durch die Gruppierung ähnlicher oder zusammengehöriger Funktionen kann ein gewisses Maß an Modularität geschaffen werden. Das Hinzufügen oder Entfernen von Agenten und ihrer Funktionen ermöglicht die Skalierbarkeit des Systems. Ein Grund für die Verteilung von identischen Funktionalitäten kann in der Schaffung von Redundanz zur Verringerung der Ausfallwahrscheinlichkeit oder zur Erreichung einer Lastverteilung bestehen. Nicht zuletzt muss auch die Frage nach der Autonomie geklärt werden. So ist es beispielsweise nicht sinnvoll, sicherheitsrelevante Aufgaben an vertrauensunwürdige Anbieter zu delegieren.

Ebene der Agentenplattformen

Hinsichtlich der Dienstumgebung stellt eine Agentenplattform (AP) ein Strukturierungselement dar, da der Ort, an dem Dienste angeboten werden, bei deren Vermittlung und Auswahl berücksichtigt werden kann. Hauptaufgabe der APs ist allerdings die Bereitstellung allgemeiner Funktionalitäten für die Laufzeitumgebung sowie die Adressierung der Agenten. Diese nachfolgend aufgelisteten Grundfunktionen werden durch den sogenannten Manageragenten (MA) erbracht, der somit auf jeder AP vorhanden sein muss und dort eine Sonderstellung einnimmt.

- Registrierung der Agenten
- Agentenverwaltung
- Agentenmigration
- Zustellung von Sprechakten
- Dienstvermittlung
- Agentenüberwachung

Tritt ein Agent in die AP ein, so wird er in einem ersten Schritt beim MA registriert. Der MA kennt somit stets die Art und Anzahl der aktuell auf seiner AP verfügbaren Agenten und dessen Konfiguration. Der Lebenszykluszustand dieser Agenten kann über den MA verwaltet und verändert werden. Eine Migration der Agenten wird ebenfalls unterstützt. Ein Agent kann Sprechakte über den MA zustellen lassen, falls der Kommunikationspartner temporär nicht erreichbar ist. Bei der Dienstvermittlung handelt es sich um einen Suchdienst (Gelbe-Seiten-Dienst), mit dessen Hilfe Anwendungsdienste auf der Basis gewisser Eigenschaften ermittelt werden. Zusätzlich zu diesen bereits in FIPA definierten Infrastrukturfunktionalitäten (siehe Abschnitt 3.2.1) kann der MA auch den Aufenthaltsort und die Handlungen der auf der lokalen AP erzeugten Agenten überwachen (siehe Abschnitt 6.6).

Ebene der Dienste

Hier sind vor allem die Managementdienste interessant. Sie werden genauso behandelt wie Anwendungsdienste, d.h. sie sind gleichermaßen aufgebaut und laufen in derselben Umgebung. Sie unterscheiden sich nur in den auszuführenden Handlungen, die durch entsprechende managementspezifische Komponenten ermöglicht werden. Managementdienste können entweder allgemeine Dienste zur Administration der Agenten oder aber spezielle, auf einen konkreten Anwendungsdienst bezogene Verwaltungs- und Steuerungsdienste sein. Wie bereits erwähnt erstrecken sich also die Aufgaben des Managements über alle Ebenen einer agentenbasierten Anwendung. Demzufolge lassen sich auch bei den Managementdiensten folgende drei Bereiche unterscheiden:

- *Agentenmanagement*: Das Agentenmanagement erlaubt einen Eingriff in die Agenten, die die Basisfunktionalitäten des Systems darstellen. Voraussetzung hierfür ist eine Modellierung des Wissens bzw. der Fähigkeiten von Agenten, um über andere Agenten reflektieren zu können. Auch das Management selbst sollte administrierbar sein. Jeder Agent sollte sich aber aufgrund seiner Autonomie gegen eine Beobachtung sperren können. Bei dem Agentenmanagement sind Basismechanismen (siehe Kapitel 6) zu berücksichtigen, die für die unter FCAPS zusammengefassten Funktionalitäten notwendig sind. Dabei sind abhängig von der jeweiligen Agentenarchitektur entsprechende Komponenten vorzusehen.

- *Plattformmanagement*: Beim Plattformmanagement sind aufgrund der Middlewarefunktion mehrere Akteure mit unterschiedlichen Rollen involviert. Dadurch ergibt sich ein stärkeres Zusammenspiel der beteiligten Managementaktivitäten (z.B. Abrechnung, Sicherheit). Das Management umfasst hier nicht nur die Steuerung der aktiven Dienste sondern auch deren Erzeugung und Installation. Da die Plattform aus spezialisierten Diensten und letztlich Agenten besteht, greift das Management der Plattform auf Funktionalitäten des Agenten- und Dienstmanagements zurück.

- *Dienstmanagement*: Zum Dienstmanagement gehören alle administrativen Aktionen zur Kontrolle der reinen Dienstfunktionalität. Hier sind vor allem die unterschiedlichen Benutzerrollen und deren spezifische Anforderungen an die Dienste zu berücksichtigen (siehe Abschnitt 1.1).

5.2.2 Die Bereitstellung eines Dienstes

Im Rahmen der Bereitstellung eines agentenbasierten Dienstes werden ähnlich zu anderen Diensten folgende Phasen durchlaufen:

1. Konstruktionsphase
2. Bereitstellungsphase
3. Nutzungsphase
4. Terminierungsphase.

Zur Konstruktionsphase eines Dienstes gehört die Definition seiner Funktion und der beteiligten Agenten. Hierunter fällt insbesondere der Designprozess, die Implementierung sowie das Testen des entwickelten Dienstes. In der Testphase werden mögliche Anwendungsfälle vor der eigentlichen Inbetriebnahme des Dienstes simuliert. Sie ist demzufolge nicht mit dem bereits erwähnten Fehlermanagement gleichzusetzen, das für die Behandlung von Ausnahmefällen bei laufenden Diensten zuständig ist. Für die Testphase von Diensten werden geeignete Entwicklungsumgebungen benötigt.

Zur Bereitstellungsphase gehören die Erzeugung der Agenten auf den entsprechenden APs sowie ihre Registrierung zum Ermöglichen gegenseitiger Dienstnutzungen. Durch Berücksichtigung von Skalierbarkeit und Modularität kann während der Konstruktions- und Bereitstellungsphase auch auf bereits vorhandene und aktive Agenten zurückgegriffen werden. Somit kann die Dienstbereitstellung auch mit einem Verteilungs- und Initialisierungsprozess verglichen werden.

Die Nutzungsphase umfasst sowohl die Inanspruchnahme der Dienste als auch die Ausführung von Aktivitäten zu deren Wartung.

Während der Terminierungsphase wird der Dienst aus seiner Laufzeitumgebung entfernt und ist anschließend für eine Nutzung nicht mehr verfügbar. Dieser Vorgang kann auch durch die Entfernung der anbietenden Agenten oder wesentlicher Bestandteile aus ihnen erfolgen.

In MIAS werden vor allem die Managementfunktionen berücksichtigt, die für die Administrierung von Diensten in den Phasen zwei bis vier sinnvoll sind.

5.2.3 Die Nutzung eines Dienstes

Während der Dienstinanspruchnahme werden die Zugriffsphase und die eigentliche Benutzungsphase durchlaufen. Die Zugriffsphase umfasst sowohl die Lokalisierung als auch den Aufruf des Dienstes. Dabei kann auf die Fähigkeiten des Manageragenten zurückgegriffen werden, der die Funktion eines Dienstvermittlers übernimmt. Für den eigentlichen Aufruf eines Dienstes werden zwei Bedingungen vorausgesetzt, die ggf. durch Nutzung entsprechender Managementdienste zu erfüllen sind:

- Der Name bzw. die Beschreibung des Dienstes ist bekannt.
- Der Anbieter bzw. seine Adresse ist bekannt.

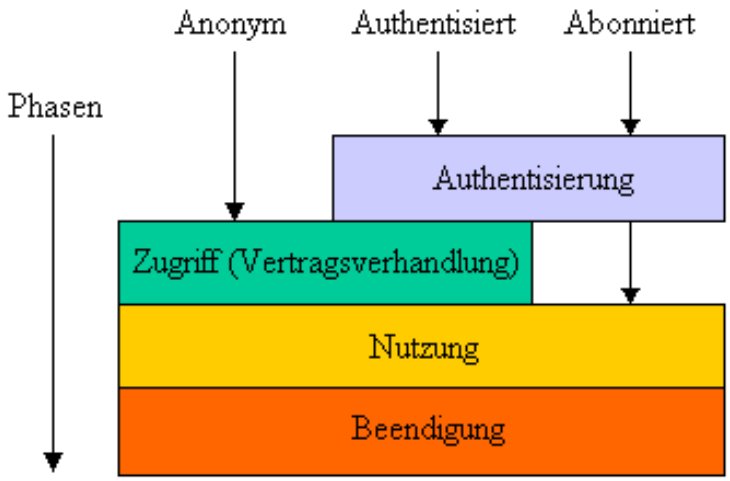

Abbildung 5.3: Aspekte der Dienstinanspruchnahme

Nach der Zugriffsphase findet die Benutzungsphase eines Dienstes statt (siehe Abbildung 5.3). Bei den Benutzungsaspekten werden alle Modalitäten unterschieden, die für die gesamte Phase der Dienstinanspruchnahme relevant sind.

Zugriffsphase

Bei der Benutzung eines Dienstes ist der Bekanntheitsgrad des Nutzers zu unterscheiden. Hier stehen möglicherweise ein anonymer Zugriff oder ein Zugriff durch einen bekannten Benutzer zur Auswahl. Im nicht-anonymen Fall findet vor der Dienstnutzung eine Authentisierung des Benutzers durch den Anbieter statt. Dabei spielt es keine Rolle, ob es sich um eine einfache Authentisierung bzw. Registrierung des Benutzers oder um ein Abonnieren des Dienstes handelt. Für einen abonnierenden Nutzer werden die zur regelmäßigen Nutzung des Dienstes notwendigen Ressourcen bereitgestellt. Nachfolgend werden Dienstnutzer als „authentisiert“ bezeichnet, wenn sie zwar authentisiert sind aber kein Abonnement für den Dienst besitzen.

Der Zugriff auf Dienste kann außerdem sowohl online als auch offline erfolgen. Letzteres lässt sich aus Sicht eines menschlichen Nutzers über mobile Agenten oder über sitzungsbasierte Benutzerschnittstellen erreichen. Desweiteren spielen die Sicherheitsbestimmungen und Zugriffsrechte eine Rolle. So ist die Benutzung eines Dienstes nur möglich, wenn der Zugriff auch erlaubt wird. Der Sicherheitsaspekt bei der Dienstnutzung wird in [Sch02] detailliert betrachtet.

Benutzungsphase

In dieser Phase erfolgt ebenso eine Unterscheidung zwischen anonymen, authentisierten und abonnierenden Dienstkunden, da diese Information auch für die Ausführung des Dienstes wichtig sein kann.

- *Anonymer Dienstnutzer*: Damit ein anonymer Dienstnutzer, der seine Identität dem Dienstanbieter nicht mitteilt, neben kostenlosen auch kostenplichtige Dienste nutzen kann, ist die Einschaltung einer unabhängigen Instanz (z.B. Notar) notwendig, der gegenüber die Authentisierung des Dienstnutzers erfolgt. Die Vergebührung der Dienstnutzung vollzieht sich dann entweder über diese vertrauenswürdige Instanz oder über die Bezahlung mit elektronischem Geld.

- *Authentisierter Dienstnutzer*: Nach der Authentisierung des Dienstnutzers kennt der Dienstanbieter dessen Identität. Diese Art der Dienstnutzung ist vor allem bei kostenpflichtigen Diensten sinnvoll, unabhängig davon ob die Registrierung einmalig oder aber pro Dienstnutzung erfolgt. Eine Registrierung kann aber auch für kostenlose Dienste sinnvoll sein, wenn der Dienstanbieter personalisierte Inhalte liefern möchte.

- *Abonnierender Dienstnutzer*: Nach dem Abonnieren des Dienstes durch den Dienstnutzer werden speziell für ihn entsprechende Ressourcen bereitgestellt. Beispielsweise kann für den Dienstnutzer dadurch eine hohe Verfügbarkeit und Qualität des Dienstes sichergestellt werden. Unabhängig davon, ob es sich dabei um einen kostenpflichtigen oder kostenlosen Dienst handelt, ist ein Abonnement auch bei häufigeren Dienstnutzungen anzuraten. Für die Dauer des Abonnierens sollte eine Gebühr erhoben werden können. Vor Inanspruchnahme des abonnierten Dienstes ist zwar eine Authentisierung aber kein erneuter Abschluss eines Dienstnutzungsvertrages notwendig.

5.2.4 Einheitliches Dienstzugangsprotokoll

Um ein allgemeingültiges Management von Diensten und Dienstnutzungen zu ermöglichen, muss die Nutzung von Anwendungs- und Managementdiensten durch Agenten mittels eines einheitlichen Dienstzugangsprotokolls vollzogen werden. Dieses Protokoll sollte im wesentlichen die in Abschnitt 5.2.3 definierten Nutzungsaspekte berücksichtigen und sämtliche generische Abläufe der nachfolgend beschriebenen Phasen einer Dienstnutzung realisieren.

In Abbildung 5.4 ist beispielhaft ein Protokoll dargestellt, das diesen Anforderungen genügt. Dieses in CASA (Component Architecture for Service Agents) definierte Protokoll wird als „Metaprotokoll“ bezeichnet, da spezifische Protokolle für die Dienstaushandlung und -nutzung eingebettet werden können. Dieses Merkmal ermöglicht die Verwendung beliebiger Aushandlungsverfahren und zusätzlicher Dienstparameter.

Dienstanforderungsphase

Der Kunde fordert den Dienst mit Hilfe eines „request“-Sprechaktes an, der die Dienstinstanz enthält und dem optional schon Parameter für die Dienstnutzung mitgegeben werden können. Daraufhin beginnt der Anbieter mit der Auswertung dieser Anforderung. Überprüft werden der angeforderte Dienstname sowie evtl. mitgelieferte Parameter. Der Dienst kann vom Anbieter auf bestimmte Nutzer (z.B. mit Abonnement) eingeschränkt sein. In dieser Phase werden fest vorgegebene generische Dienstparameter zwischen dem Anbieter und Nutzer ausgehandelt. Hierzu können u.a. die zur Absicherung der Verhandlungsphase notwendigen Schlüssel gehören. Aufgrund dieser Überprüfung lehnt der Anbieter die Dienstanforderung entweder per „refuse“-

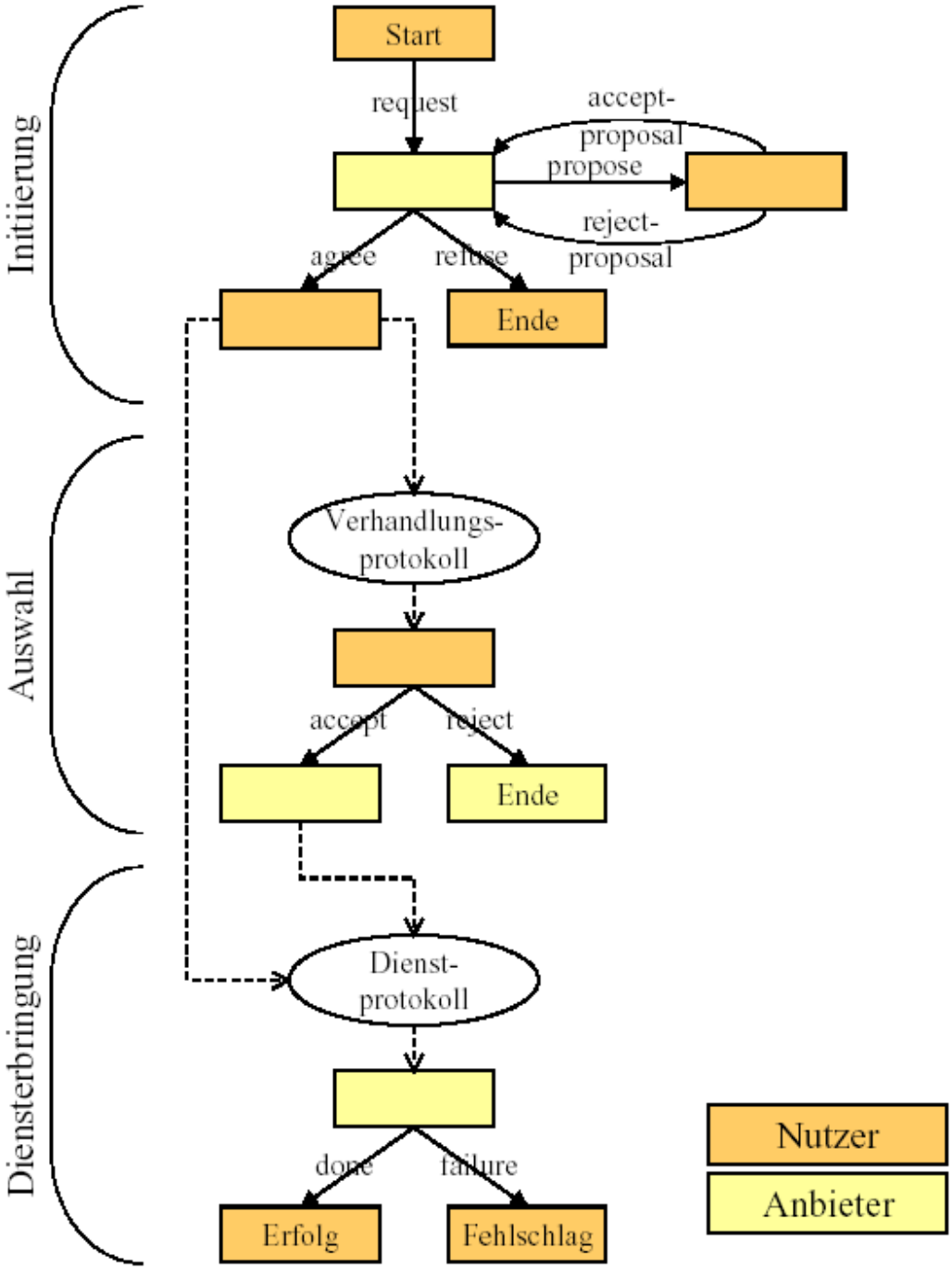

Abbildung 5.4: Dienst-Metaprotokoll in CASA [Ses02]

Sprechakt ab, oder aber er nimmt sie durch Senden eines „agree“-Sprechaktes an und geht direkt in die Verhandlungsphase.

Dienstverhandlungsphase

In der Verhandlungsphase werden die dienstspezifischen Parameter und optional Sicherheits- sowie Accountingparameter ausgehandelt. Die Sicherheitsparameter beziehen sich auf die Absicherung der reinen Dienstnutzungsphase und überlagern somit die während der Anforderungsphase ermittelten Vereinbarungen. Im Zuge der Dienstverhandlung werden die schrittweise ausgehandelten Verträge der in der Anforderungsphase erzeugten Dienstinstanz zugeordnet. Am Ende der Dienstverhandlungsphase teilt der Nutzer durch einen „accept“- bzw. „reject“-Sprechakt mit, ob der Anbieter innerhalb der abschließenden Dienstnutzungsphase den Dienst erbringen soll oder nicht. Ist keine Aushandlung notwendig, so kann diese Phase auch ausgelassen werden.

Vertragsabschluss

Der Vertragsabschluss beendet die Verhandlungsphase und bildet den möglichen Einstieg in die Nutzungsphase. Hier bietet sich die Möglichkeit die im Vertrag ausgehandelten Konditionen durch einen Notar, auf welchen sich beide Parteien vorab geeinigt haben müssen, beglaubigen zu lassen.

Dienstnutzungsphase

In der Dienstnutzungsphase erfolgt die eigentliche Diensterbringung durch den Anbieter. Sollten hierfür andere Sicherheitsparameter als für die Dienstverhandlung zwischen Kunde und Anbieter vereinbart worden sein, oder sind die in der Dienstverhandlungsphase benutzten Schlüssel ungültig geworden, so muss erneut eine entsprechende Schlüsselaushandlung durchgeführt werden. Ebenfalls zu Beginn dieser Phase wird der Abrechnungsprozess für den Dienst gestartet, sofern die notwendigen Angaben im Vertrag definiert sind. Der Anbieter beendet schließlich die Dienstnutzung, indem er durch einen „done“- bzw. „failure“-Sprechakt entweder die erfolgreiche Ausführung inklusive des Ergebnisses oder einen Fehlschlag mitteilt.

5.2.5 Bestandteile eines Dienstnutzungsvertrages

Ein Dienstnutzungsvertrag besteht gewöhnlich aus mehreren Teilen, die beispielsweise folgende Inhalte haben können:

1. Allgemeine Angaben:
   - eindeutige Nummer des Vertrages
   - Datum des Abschlusses
   - Gültigkeitsdauer
2. Vertragspartner:
   - Angaben zum Diensterbringer
   - Angaben zum Dienstkunden
   - eventuell ein Notar
3. Dienstangaben:
   - Name des Dienstes
   - Dauer der Diensterbringung
   - Dienstgüte (QoS)
4. Angaben zur Abrechnung:
   - Gebührenkontrolle durch den Dienstnutzer
   - Tarif (Zeittakt, Pauschale, Preislimit, etc.)
   - Art der Rechnungszustellung (Brief, Email)
   - Detaillierungsgrad der Rechnungsdaten
5. Zahlungsmodalitäten:
   - Bezahlungsform (Vorauskasse, Nachnahme, per Rechnung)
   - Zahlungsmittel (Kreditkarte, SET, elektronische Münzen)

Auch für den Dienstnutzungsvertrag sind die verschiedenen Bekanntheitsgrade der Kunden zu berücksichtigen (siehe Abschnitt 5.2.3). Das zuvor genannte Beispiel enthält Aspekte für die anonyme, registrierte und auch abonnierte Dienstnutzung.

Beim Abschluss von Dienstnutzungsverträgen mit Organisationen als Dienstkunden könnte zusätzlich festgelegt werden, welche oder wie viele Mitglieder mit welchen Rechten oder Einschränkungen die vom Dienstkunden abonnierten Dienste nutzen dürfen. Dabei kann es Sinn machen auch Informationen über die potentiellen Dienstnutzer aufzunehmen.

5.3 Architektur von MIAS

Die Architektur der Managementinfrastruktur MIAS setzt sich aus drei Ebenen zusammen (vgl. [KHA07]). Jedes dieser in Abbildung 5.5 dargestellten Ebenen realisiert für die übergeordneten Ebenen eine Menge von Funktionen, die auf Funktionen der unteren Ebenen aufbauen können.

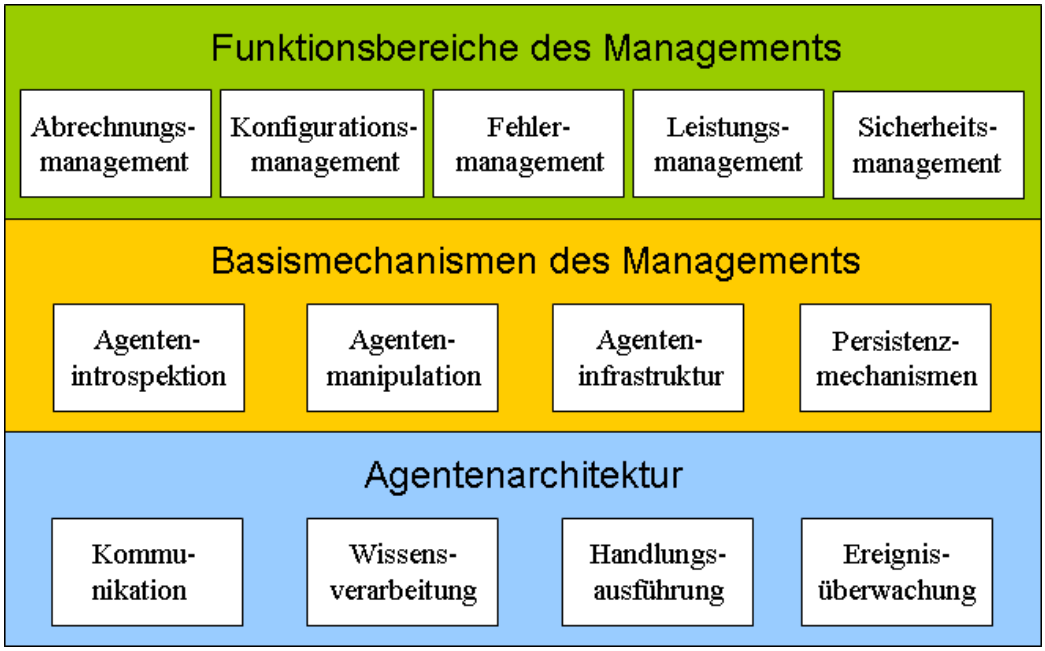

Abbildung 5.5: Das dreistufige Modell von MIAS

Auf der untersten Ebene befindet sich die zu verwaltende Agentenarchitektur mit den in Abschnitt 5.2 beschriebenen Eigenschaften einer Dienstumgebung. Jede der zu berücksichtigenden Agentenarchitekturen sollte auf beliebige Art und Weise eine Kommunikation zwischen Agenten, eine Wissensverarbeitung und Handlungsausführung durch Agenten sowie eine Überwachung von agenteninternen und architekturspezifischen Ereignissen unterstützen.

Aufbauend auf den Fähigkeiten und Eigenschaften der Agentenarchitektur werden auf der zweiten Ebene Basismechanismen der Managementinfrastruktur bereitgestellt (siehe Kapitel 6). Hierzu gehören die Introspektion und Manipulation von einzelnen Agenten und deren Bestandteile, Fähigkeiten und Verhaltensweisen, die Bereitstellung einer FIPA-konformen verteilten Agenteninfrastruktur zur Erzeugung, Migration und Terminierung von Agenten sowie zur Registrierung und Suche von Agenten und Diensten und nicht zuletzt Mechanismen zur persistenten Speicherung von Agentenzuständen. Die Realisierung dieser Basismechanismen hängt sehr stark von den jeweiligen Eigenschaften der zugrundeliegenden Agentenarchitektur ab und ermöglicht demzufolge nur wenige generische Lösungsansätze.

Auf der obersten Ebene werden unter Verwendung der auf der zweiten Ebene bereitgestellten Basismechanismen die im OSI-Management definierten Funktionsbereiche abgedeckt. Hierzu gehören u.a. das in Kapitel 7 und Kapitel 8 beschriebene Abrechnungs- und Konfigurationsmanagement sowie das Fehler-, Leistungs- und Sicherheitsmanagement. Die Realisierung dieser Managementbereiche kann größtenteils unabhängig von der zugrundeliegenden Agentenarchitektur erfolgen und erlaubt vorwiegend generische Lösungen. Für die Implementierung der Managementinfrastruktur kann selbstverständlich auch die eigentlich zu verwaltende Agentenarchitektur verwendet werden.

5.4 Grundlegende Ontologie von MIAS

Um die Vorteile der Agententechnologie auch für das Management nutzbar zu machen und den Zugriff der Agenten auf die Managementfunktionen möglichst einfach zu halten, wird die Managementinfrastruktur teilweise selbst als agentenbasiertes System realisiert. Die Anwendungs- wie auch die Managementdienste werden demzufolge in homogener Form auf den Plattformen angeboten. Aus diesem Grund müssen ähnlich wie bei der Entwicklung agentenbasierter Anwendungen (vgl. [Hes]) auch für die Managementinfrastruktur entsprechende Ontologien entworfen werden. Dabei lässt sich direkt aus den in Abschnitt 5.2 beschriebenen Eigenschaften der zu berücksichtigenden Dienstumgebungen eine zugehörige Ontologie ableiten, die die Basis für die zu entwickelnden Managementfunktionen und weitere Ontologien darstellt. Sie definiert auf formale Weise die Struktur der grundlegenden Managementinformationen und ihre Beziehungen untereinander. Im folgenden

werden die Elemente dieser Basisontologie und ihre Semantik ausführlich vorgestellt.

5.4.1 Managebare Einheiten

Da sich Management auf konkret existierende Objekte bezieht, wurden im Rahmen der Ontologie vorab allgemein administrierbare Einheiten definiert. Aus einer Entität (Entity) leiten sich dann gemäß der Abbildung 5.6 die Agentenplattformen (Market), die Agenten (Agent) sowie die beteiligten Personen bzw. Organisationen (Human) ab. Eine Entität besitzt immer einen eindeutigen Namen (uid), unter dem sie identifizierbar ist, Informationen über den Aussteller der UID (uidIssuer) und eine Identitätsbeschreibung (identifications).

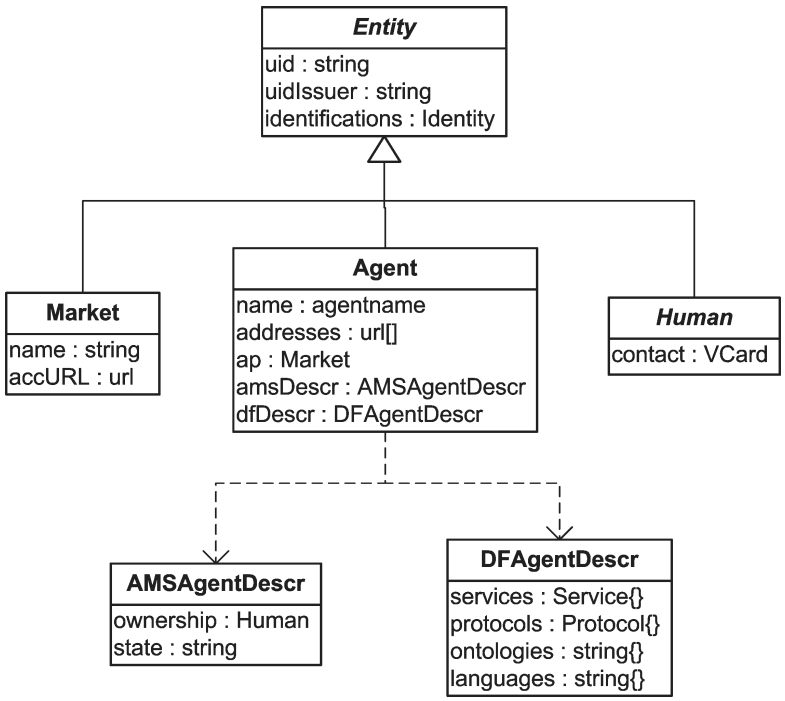

Abbildung 5.6: Beschreibung der managebaren Einheiten

Eine Agentenplattform verfügt über weitere Informationen. Hierzu zählen in Anlehnung an die FIPA-Struktur (vgl. [FIP02c]) der Name zur Charakterisierung der Plattform (name) sowie die Adresse des zugehörigen ACC (accURL).

Ein Agent wird zusätzlich durch einen Namen (name), der für die global eindeutige Identifizierung des Agenten die URL der Heimatplattform (HAP) enthält, durch eine Liste von Kommunikationsadressen (addresses), unter denen der Agent direkt erreichbar ist, und durch die Plattform, auf der sich der Agent momentan befindet (ap) beschrieben. Weitere Informationen über den Agenten werden sowohl durch das AMS als auch durch den DF verwaltet. So kann das AMS (AMSAgentDescr) jedem Agenten jeweils einen menschlichen Besitzer (ownership) und einen der folgenden Lebenszykluszustände (state) zuordnen.

- *active*: Bedeutet, dass der Agent zur Zeit läuft und seine Aufgaben erfüllt.
- *suspended*: Ein solcher Agent ist inaktiv und verbraucht keine Rechenzeit.
- *serialized*: Bedeutet, dass der Agent in eine Datenstruktur überführt worden ist, auf deren Basis er entweder persistent gespeichert oder im Netzwerk übertragen werden kann.
- *transit*: Zustand während der Übertragung auf eine andere Plattform.
- *defective*: Zustand für den Fehlerfall.
- *stepping*: Erlaubt die schrittweise Abarbeitung von primitiven Handlungen.

Der DF (DFAgentDescr) kennt zu jedem Agenten die angebotenen Dienste (services), die beherrschten Protokolle (protocols), die bekannten Ontologien (ontologies) sowie die verstandenen Wissensrepräsentationssprachen (languages).

5.4.2 Dienstbeschreibung

Die Beschreibung der Dienste (Service) besteht aus den beiden Anteilen für die Seite der Diensterbringung (ServiceProvision) sowie der Dienstnutzung (ServiceUsage). Hierdurch werden die Informationsanforderungen seitens des Dienstanbieters und die des Dienstkunden erfüllt (siehe Abbildung 5.7).

Eine Dienstbeschreibung umfasst den Namen (name), eine textuelle Beschreibung (description) und den Typ (type) des Dienstes, eine Liste von Schlüsselworten (keywords), wodurch die Funktionalität des Dienstes metasprachlich

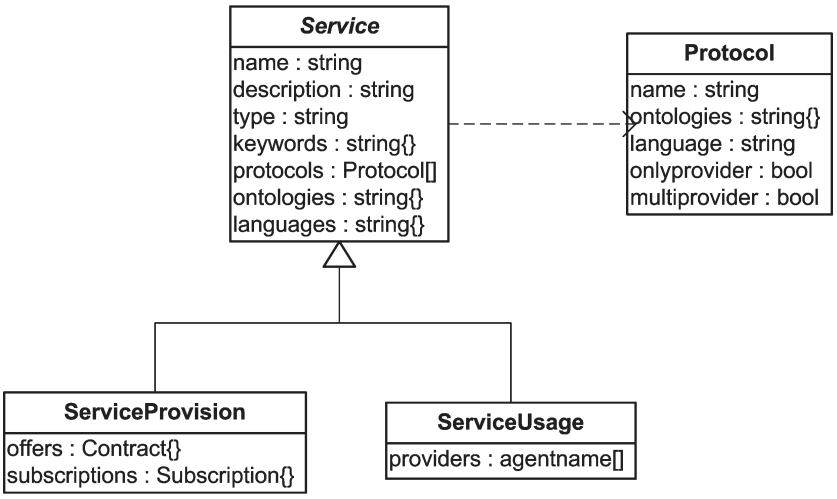

Abbildung 5.7: Beschreibung von Diensten

umrissen wird, die berücksichtigten Wissensrepräsentationssprachen (languages), die verwendeten Ontologien (ontologies) und die definierten Protokolle (protocols).

Der Anbieter hält für jeden der von ihm angebotenen Dienste Vertragsangebote (offers) bereit, die für eine Dienstnutzung in Frage kommen. Desweiteren sind die für einen Dienst registrierten Abonnenten (subscriptions) gespeichert. Aus Sicht des Dienstnutzers ist zusätzlich die Liste der Agenten (providers) interessant, die den Dienst anbieten. Diese Information ist für eine Dienstrecherche über den verteilten Verzeichnisdienst verfügbar.

Die dynamische Seite der Dienstnutzung ist in Abbildung 5.8 dargestellt. Für jede Dienstnutzung werden vom Kunden wie auch vom Anbieter eine entsprechende Beschreibung (ServiceInst) vorgehalten. Zu Anfang der Dienstnutzung, d.h. vor der Verhandlungsphase, ist die Liste der Verträge (contracts) für diese Dienstnutzung noch leer. Erst nach Abschluss der Verhandlungen sollte diese Liste die gültigen Dienstvereinbarungen enthalten. Desweiteren sind in der Dienstinstanz eine eindeutige ID der Konversation (conversation), der genutzte Dienst (service) und das verwendete Dienstnutzungsprotokoll (protocol), der Name des anbietenden (provider) und des nutzenden (user) Agenten sowie die Startzeit (startTime) des Dienstes angegeben. Der Zeitpunkt des Dienstendes (endTime) wird selbstverständlich erst nach Ablauf der Dienstnutzung eingetragen.

Unterschieden werden dabei eine einfache und eine abonnierte Dienstnutzung sowie das Abonnieren selbst, das ebenfalls eine Dienstnutzung darstellt.

Bei einer einfachen Dienstnutzung (UnsubscribedUsage) ist die Identität des Dienstnutzers (human) eingetragen, sofern es sich nicht um eine anonyme Nutzung handelt. Bei der abonnierten Nutzung (SubscribedUsage) ist der Dienstnutzer durch die Referenz auf das Abonnement als solcher identifizierbar. Zu jedem Abonnement (Subscription) ist die Identität des Kunden (human) angegeben.

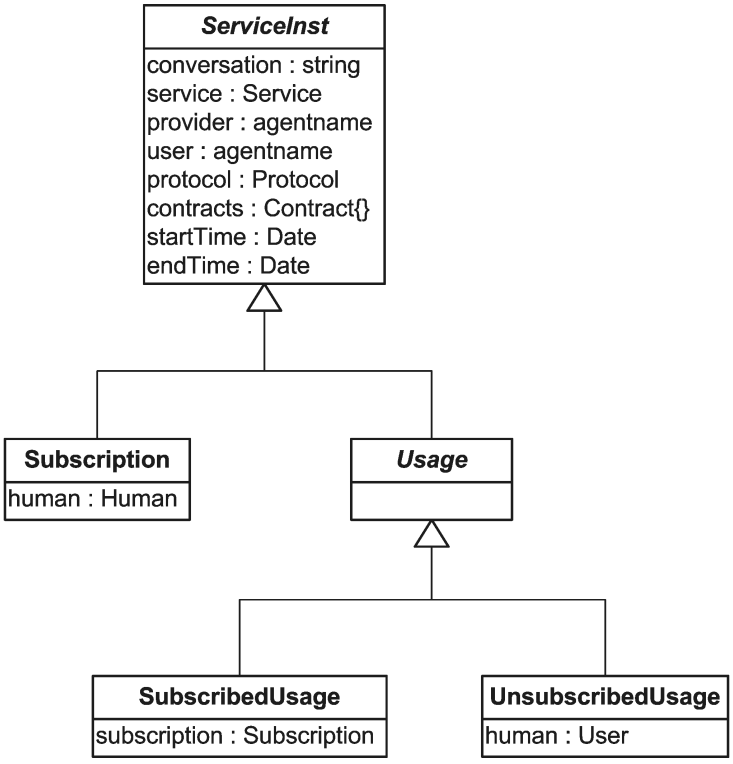

Abbildung 5.8: Beschreibung von Dienstnutzungen

Allen Dienstnutzungen können unter anderem dienst-, abrechnungs- und sicherheitsbezogene Verträge zugeordnet werden (siehe Abbildung 5.9). Sie stellen eine Verfeinerung allgemeiner Verträge (Contract) dar und bieten ein generisches Grundgerüst, welches sich zu konkreten Verträgen verfeinern lässt. Diese Aufgabe obliegt dem Dienstentwickler.

Ein Vertrag enthält eine eindeutige ID (id), das Datum des Vertragsabschlusses (conclusion), den Vertragsgegenstand (service), den Dienstanbieter (provider) und den Dienstkunden (customer), sowie optional den Namen der vertrauenswürdigen Instanz (notary) und jeweils das Datum für den Beginn und das Ende des Vertrages. Damit wird ausgedrückt, dass der Dienstnutzungsvertrag lange vor der eigentlichen Dienstinanspruchnahme geschlossen

werden kann. Für einen dienstbezogenen Vertrag (ServiceContract) können zusätzlich noch die voraussichtliche Dauer der Diensterbringung (duration) und die gewünschte Qualitätsstufe (quality) angegeben sein.

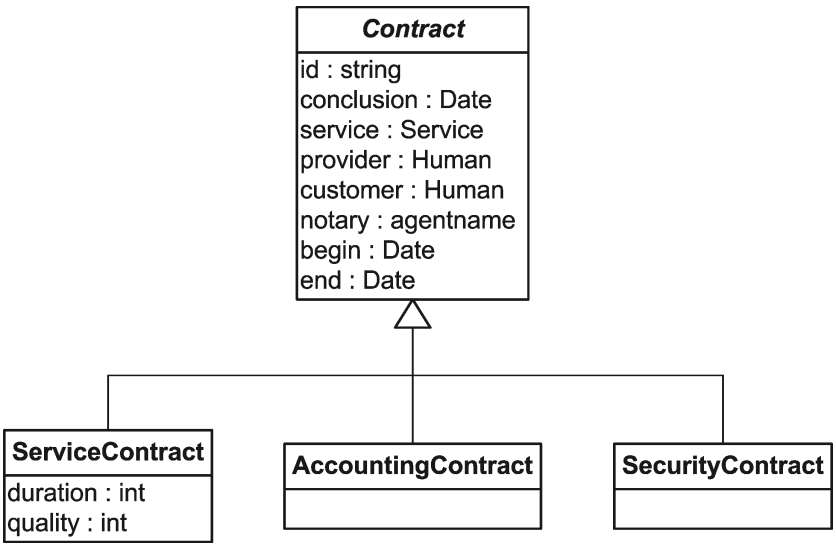

Abbildung 5.9: Beschreibung von Verträgen

Da der Bereich der Sicherheit in MIAS vorerst keine Berücksichtigung findet, sei für die Definition von Verträgen für die Sicherheitsbelange (SecurityContract) auf [Sch02] verwiesen. Die Bestandteile eines Vertrages zur Vergebührung (AccountingContract) werden in Abschnitt 7.5 erwähnt.

5.5 Zusammenfassung

In diesem Kapitel wurde das Grobkonzept von MIAS vorgestellt. Ausgehend von den Eigenschaften agentenbasierter Systeme und den Anforderungen an deren Management erfolgte die Definition von Anwendungsfällen, die die allgemeinen Aufgaben der Managementinfrastruktur widerspiegeln. Es hat sich herausgestellt, dass sämtliche Managementfunktionen wie beispielsweise die Behandlung von Fehlern oder Engpässen egal ob sie automatisiert oder durch einen Administrator gesteuert ablaufen auf elementaren Aktionen zur Überwachung und Manipulation der Agenten und Agentenplattformen aufbauen. Zusätzlich wurde berücksichtigt, dass Agenten auch Dienstleistungen erbringen und die Kooperation in typischen Agentensystemen über Dienstnutzun-

gen abgewickelt wird. Somit spielt auch die Verwaltung von Diensten und die Unterstützung der Agenten bei der Dienstnutzung eine Rolle.

Um eine allgemeingültige Verwaltung von Diensten zu ermöglichen, wurden die wichtigsten Eigenschaften der Dienste aus Sicht von MIAS zusammengefasst. Unter anderem wurden die Beziehung zwischen Diensten und Agenten, die Phasen einer Dienstbereitstellung und Dienstnutzung sowie beispielhaft ein Dienstzugangsprotokoll beschrieben.

Unter Berücksichtigung der Aufgaben einer Managementinfrastruktur und den Eigenschaften typischer agentenbasierter Systeme erfolgte die Definition eines dreistufigen Modells für MIAS. Dieses Modell erlaubt eine weitgehende Unabhängigkeit zwischen der konkret unterstützten Agentenarchitektur und den höherwertigen, weitestgehend generischen Managementfunktionen. Das bedeutet, dass MIAS prinzipiell für beliebige Agentensysteme eingesetzt werden kann, ohne diese neu implementieren oder ändern zu müssen. Die mittlere Ebene übernimmt durch Bereitstellung von Basismechanismen wie Introspektion und Manipulation die Abbildung der generischen Managementfunktionalitäten auf elementare Aktionen der verwendeten Agentenarchitektur.

Abschließend wurde in diesem Kapitel eine Ontologie beschrieben, die das Informationsmodell von verwaltbaren Einheiten (z.B. Agenten oder Benutzer), Diensten, Dienstnutzungen und Verträgen formal definiert. Die Agentenbeschreibung orientiert sich an FIPA, die Dienstbeschreibung kann sowohl Informationen für den Anbieter als auch Nutzer enthalten und bei den Dienstnutzungen wird auch ein Subscription berücksichtigt. Diese Managementinformationen bilden die Basis für alle zu entwickelnden Managementfunktionen und weiterer Ontologien.

Kapitel 6
Basismechanismen von MIAS

Unter den Basismechanismen von MIAS werden alle Funktionen der zweiten Ebene dieser Managementinfrastruktur verstanden, die in erster Linie auf den Fähigkeiten der zugrundeliegenden Agentenarchitektur aufbauen und eine einheitliche Schnittstelle für höherwertige Managementfunktionen anbieten, um die in Abschnitt 5.1 beschriebenen Aufgaben einer Managementinfrastruktur lösen zu können. Es wird unter anderem gezeigt, inwiefern diese Basismechanismen für die Lösung von FCAPS-spezifischen Aufgaben der obersten Ebene von MIAS nützlich sind.

In diesem Kapitel steht vor allem das Management einzelner Agenten im Vordergrund, da sie die Grundbausteine für die Realisierung von Plattformen und Anwendungen darstellen (siehe Abschnitt 5.2.1). Je nach Art der bereitgestellten Managementschnittstelle handelt es sich dabei um agenteninterne oder agentenexterne Managementfunktionen. Zu den agenteninternen Managementfunktionen gehören ein Zeitgeber, die Beobachtung und Introspektion der Bestandteile und Aktionen von Agenten sowie die Manipulation der Agenten. Managementfunktionen, die als Dienste auch von anderen Agenten genutzt werden können, sind die Agenteninfrastruktur zur Verwaltung kompletter Plattformen, Agenten und Dienste, Mechanismen zur Persistenz von Agenten und die Überwachung von mobilen Agenten.

Am Ende dieses Kapitels werden Werkzeuge vorgestellt, mit denen diese Managementaktivitäten über geeignete Benutzerschnittstellen auch von Menschen durchgeführt werden können.

6.1 Zeitgeber

Der Zeitgeber ist optionaler Bestandteil eines Agenten und stellt eine innere Uhr mit Weckfunktion dar, um zeitgesteuerte Aktionen ausführen zu können. Der Agent kann dabei über das Erreichen beliebig vieler Zeitpunkte oder in Form eines Kurzzeitweckers über den Ablauf von Zeitspannen informiert werden. Die Überwachung einer Zeitspanne kann einmalig oder periodisch erfolgen.

Eine Besonderheit des Zeitgebers von MIAS ist die Berücksichtigung des Schrittmodus von Agenten bei der Überwachung von Zeitspannen. Führt der Agent beispielsweise zum Zwecke des Testens und der Fehlersuche seine Aktionen nur schrittweise aus, so verlängern sich automatisch alle überwachten Zeitspannen um die aufgetretenen Stillstandszeiten. Diese zwischenzeitlichen Unterbrechungen haben also keinen Einfluss auf die zeitliche Abfolge von Aktionen und somit auf die Arbeitsweise des Agenten. Die Berücksichtigung von überwachten Zeitpunkten macht hierbei keinen Sinn.

6.2 Agentenintrospektion

Unter dem Begriff Introspektion wird die Fähigkeit eines Agenten verstanden, seine Bestandteile und Aktionen zu beobachten, um relevante Informationen über sich selbst zu gewinnen. Diese Informationen können sich sowohl in zeitlicher als auch in inhaltlicher Hinsicht stark unterscheiden.

Zeitlich gesehen ist die Aktualität der Informationen von unterschiedlicher Dauer. So ändern sich beispielsweise die Aktionen eines Agenten gewöhnlich wesentlich häufiger als die Fähigkeiten des Agenten. Zur Reduzierung der Menge an erzeugten Daten kann es bei häufigen Änderungen sinnvoller sein die vollständigen Zustandsinformationen nur einmal am Anfang der Überwachung bereitzustellen und anschließend jeweils nur die Änderungsinformationen mitzuteilen.

Die Struktur der gewonnenen Informationen hängt einerseits von der erzeugenden Agentenkomponente ab, andererseits können sie nach der Art der Managementaufgabe klassifiziert werden. In MIAS wird nach den OSI-Managementbereichen unterschieden, so dass die Komponenten eines Agenten unterschiedliche Informationen zur Erkennung von Fehlern und Engpässen

sowie über die Konfiguration des Agenten und für die Abrechnung von Diensten liefern. Hierbei sollte zudem eine Detaillierung angegeben werden, mit der sich die Menge an Managementinformationen skalieren lässt.

Ein wesentlicher Aspekt bei der Introspektion betrifft die Art der Informationsanforderung und -bereitstellung. Eine Möglichkeit besteht im „Polling“, bei dem bestimmte Informationen zum Agenten bei Bedarf oder in definierten Zeitabständen abgefragt werden. Dieser Ansatz eignet sich besonders dann, wenn die Informationen nicht zeitnah und im Vergleich zu den Änderungen seltener benötigt werden. Ein anderes Verfahren stellt das „Logging“ dar, bei dem die beobachteten Bestandteile eines Agenten selbständig die vordefinierten Änderungen erfassen, protokollieren und diese Informationen entweder sofort oder in vorgegebenen Zeitabständen an alle angemeldeten Interessenten innerhalb des Agenten weiterreichen. Für die zeitnahe Übermittlung sich häufig ändernder Daten eignet sich dieses Verfahren deutlich besser. In den folgenden beiden Abschnitten wird nun etwas genauer auf das Logging und das Polling von Agenteninformationen eingegangen.

6.2.1 Logging von Agentenereignissen

Für die Überwachung von Agentenereignissen müssen entsprechende Datenstrukturen existieren, die auch zur gerichteten Weiterleitung relevanter Informationen dienen. Dadurch können strukturierte und detaillierte Änderungsinformationen und nicht nur einfache Meldungen, wie sie beispielsweise Log4J[1] für eine mögliche Fehlersuche definiert, bereitgestellt werden. Bevor jedoch die Daten weitergeleitet werden können, müssen sie erzeugt werden. Die Generierung der Daten erfolgt unabhängig davon, welche Managementbereiche[2] diese Daten verwenden. Um das Logging an verschiedene Anforderungen anpassen zu können, muss es so generisch sein, dass prinzipiell jede beliebige Aktivität des Agenten und seiner Bestandteile überwacht werden kann. Typische Beispiele von Agentenereignissen, die für das Logging von Agenten interessant sein können, sind:

[1] Log4J (http://logging.apache.org/log4j/) ist ein Projekt der Apache Software Foundation, das Softwareentwickler ermöglicht textuelle Meldungen geordnet nach Typ und Wichtigkeit zu protokollieren.

[2] Logdaten stehen allen Managementfunktionen, sowohl den Basisfunktionalitäten als auch den Funktionen der FCAPS-Managementbereiche, zur Verfügung. Sie müssen somit nur einmal generiert und gespeichert werden.

- Kontaktaufnahme zu (oder von) einem anderen Agenten
- Erbringung eines Dienstes für andere Agenten
- internen Aktionen, die viel Leistung kosten oder abgerechnet werden sollen
- Migration des Agenten
- spezielle Ereignisse (z.B. Fehlersituationen)

Damit nur relevante Änderungen gesammelt und übertragen werden, müssen sich Interessenten unter Angabe eines Ereignismusters und eines Übertragungsintervalls oder -zeitpunktes bei entsprechenden Vermittlern registrieren und deregistrieren lassen. Die Vermittler sammeln sämtliche Ereignisse der angegebenen Typen von den zugehörigen Erzeugern ein, vergleichen jedes Ereignis mit den vorliegenden Mustern und leiten die Information an alle Interessenten weiter sofern der Zeitgeber das Erreichen des Zeitpunktes für das Informieren gemeldet hat. Zeitspannen werden im Schrittmodus des Agenten um die entsprechenden Stillstandszeiten verlängert (siehe Abschnitt 6.1). Wurde bei der Registrierung keine Zeit angegeben, so erfolgt das Weiterleiten der Ereignisse sofort. Bei einmaliger Überwachung entfällt das explizite Deregistrieren beim Vermittler. Für die Definition von Ereignissen und Ereignismustern wurde die in Abbildung 6.1 und Abbildung 6.2 dargestellte Ontologie entworfen, die um neue agenten- und dienstspezifische Elemente erweitert und verfeinert werden kann.

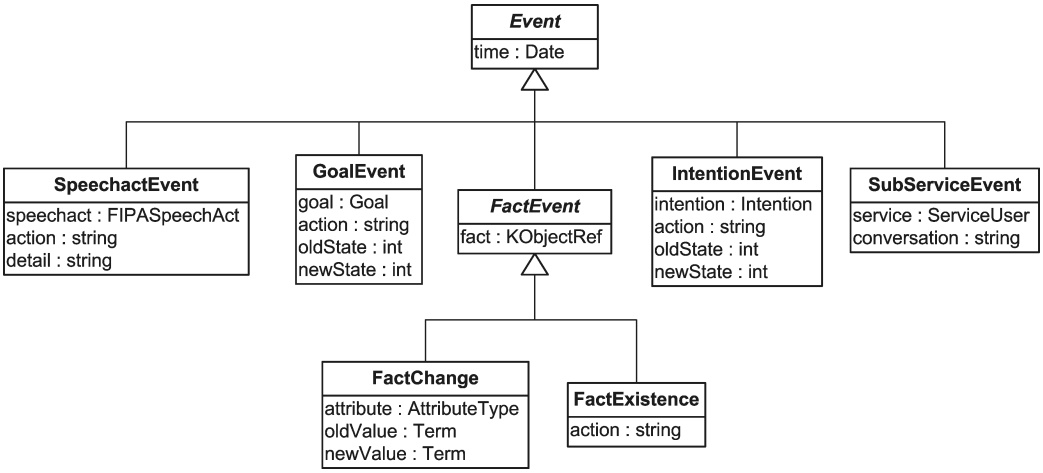

Abbildung 6.1: Ontologie zur Definition von Ereignissen

Zu jedem Ereignis (Event) ist der Zeitpunkt (time) angegeben, zu dem das Ereignis eingetreten ist. Spezielle vordefinierte Ereignistypen berücksichtigen bereits teilweise die Besonderheiten der JIAC-Architektur. Hierzu gehören der Austausch von Sprechakten (SpeechactEvent), das Starten einer Dienstnutzung (SubServiceEvent) sowie Änderungen am Zielstapel (GoalEvent), an der Faktenbasis (FactEvent) und an der Intentionenstruktur (IntentionEvent). Bei Änderungen an der Faktenbasis wird zwischen dem Hinzufügen oder Entfernen eines Faktes (FactExistence) und dem Ändern eines Attributwertes eines Faktes (FactChange) unterschieden. Die meisten dieser Ereignistypen enthalten das geänderte Element, die ausgeführte Aktion (z.B. „added“, „changed“ oder „removed“) sowie den alten und den neuen Zustand des Elementes. Zum Austausch eines Sprechaktes ist der Typ des Sprechaktes (detail) und als Aktion das Senden oder Empfangen angegeben. Beim Starten einer Dienstnutzung sind die ID (conversation) und detaillierte Informationen wie der Verursacher (service) von Interesse.

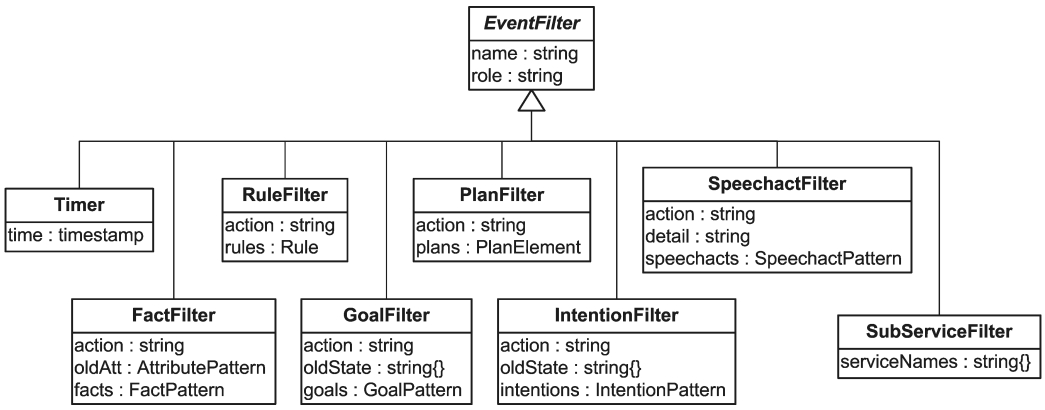

Abbildung 6.2: Ontologie zur Definition von Ereignismustern

Ereignismuster (EventFilter) enthalten zum Zwecke der späteren Zuordnung eine ID (name) und den Namen des Vermittlers (role), der für dieses Ereignismuster zuständig ist. Zu jedem Ereignistyp sollte es eine Definition eines entsprechenden Ereignismusters geben, mit dem die Überwachung eingeschränkt werden kann. So existiert jeweils ein Muster für die Überwachung des Sprechaktverkehrs (SpeechactFilter), von Dienstnutzungsstarts (SubServiceFilter), der Zielverfolgung (GoalFilter), des Faktenwissens (FactFilter) und der Handlungsausführung (IntentionFilter). Zusätzliche Ereignismuster erlauben die vom Zeitgeber (Timer), von der Planbibliothek (PlanFilter) und von der Regelbasis (RuleFilter) erzeugten Ereignisse zu überwachen. Wird ein optionales Attribut eines Ereignismusters nicht gesetzt, so sollten die

zugehörigen Ereignisse unabhängig von der Ausprägungen dieses Attributes berücksichtigt werden.

Am Beispiel der JIAC-Agentenarchitektur wurde ein Ereignisvermittler als eine abstrakte Agentenkomponente (MediationBean) realisiert, die Anfragen in Form einer Nachricht (MonitorKnowledgeMessage) von anderen Komponenten des Agenten entgegennimmt. Über eine Antwortnachricht (ResultMessage) wird mitgeteilt, ob die Anfrage über die Registrierung oder Deregistrierung akzeptiert worden ist. Eine Registrierung erfolgt nicht, falls die Komponente aufgrund einer identischen Nachricht bereits registriert ist oder das angegebene Ereignismuster nicht unterstützt wird. Eine Deregistrierung ist nur möglich, wenn eine Registrierung zu der angegebenen Nachricht vorliegt. Sämtliche Registrierungen werden zusammen mit den bereits gesammelten Ereignissen intern in einer Datenstruktur abgelegt, die in die einzelnen Ereignistypen unterteilt ist. Sobald eine Registrierung zu einem dieser Ereignistypen existiert, wird bei der zugehörigen Messkomponente die Überwachung eingeschaltet. Informiert diese Komponente über ein Ereignis, so kann eine speziell bereitgestellte Methode aufgerufen werden, die das oben beschriebene Verhalten aufweist. Dabei wird das Ereignis mit den Ereignismustern aller Registrierungen dieses Typs verglichen und gegebenenfalls der internen Datenstruktur hinzugefügt. Wurde bei einem dieser Registrierungen keine Mitteilungszeit angegeben, erfolgt sofort die Benachrichtigung (InformKnowledgeMessage) der registrierten Komponente. Anderenfalls wird erst auf das Eintreffen der Nachricht vom Zeitgeber gewartet. Wurde keine permanente Wiederholung der Überwachung gewünscht, so wird die Registrierung nach der Mitteilung automatisch entfernt.

Jede spezialisierte Vermittlerkomponente muss zwei abstrakte Methoden implementieren. Eine Methode aktiviert und deaktiviert die Überwachung von Ereignissen zu den zu berücksichtigenden Typen in den entsprechenden Messkomponenten. Die andere Methode entscheidet darüber, ob ein gemessenes Ereignis zu dem angegebenen Ereignismuster gehört oder nicht. Beispielhaft wurden bereits zwei Vermittlerkomponenten realisiert, die sämtliche Ereignisse von den Messkomponenten über Nachrichten zugestellt bekommen. Eine Komponente (SubServiceMediationBean) sendet Nachrichten (MonitorServiceUsageMessage) an die Kommunikationskomponente (CommunicationBean) um über das Starten neuer Dienstnutzungen informiert zu werden. Die andere Komponente (ControlBean) überwacht alle Ereignisse, die mit den elementaren Aktionen eines JIAC-Agenten zu tun haben. Der Kontext dieser Ereignisse und die zugehörigen Nachrichtentypen sowie die Rollen der beteiligten Messkomponenten sind in Tabelle 6.1 genannt.

Kontext	Nachricht	Komponentenrolle
Komponente	FindBeanMessage	KernelRole
Fakt	FindFactMessage	FactBaseRole
Ziel	FindGoalMessage	GoalStackRole PlanSelectionRole
Intention	FindIntentionMessage	ExecutionRole CommunicationRole IntentionStructureRole
Planelement	FindPlanMessage	PlanLibraryRole ServiceLibraryRole
Regel	FindRuleMessage	RuleBaseRole
Sprechakt	MonitorSpeechActMessage	CommunicationRole

Tabelle 6.1: Überwachung eines JIAC-Agenten über Nachrichten

Folgende zusätzliche Ereignisse können über einen direkten Methodenaufruf bei den entsprechenden Komponenten überwacht werden:

- Loggingmeldung eines Threads (z.B. Leistungsdaten)
- Nachrichtenaustausch zwischen zwei Komponenten
- Änderung des Lebenszykluszustandes eines Agenten oder einer Komponente

Innerhalb der Managementinfrastruktur spielt die Loggingfunktion eine Basisrolle, da sie detaillierte Informationen über einzelne Agenten liefert, auf deren Grundlage Management betrieben wird. Die übrigen Managementfunktionen werten die jeweils für sie relevanten Daten aus, verarbeiten sie und reagieren entsprechend ihrer Aufgabe.

6.2.2 Polling von Agentendaten

Während die Loggingfunktion selbständig Daten über Veränderungen innerhalb eines Agenten liefert, muss es auch die Möglichkeit geben, Informationen über den aktuellen Zustand des Agenten zu jedem beliebigen Zeitpunkt abzufragen. Der Agent und die Bestandteile eines Agenten müssen also auf Anfrage Auskunft über sich selbst geben können. Dabei sind folgende Angaben von Interesse:

- Allgemeine Angaben:
  - Agententyp
  - Herkunft
  - Alter
  - Besitzer bzw. Auftraggeber
  - Aufenthaltsort
  - Kommunikationsadressen
- Zustand des Agenten:
  - aktuelle Bestandteile des Agenten
  - Wissen und Fähigkeiten des Agenten
  - Betriebsmodus des Agenten
  - aktuelle Kommunikations- und Kooperationsbeziehungen
  - eventuell der als nächstes auszuführende Plan
  - aktive und wartende Arbeitsschritte

Wie schon zuvor begründet muss der Mechanismus für die Datenabfragen eng mit der jeweiligen Agentenarchitektur und den Sicherheitsmechanismen abgestimmt sein. Die JIAC-Architektur ermöglicht zum Beispiel bereits beliebigen Komponenten über den Nachrichtenmechanismus Daten zu verschiedenen Elementen abzufragen. Tabelle 6.2 zeigt für die verschiedenen Bestandteile eines JIAC-Agenten die zur Verfügung stehenden Nachrichtentypen und die Rollen der beteiligten Komponenten.

Jeder JIAC-Agent enthält in seiner Faktenbasis ein Objekt mit dem Namen „ThisAgent“, das allgemeine Angaben über den Agenten zur Verfügung stellt. Durch den in Tabelle 6.2 genannten Zugriff auf die Faktenbasis können alle Komponenten des Agenten auf diese Informationen zurückgreifen. Die Aktualisierung dieser Daten insbesondere des Aufenthaltsortes und der Kommunikationsadressen des Agenten erfolgt nach jeder Migration und bei jeder Initialisierung einer Transportkomponente.

Folgende zusätzliche Daten eines JIAC-Agenten stehen als Properties zur Verfügung und können über einen direkten Methodenaufruf bei den entsprechenden Komponenten abgefragt werden:

Elemente	Nachricht	Komponentenrolle
Komponenten	FindBeanMessage	KernelRole
Fakten	FindFactMessage	FactBaseRole
Ziele	FindGoalMessage	GoalStackRole PlanSelectionRole
Intentionen	FindIntentionMessage	ExecutionRole CommunicationRole IntentionStructureRole
Planelemente	FindPlanMessage	PlanLibraryRole ServiceLibraryRole
Regeln	FindRuleMessage	RuleBaseRole

Tabelle 6.2: Abfrage von Daten eines JIAC-Agenten über Nachrichten

- Loggingtypen, Loggingstufe oder Leistungsstufe eines Agenten oder einer Komponente
- Größe des Nachrichtenspeichers einer Komponente
- Aktuelle Anzahl unbearbeiteter Nachrichten einer Komponente
- Existenz eines eigenen Threads innerhalb einer Komponente
- Priorität einer Komponente mit eigenem Thread
- Lebenszykluszustand eines Agenten oder einer Komponente
- Aktualisierung der Anbieterliste vor jeder Dienstnutzung
- Maximale Wartezeit für den Empfang eines Sprechaktes
- Timeout für Server-Sockets bei HTTP- oder TCPIP-Kommunikation
- Maximale Anzahl aktiver Handlungen, Ziele und Intentionen
- Verhalten bei der Planelementauswahl (partial matching, backtracking)

Die Agentenintrospektion auf der Basis der Datenabfragen liefert vor allem Statusinformationen über den Agenten und seine Bestandteile. Diese können von allen Managementfunktionen innerhalb des Agenten verwendet werden.

6.3 Agentenmanipulation

Unter dem Begriff der Agentenmanipulation wird die Fähigkeit eines Agenten verstanden, auf seine Bestandteile und sein Verhalten direkten Einfluss nehmen zu können. Es bildet die Grundlage für verschiedene Aufgaben in unterschiedlichen Bereichen des Managements. So sind Eingriffe in den Agenten Voraussetzung für das Konfigurationsmanagement, können aber auch für das Fehlermanagement (z.B. Austausch einer fehlerhaften Komponente) oder das Abrechnungsmanagement (z.B. Hinzufügen einer Messkomponente) wichtig sein. Gewöhnlich stellt die Introspektion eine Voraussetzung für die Manipulation dar, da sich die kontrollierende Instanz meist über den Vollzug und die Auswirkung des Eingriffs informieren möchte. Gegenstand von Manipulationen innerhalb eines Agenten sind vor allem seine sogenannten „aktiven“ Bestandteile. Dabei ist zu klären inwiefern solche Manipulationen bereits laufende Handlungen beeinflussen oder gar zerstörerisch auf den Agenten wirken. So könnte zum Beispiel die Modifikation eines Skriptes oder eines Planes während dessen Ausführung unakzeptable Auswirkungen haben. Im Vergleich dazu ist das Herunterladen und Hinzufügen neuer Skripte oder Pläne relativ unkritisch. Komplexe Manipulationen müssen gegebenenfalls zeitlich organisiert ablaufen, um inkonsistente Zustände zu vermeiden. Die entsprechenden Bestandteile der Agentenarchitektur sollten diese Besonderheiten berücksichtigen. Zu den typischen Manipulationen an einem Agenten gehören die:

- Änderung des Wissens und der Fähigkeiten des Agenten
- Änderung der Verhaltensweisen des Agenten
- Änderung des Betriebsmodus des Agenten
- Ausführung einer Handlung durch den Agenten
- Migration des Agenten

Ähnlich wie bei der Agentenintrospektion hängt die Realisierung der Manipulationsfunktion sehr stark von der Architektur und den bestehenden Sicherheitsmechanismen ab. Die JIAC-Architektur bietet über den Nachrichtenmechanismus bereits beliebigen Komponenten Aktionen zum Hinzufügen, Entfernen, Aktivieren und Suspendieren von verschiedenen Wissenselementen eines Agenten an. Tabelle 6.3 zeigt die jeweils zur Verfügung stehenden Nachrichtentypen und die Rollen der beteiligten Komponenten.

Wissenselement	Nachricht	Komponentenrolle
Fakt	ChangeFactMessage	FactMaintenanceRole
Ziel	ChangeGoalMessage	GoalSelectionRole
Intention	ChangeIntentionMessage	SchedulerRole
Planelement	ChangePlanMessage	PlanLibraryRole ServiceLibraryRole
Regel	ChangeRuleMessage	RuleBaseRole SituationAssessmentRole

Tabelle 6.3: Manipulation eines JIAC-Agenten über Nachrichten

Folgende zusätzliche Aktionen innerhalb eines JIAC-Agenten können über einen direkten Methodenaufruf (z.B. Setzen einer Property) bei den entsprechenden Komponenten ausgeführt werden:

- Änderung des Lebenszykluszustandes sowie der Loggingtypen, Loggingstufe oder Leistungsstufe eines Agenten oder einer Komponente
- Hinzufügen, Entfernen oder Austausch einer Komponente
- Änderung der Eigenschaften einer Komponente (Größe des Nachrichtenspeichers, eigener Thread, Priorität)
- Ausführung eines Agentenschrittes innerhalb des Schrittmodus
- Aktualisierung der Anbieterliste vor jeder Dienstnutzung
- Änderung der Timeouts für die Kommunikation (Empfang eines Sprechaktes, Aufbau von Server-Sockets)
- Änderung der maximalen Anzahl aktiver Handlungen, Ziele oder Intentionen
- Änderung des Verhaltens bei der Planelementauswahl (partial matching, backtracking)

Die Erhöhung der Leistungsstufe wird durch eine Verringerung der Anzahl an Konsistenzprüfungen erreicht. Eine Agentenkomponente kann nur durch eine Komponente mit der gleichen Rolle ausgetauscht werden. Die Werte der rollenspezifischen Properties werden dabei von der neuen Komponente übernommen. Es ist sinnvoll die Abarbeitung eingehender Nachrichten einer Komponente in einem eigenen Thread durchzuführen, wenn sie durch andere Komponenten nicht blockiert werden soll. Damit keine Nachrichten

verloren gehen, ist der Nachrichtenspeicher der Komponenten nie kleiner als die Anzahl noch nicht verarbeiteter Nachrichten. Demzufolge wird der Nachrichtenspeicher bei Bedarf automatisch vergrößert und kann minimal auf die Anzahl der zu speichernden Nachrichten verkleinert werden. Die maximale Anzahl aktiver Ziele, Intentionen und Handlungen kann hingegen auf einen kleineren Wert gesetzt werden. In diesem Fall erfolgt erst einmal keine neue Aktivierung.

6.4 Agenteninfrastruktur

Nachdem in den vorhergehenden Abschnitten die agenteninternen Managementfunktionen behandelt worden sind, werden in den nächsten Abschnitten die agentenübergreifenden Basismechanismen beschrieben. Hierzu gehören unter anderem die von der FIPA definierten Managementdienste (siehe Abschnitt 3.2.1). Sie sind wesentliche Grundfunktionen des Managements in einer heterogenen und dynamischen Umgebung mit autonomen und zum Teil mobilen Agenten und gewährleisten die Zusammenarbeit zwischen allen beteiligten Agenten. Damit sind Agenten jederzeit in der Lage ihre Kooperationspartner ausfindig zu machen und die Besitzer von Agenten wissen wo sich ihre Agenten gerade befinden. Es könnte fatale Folgen für die Funktionalität des gesamten Agentensystems haben, wenn ein wandernder oder stationärer Agent irgendwann für seine Partner oder Besitzer verloren gehen würde. Im folgenden wird die Realisierung dieser Grundfunktionen beschrieben.

Jeder Agent besitzt eine eindeutige Kennung, die sich aus dem Agentennamen, aus der Kennung der Heimatplattform[3] und gegebenenfalls aus dem Erzeugungsdatum zusammensetzt (siehe Abschnitt 5.4.1). Das Erzeugungsdatum gewährleistet auch dann die Eindeutigkeit von Kennungen wenn Agenten mehrfach nacheinander mit dem gleichen Namen erzeugt und wieder beendet werden sollen. Die Kennung einer Plattform entspricht einer Socket-basierten Kommunikationsadresse des jeweiligen Manageragenten[4]. Dadurch ist die Eindeutigkeit von Agentenplattformen gewährleistet, weil Kommunikationsadressen den Namen des Rechners und eine Portnummer enthalten,

[3]Die Heimatplattform eines Agenten ist die Agentenplattform auf der der Agent erzeugt wurde. Bei mobilen Agenten muss demzufolge die Heimatplattform nicht mit der Agentenplattform identisch sein, auf der sich der Agent befindet.

[4]Der Manageragent einer Plattform hat den Namen „ams“ und beinhaltet die in FIPA definierten Funktionen einer AP und eines AMS sowie optional die Funktionen eines ACC und DF (vgl. [FIP02c]).

die innerhalb eines Rechners nur maximal einem Socket zugewiesen werden kann. Ein weiterer Vorteil besteht darin, dass alle Agenten automatisch zu jedem Manageragenten einer Plattform jeweils eine gültige Kommunikationsadresse kennen, unter der er erreichbar ist. Denn eine Abfrage dieser Adresse ist nicht möglich, da nur der Manageragent als *Agent Management System* (AMS) die entsprechenden Dienste anbietet und deshalb die Adresse bereits bekannt sein muss.

6.4.1 Registrierung und Suche von Agenten

Alle AMS sorgen untereinander dafür, dass sie sowohl die auf ihre Plattform migrierten Agenten als auch die Agenten kennen, die sie selbst erzeugt haben. Somit ist jeder Agent immer automatisch bei seiner aktuellen Plattform und seiner Heimatplattform registriert. Über den Dienst „register“ können sich Agenten aber auch bei anderen Plattformen registrieren lassen. Ändert sich die Beschreibung eines Agenten durch Nutzung entsprechender Dienste der aktuellen AP (z.B. Zustandsänderung, Migration oder Terminierung), so informiert der AMS die Heimatplattform durch Nutzung des Dienstes „modify“. Ändert aber der Agent seine Eigenschaften selbst (z.B. Zustand oder Kommunikationsadressen), informiert er den AMS der aktuellen sowie der Heimatplattform und die Plattformen, bei denen er sich zusätzlich registriert hat.

Um die aktuelle Beschreibung inklusive des Aufenthaltsortes und der Kommunikationsadressen eines Agenten zu erhalten um beispielsweise mit diesem Agenten zu kommunizieren, steht der Dienst „search“ beim AMS der Heimatplattform dieses Agenten zur Verfügung. Existiert die Heimatplattform nicht mehr, bleibt noch die Möglichkeit den globalen Gelbe-Seiten-Dienst des *Directory Facilitator* (DF) zu nutzen. Dort sind alle regionalen Dienste und Dienstanbieter mit ihren aktuellen Beschreibungen registriert, die ebenfalls nach jeder Änderung aktualisiert werden. Die Regionen bilden sich automatisch durch Nutzung der gleichen Registrierungsdaten durch mehrere DFs, indem sie entweder miteinander vernetzt sind oder denselben LDAP-Server verwenden.

6.4.2 Zustellung von Sprechakten

Ist trotz bekannter Kommunikationsadressen beispielsweise aufgrund eines vorübergehenden Aussetzen des Partners oder unterschiedlicher Transportprotokolle keine direkte Kommunikation möglich, können die Sprechakte mit dem Dienst „forward“ auch an den lokalen *Agent Communication Channel* (ACC) gesendet werden, der sie an den ACC des Partners weiterleitet. Ist der Empfänger vorübergehend nicht erreichbar, werden die Sprechakte vom ACC bis zu deren Abholung gespeichert. Die maximale Dauer der Speicherung von Sprechakten kann für jeden ACC individuell eingestellt werden.

6.4.3 Migration von Agenten

Für das Wandern mobiler Agenten von einer zur anderen Plattform wird mit dem Dienst „migrate“ eine sogenannte „starke“ Migration bereitgestellt, bei der neben dem Aufbau und den Fähigkeiten des Agenten auch der komplette Zustand inklusive der aktuellen Handlungen übertragen werden. Dadurch wird die Arbeitsweise des mobilen Agenten nicht beeinflusst und eine Migration ist jederzeit möglich. Eine Migration ist beispielsweise sinnvoll, um den Aufwand einer häufig auftretenden Kommunikation mit einem anderen Agenten zu reduzieren oder wenn die aktuelle Plattform überlastet ist, eine benötigte Ressource nicht besitzt oder beendet werden soll.

Während einer Migration läuft zwischen den Manageragenten der beiden beteiligten Plattformen das in Abbildung 6.3 dargestellte Protokoll ab. Zu Beginn prüfen beide Manageragenten durch Untersuchung entsprechender Vertrauenslisten gegenseitig ihre Vertrautheit. Wird Vertrauen gegeben, erfolgt beim Manager der Zielplattform eine Anfrage, ob der Agent migrieren darf und die notwendigen Ressourcen dort vorhanden sind. Denn eine Migration ist nur auf eine Agentenplattform mit kompatibler Laufzeitumgebung möglich. Ist auch der anwendungsspezifische Code des mobilen Agenten auf der Zielplattform vollständig vorhanden, erfolgt das Einfrieren und die Serialisierung des mobilen Agenten sowie der Transport einer Kopie zum Manager der Zielplattform, bevor diese Kopie auf der Zielplattform wieder deserialisiert und in den ursprünglichen Zustand versetzt wird. Anderenfalls wird vorher der für die Deserialisierung des Agenten notwendige anwendungsspezifische Code in Form eines Archivs auf die Zielplattform übertragen und der Laufzeitumgebung hinzugefügt. War die Migration erfolgreich wird neben dem Manager der Heimatplattform und dem DF auch der Manager der

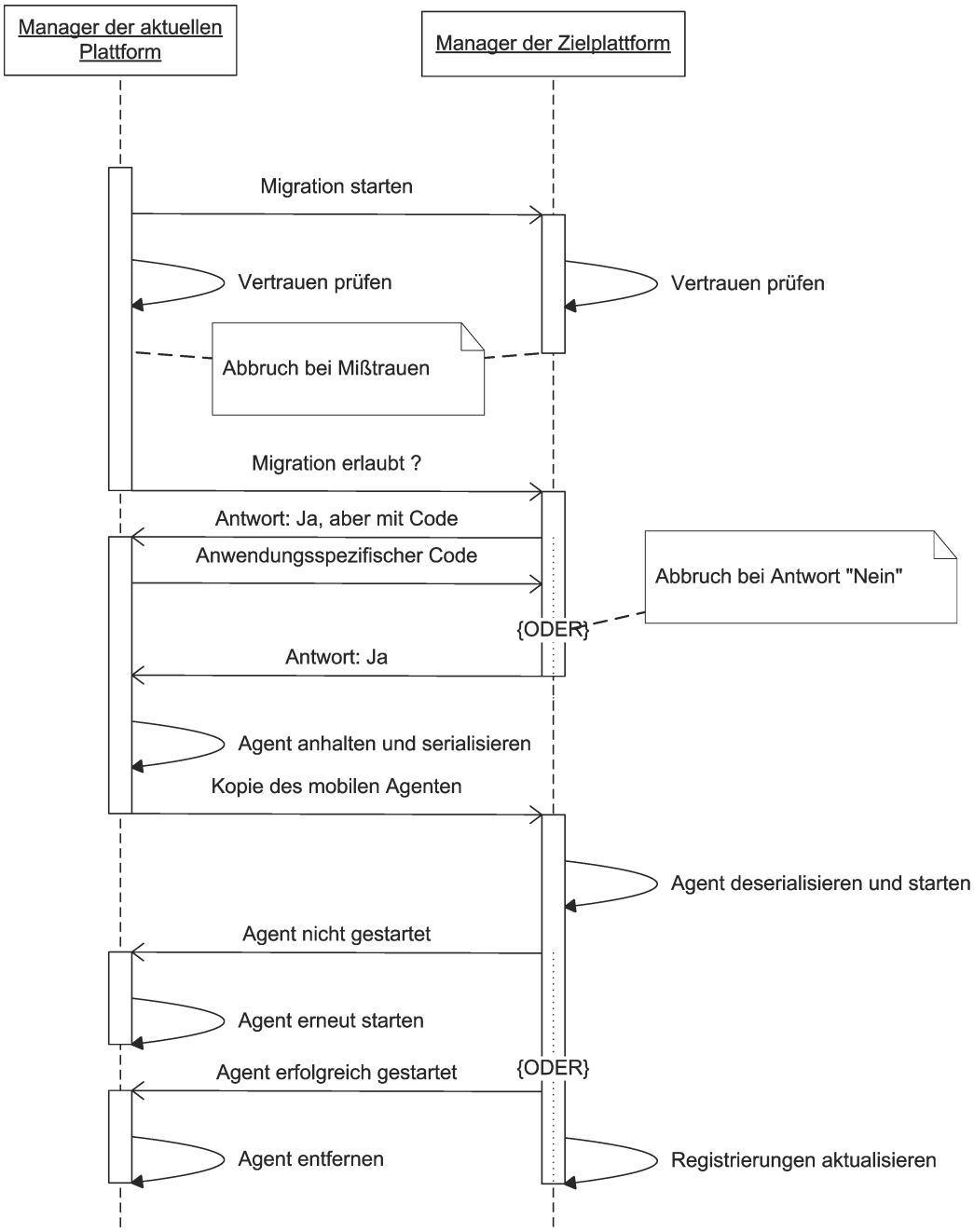

Abbildung 6.3: Sequenzdiagramm des Migrationsprotokolls

vorherigen Plattform informiert, der den eingefrorenen Agenten beendet. Anderenfalls wird er wieder in den Originalzustand überführt.

6.4.4 Erzeugen und Beenden von Agenten

Für das Erzeugen und Beenden von Agenten ist der Manageragent der Plattformen zuständig. Dazu bietet er entsprechende Dienste an, die von anderen Agenten genutzt werden können.

Für die Erzeugung eines Agenten ist die Angabe der initialen Konfiguration notwendig. Die Repräsentation der Konfigurationsdaten hängt von der Laufzeitumgebung ab, in der der Agent erzeugt werden soll. Die Erzeugung eines Agenten kann nur erfolgreich sein, wenn die Laufzeitumgebung alle in der Konfiguration angegebenen Ressourcen besitzt. Gegebenenfalls muss ähnlich wie bei der Migration der anwendungsspezifische Code des Agenten vorher auf die Agentenplattform übertragen werden.

Das Beenden von Agenten kann auf zwei verschiedene Arten vollzogen werden. Ein zwingendes bzw. sofortiges Beenden kann der Agent nicht hinauszögern oder verhindern. Dieses Vorgehen ist beispielsweise nötig, wenn der Agent nicht mehr reagiert. Beim angekündigten Beenden hingegen hat der Agent eine bestimmte Zeit lang die Möglichkeit sich auf dieses Ereignis durch entsprechende Aktionen vorzubereiten. Dadurch kann sich der Agent noch rechtzeitig in einen konsistenten Zustand bringen.

6.5 Persistenz von Agenten

Innerhalb des Managements agentenbasierter Systeme können eine Menge von Informationen anfallen, die verarbeitet und archiviert werden müssen. Sie können über verschieden lange Zeitspannen hinweg unterschiedliche Dringlichkeit und Wichtigkeit besitzen. Aus diesem Grund ist eine Datenhaltung notwendig, die sowohl die schnelle und zuverlässige Übermittlung dringlicher Daten (z.B. Agentenbeschreibungen oder Ereignisse) als auch die Persistenz und Sicherheit bestimmter Daten für definierte Zeiträume (z.B. Rechnungen) ermöglicht.

Die agententypische Form der Speicherung in einer Wissensbasis erlaubt zwar allen Bestandteilen des Agenten einen direkten Zugriff auf die Daten garantiert aber keine Persistenz. Jeden Agenten für die Persistenz seiner

Daten selbst verantwortlich zu machen ist unökonomisch, weil dadurch der Implementierungs- und Konfigurationsaufwand unnötig in die Höhe getrieben wird. Deshalb wurde eine für die verschiedenen Ansprüche geeignete Datenhaltung realisiert, die nicht nur auf den Managementaspekt von Anwendungen oder die Wissensbasis von Agenten sondern auf die Agentenplattform als Ganzes ausgerichtet ist und somit von allen Agenten gleichermaßen verwendet werden kann. Insbesondere werden Agenten und Agentenplattformen in dem Zustand (Wissen, Fähigkeiten und Aktionen) wieder gestartet, in dem sie aus welchem Grund auch immer beendet worden sind oder sich vorher befunden haben. Dieser Mechanismus ist sowohl durch den Administrator als auch aus dem System heraus nutzbar. Gerade für kritische Anwendungen in sich ändernden Umgebungen oder für langfristig lernende Systeme ist diese Infrastruktur eine zwingende Voraussetzung.

Um die Realisierung möglichst einfach zu halten und nicht jede Art von Statusinformationen einzeln zusammensuchen zu müssen, werden die Agenten komplett serialisiert. Die Serialisierung und Speicherung eines Agenten oder aller Agenten einer Plattform kann dabei in angegebenen Zeitabständen oder zu bestimmten Ereignissen (siehe Abschnitt 6.2.1) erfolgen, die beispielsweise eine Transaktion abschließen oder einen konsistenten Zustand definieren. Insbesondere vor dem Beenden eines Agenten oder einer Plattform ist eine Speicherung möglich. Diese Aufgabe wird vom Plattformmanager übernommen, da er für das Starten und Beenden von Agenten zuständig ist. Der Nachteil dabei ist allerdings, dass der Manageragent selbst nicht serialisiert werden kann, sondern die relevanten Informationen bezüglich des AMS, DF und ACC explizit gespeichert werden. Er darf deshalb keine zusätzlichen Aufgaben übernehmen, die den Erhalt von weiteren Statusinformationen erfordern.

Ähnlich zur Migration werden zuerst der agentenspezifische Code und anschließend der serialisierte Agent gespeichert, damit später die Deserialisierung durchgeführt werden kann. Die Speicherung selbst erfolgt unabhängig von der verwendeten Technologie über eine einheitliche Schnittstelle, an die je nach Konfiguration verschiedene Systeme (z.B. relationale Datenbanken, Verzeichnisdienste, Dateisysteme) angekoppelt sein können. Beim erneuten Starten der Plattform (Initialisierung des Manageragenten) oder beim Erzeugen eines Agenten wird automatisch geprüft, ob bereits serialisierte Daten vorhanden sind. In diesem Fall erfolgt das Laden und die Deserialisierung der Agenten, so dass sie wie ein mobiler Agent ihre Aufgaben genau in dem zuletzt gespeicherten Zustand fortführen. Anderenfalls wird die Plattform oder der Agent im initialen Zustand gestartet.

6.6 Tracingdienst

Das Tracing umfasst die Protokollierung, Weitergabe und Darstellung aller Aktionen eines Agenten innerhalb eines bestimmten Zeitraumes, um den Ablauf dieser Aktionen auch später noch außerhalb des Agenten nachvollziehen zu können. Dazu müssen alle relevante Ereignisse überwacht und zusammen mit entsprechenden Datenabfragen verarbeitet werden. Der Überwachungszeitraum kann sich entweder über die gesamte Lebenszeit des Agenten oder über einen bestimmten Zeitabschnitt erstrecken, der durch entsprechende Zeit- (z.B. von ... bis, ab jetzt) oder Ortsangaben (z.B. solange Agent auf Plattform X) spezifiziert wird. Damit lassen sich beispielsweise mobile Agenten automatisch während ihrer Reise überwachen, so dass alle auswärtigen Aktionen zurückverfolgt werden können. Zu den Tracinginformationen gehören unter anderem Daten über:

- den Weg des Agenten,
- die Nutzung von Diensten anderer Agenten und
- die Erbringung von Diensten für andere Agenten.

Bei umfangreichen Aufgaben und längerem Überwachungszeitraum können sehr viele Daten anfallen. Deshalb bietet die Tracingfunktion verschiedene Detaillierungsstufen an, die sich in der Menge der gelieferten Informationen unterscheiden. So wird beispielsweise im einfachen Modus nur aufgezeichnet, wo der Agent war und welchen Dienst er genutzt oder erbracht hat. Im erweiterten Modus sind dann auch Aussagen darüber möglich, wann der Agent wohin gegangen ist, wann die Nutzung oder Erbringung eines Dienstes begonnen hat, wie lange sie gedauert hat, mit welchen Parametern sie gelaufen ist und mit welchem Resultat sie beendet wurde. Zusätzlich kann die Überwachung auch auf bestimmte Dienste eingeschränkt sein oder Dienste ignorieren.

Die gesammelten Informationen werden an zentralen Stellen gespeichert und können für andere Managementaufgaben verwendet werden. So kann das Leistungsmanagement durch Auswertung dieser Daten Informationen über Diensthäufigkeit, Ausfälle oder Dauer von Diensten gewinnen. Auch zum Auffinden von Fehlerursachen durch das Fehlermanagement und zum Aufschlüsseln der Kosten durch das Abrechnungsmanagement können diese Daten verwendet werden. Im folgenden werden die Anforderungen an die Erfassung, Speicherung und Weitergabe der Tracingdaten verfeinert.

6.6.1 Aufgaben und Anforderungen

Die vollständige Sammlung der relevanten Daten findet innerhalb des Agenten durch Introspektion statt (siehe Abschnitt 6.2). Diese Daten müssen insbesondere bei mobilen Agenten gespeichert werden und unabhängig vom Aufenthaltsort noch während der Laufzeit des Agenten an den geforderten Stellen des verteilten Systems zur Verfügung stehen. Folgende Nutzer des Tracingdienstes sind denkbar:

- Administrator einer Anwendung
- Eigentümer eines Agenten
- Manageragent einer Plattform
- Sonstige Agenten

Der Administrator einer agentenbasierten Anwendung möchte unter Umständen die Beziehungen zwischen den einzelnen Agenten nachvollziehen. Der Eigentümer eines Agenten ist beispielsweise daran interessiert, welche Aktivitäten sich im Agentenkern abspielen. Ein Manageragent könnte den Tracingdienst in Anspruch nehmen, um die korrekte Funktionalität der auf der Plattform angebotenen Dienste zu kontrollieren. Aber auch andere Agenten möchten eventuell die von ihnen erzeugten Agenten und ihre Aktivitäten überwachen, um mögliche Fehler, Engpässe oder Konflikte rechtzeitig zu erkennen und gegebenenfalls geeignete Maßnahmen einleiten zu können.

Über Parameter des Tracingdienstes muss einstellbar sein, wie zeitnah der Transport der Daten stattfinden soll. Die Agentenplattformen können dabei als Zwischenspeicher für die bereits angefallenen Tracingdaten dienen. Um diese Daten wieder vollständig einsammeln zu können, muss ein Verzeichnis der bisher in Anspruch genommenen Speicherungsorte existieren. Für den Transport und die Speicherung der Daten sollten folgende Strategien angeboten werden:

- *Transport*: Versand (push) oder Abholung (pull)
- *Speicherung*: lokale Plattform (near) oder Heimat- bzw. angegebene Plattform (far)

Jegliche Kombinationen dieser Strategien sind erlaubt, wobei sie als Vorschläge zu verstehen sind. Kann eine Strategie nicht mehr erfüllt werden (z.B. Versand zum abgemeldeten Benutzer), muss automatisch zur anderen Strategie gewechselt werden.

6.6.2 Einbettung in das Agentensystem

Die Agentenintrospektion (siehe Abschnitt 6.2) und eine funktionierende Agenteninfrastruktur (siehe Abschnitt 6.4) sind Voraussetzung für die Realisierung der im vorherigen Abschnitt beschriebenen Aufgaben des Tracingdienstes. Diese Aufgaben lassen sich auf vier Grundfunktionen reduzieren:

1. Konfiguration des Tracingdienstes
2. Sammeln und Erzeugen der Tracingdaten
3. Versand und Speicherung der Tracingdaten
4. Präsentation der Tracingdaten

Die ersten beiden Funktionen werden durch die zu überwachenden Agenten selbst erbracht. Eine eigene Komponente (TracingBean) sammelt dabei selbständig die relevanten Informationen. Der Transport und die Speicherung der Daten erfolgt durch eigenständige Tracingagenten, die auf die verschiedenen Agentenplattformen aufgeteilt sind. Für die Präsentation der Tracingdaten wird eine separate Komponente (TracingGUI) bereitgestellt, die in der Regel auf der Heimatplattform des überwachten Agenten existiert und eine graphische Benutzerschnittstelle enthält. Es ergibt sich somit das in Abbildung 6.4 dargestellte Schema.

In den folgenden Abschnitten werden die einzelnen Bestandteile der Tracingfunktion genauer beschrieben.

6.6.3 TracingBean

Die TracingBean ist eine Komponente des überwachten Agenten, die Konfigurationsdaten für das Tracing von der TracingGUI entgegennimmt, die Introspektion des Agenten entsprechend steuert und die gesammelten Daten an die zugehörigen Tracingagenten weiterleitet.

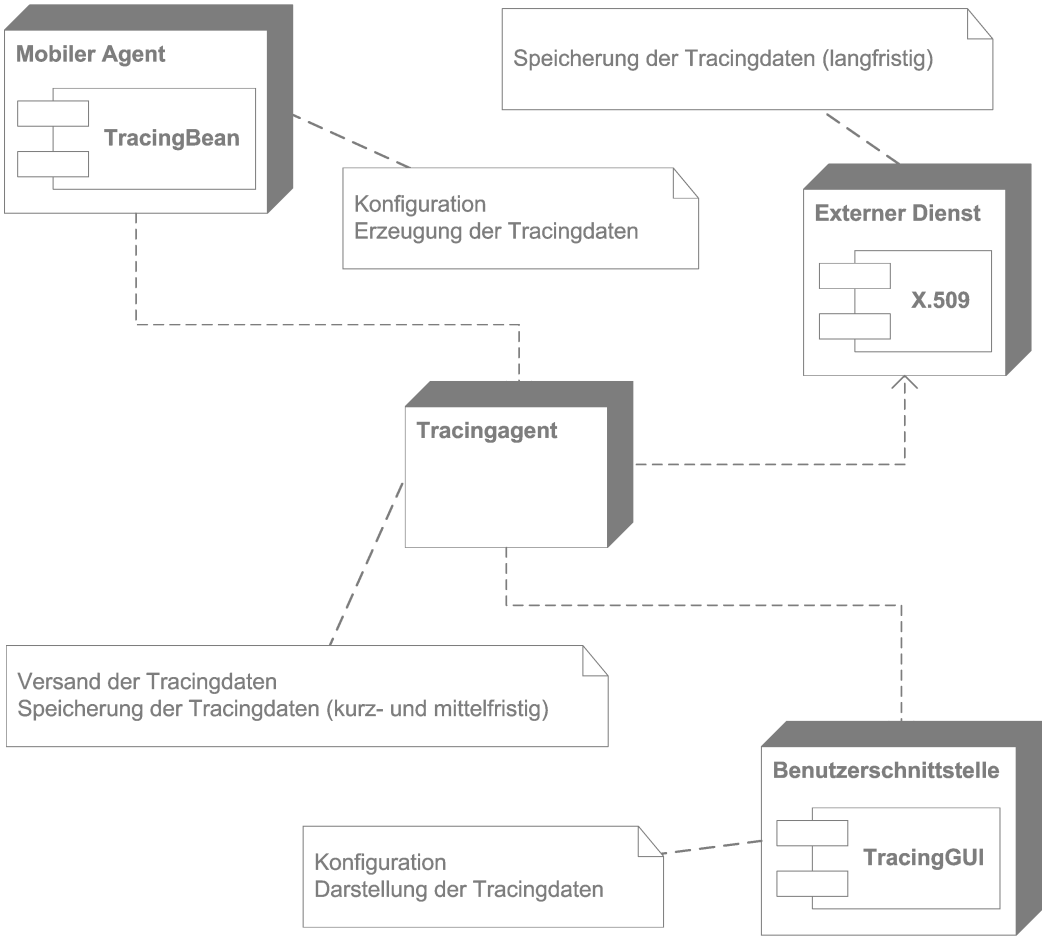

Abbildung 6.4: Komponenten des Tracingdienstes

Konfiguration des Tracingdienstes

Um insbesondere bei Nutzung des Tracingdienstes über einen längeren Zeitraum die Menge der Tracingdaten reduzieren zu können, ist eine Einschränkung bezüglich der zu erfassenden Daten möglich. Hierfür stehen entsprechend vordefinierte Überwachungsstufen bereit, um die Einschränkung auch ohne Kenntnis der existierenden Loggingdaten vornehmen zu können. Eine Überwachungsstufe ergibt sich aus der Zusammenfassung verschiedener Loggingtypen, die im Rahmen der Agentenintrospektion angeboten werden.

Der Umfang der produzierten Loggingdaten ist stark abhängig von den Aufgaben eines Agenten. Eine höhere Überwachungsstufe ist nicht immer automatisch mit einer größeren Menge von Tracingdaten gleichzusetzen. Stattdessen werden die Loggingdaten thematisch zusammengefasst. Am Beispiel von JIAC ergeben sich für den Tracingdienst folgende Überwachungsstufen:

1. Existenz und Ort
2. Dienstnutzung
3. Intentionenänderung und Planaufstellung
4. Planausführung und Sprechaktaustausch
5. Faktenaktualisierung und Zielauswahl

Erzeugung der Tracingdaten

In JIAC existieren zwei Arten von Loggingdaten, die als mögliche Quellen für die Erzeugung von Tracingdaten verwendet werden könnten.

Zum einen handelt es sich um relativ unstrukturierte Log4j-Meldungen zum Zwecke der Fehlersuche durch Entwickler. Eine Nutzung dieser Daten ist nur bedingt möglich, da sie weder genormt sind noch ihre Herkunft ermittelt werden kann. Ein Transport ohne Interpretation der Informationen wäre zwar denkbar, wird aber vom Tracingdienst vorerst nicht unterstützt.

Zum anderen werden durch die Komponenten des Agenten strukturierte Loggingdaten über Nachrichten zur Verfügung gestellt (siehe Abschnitt 6.2). Immer wenn ein entsprechendes Ereignis in einer Komponente eintritt, werden sämtliche Abonnenten (z.B. TracingBean) mittels solcher Nachrichten darüber informiert. Diese Nachrichten enthalten mindestens Herkunft und Typ der Information. In den meisten Fällen werden den Nachrichten auch

strukturierte Detailinformationen beigefügt. Diese Daten sind zur automatischen Verarbeitung wesentlich besser geeignet als Log4j-Meldungen, weil aufgrund der über die definierten Datenstrukturen vorgegebenen Semantik eine Interpretation der Daten möglich ist.

6.6.4 Tracingagent

Nach Erfassung der Daten besteht der nächste Schritt darin, diese Informationen auch außerhalb des Agenten zur Verfügung zu stellen. Dabei nimmt die TracingBean die von den Tracingagenten angebotenen Dienste in Anspruch. Mindestens auf der Heimatplattform des überwachten Agenten muss ein stationärer Tracingagent existieren.

Versand der Tracingdaten

Der Tracingagent nimmt die von der TracingBean zur Verbreitung vorgesehenen Informationen entgegen. Unabhängig davon, ob eine Speicherung in Anspruch genommen wird, können diese Daten entweder an externe Dienste oder an die in Abschnitt 6.6.5 beschriebene TracingGUI des Auftraggebers weitergeleitet werden. Der Tracingagent kann die Informationen aber auch auf Wunsch zur Abholung bereithalten. Um eine ungewollte Dauerspeicherung der Tracingdaten zu vermeiden, wird jeweils eine maximale Speicherungsdauer bei Übernahme der Informationen angegeben. Nach Ablauf dieser Zeitspanne (siehe Abschnitt 6.1) werden somit die Daten gelöscht.

Kurz- und mittelfristige Speicherung der Tracingdaten

Tracinginformationen können und sollen nicht immer sofort an den Auftraggeber des Tracings weitergeleitet werden. In diesem Fall wird eine temporäre Speicherung innerhalb der Tracingagenten vorgenommen. Dieser Speicherort eignet sich aber nicht für eine langfristige Speicherung der Daten, d.h. eine Speicherung über die Existenz des Agentensystems hinaus. Denn ein direkter Zugriff auf diese Daten ist nach Beendigung der Tracingagenten nicht mehr möglich.

Langfristige Speicherung der Tracingdaten

Die langfristige Speicherung von Tracingdaten kann von externen Verzeichnisdiensten außerhalb eines Agentensystems übernommen werden, sofern entsprechende Dienstzugangspunkte existieren. Der Vorteil ist, dass die Daten auch nach Beendigung der Tracingagenten noch zur Verfügung stehen. Nachteilig ist allerdings der in der Regel größere Kommunikationsaufwand. Ist ein externer Dienst nur bedingt erreichbar, scheidet dieser als langfristiger Speicherort aus.

Als Beispiel eines externen Verzeichnisdienstes ermöglicht ein LDAP-Server das Speichern und Abrufen von Daten in einer hierarchisch organisierten Baumstruktur. Wird für jeden Agenten ein neuer Teilbaum definiert, können unter Angabe des Agentennamens und Aktivitätszeitraumes die Tracingdaten eindeutig identifiziert werden.

6.6.5 TracingGUI

Die TracingGUI ist eine Komponente des überwachenden Agenten, die unter anderem eine Benutzerschnittstelle zur Verfügung stellt und somit auch die Nutzung des Tracingdienstes durch Menschen ermöglicht. Der Agent, der die TracingGUI zur Verfügung stellt, muss sich nicht zwingend auf der Plattform befinden, auf dem der Dienst gestartet wurde. Die Benutzerschnittstelle des Tracingdienstes sollte folgende Aufgaben erfüllen:

- Konfiguration des Tracingdienstes für einen Agenten
- Darstellung der Ausgaben des Tracingdienstes
- Wahrung der Sicherheitsvorschriften des Agentensystems

Der Benutzer muss in der Lage sein, die Überwachung eines Agenten zu starten und zu beenden. Vor und auch während der Überwachung kann die Konfiguration des Dienstes erfolgen. Dabei werden alle zur Auswahl stehenden Parameter angezeigt. Hierzu gehören insbesondere die zuvor erwähnten Transport- und Speicherstrategien.

Die zweite Aufgabe besteht in der Präsentation der übertragenen Tracinginformationen. Sofern diese Daten nicht automatisch zugestellt worden sind, müssen sie aus den verwendeten Zwischenspeichern entnommen werden. Verschiedene Filter ermöglichen dem Benutzer den Überblick über die gewonnenen Informationen zu behalten. Die Realisierung erfolgte zwar als lokale

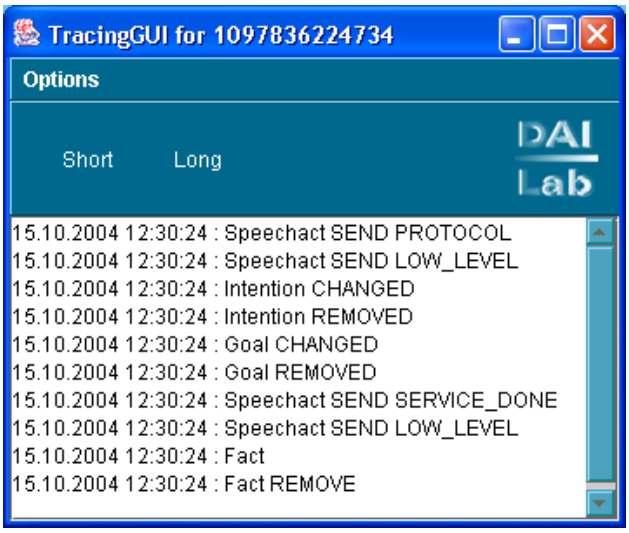

Abbildung 6.5: Benutzerschnittstelle des Tracingdienstes

GUI (siehe Abbildung 6.5), kann aber durch eine geräteunabhängige Benutzerschnittstelle ersetzt werden.

Für die Kommunikation zwischen dem Benutzer und dem System müssen die geforderten Sicherheitsrichtlinien eingehalten werden. Sie werden durch notwendige Authentisierungsmechanismen ergänzt. Bei der Verarbeitung der Tracinginformationen innerhalb des Systems muss die eindeutige Identifikation des Auftrages gewährleistet bleiben.

6.7 Werkzeuge für die Administration

Wie beim Tracing (siehe Abschnitt 6.6.1) sollten auch alle anderen Funktionen von MIAS nicht nur von Agenten sondern auch von Menschen genutzt werden können. Aus diesem Grund wurden Werkzeuge entwickelt, mit denen Administratoren agentenbasierter Systeme oder Eigentümer von Agenten unter Nutzung der Basismechanismen von MIAS verschiedene Informationen über ein Agentensystem oder einen Agenten präsentiert bekommen und Einfluss auf die Fähigkeiten, das Wissen und die Verhaltensweisen der Agenten nehmen können. Benutzerfreundliche Oberflächen stellen die Basisfunktionen abhängig von der Rolle des Benutzers (z.B. Administrator, Dienstanbieter oder Dienstnutzer) übersichtlich und intuitiv zur Verfügung. Aufgrund der

unterschiedlichen Anforderungen wurden am Beispiel von JIAC die folgenden vier Werkzeuge entwickelt:

- Agentenmonitor
- Plattformmonitor
- Systemkonfigurator
- Anwendungsmonitor

Jedes dieser Werkzeuge hat seine Vor- und Nachteile, die zusammen mit ihrer Funktionsweise im folgenden genauer beschrieben werden.

6.7.1 Agentenmonitor

Der Agentenmonitor ist eine Komponente innerhalb eines Agenten, die eine Benutzerschnittstelle zur Administration dieses Agenten und sämtlicher Bestandteile zur Verfügung stellt (siehe Abbildung 6.6). Eine gleichzeitige Überwachung mehrerer Agenten ist aber aufgrund der vielen Oberflächen sehr unübersichtlich und die Überwachung eines entfernten Agenten ist mit diesem Werkzeug nicht möglich. Dafür ist der Agentenmonitor aufgrund der direkten Zugriffe auf den Agenten sehr performant und erlaubt den Agenten bis ins Detail zu überwachen.

Eine Liste von Schaltflächen zeigt den aktuellen Lebenszykluszustand des Agenten an, der darüber auch verändert werden kann. Für den Schrittmodus kann die Schrittweite und eine Wartezeit zwischen den Arbeitsschritten angegeben werden. Die verschiedenen Registerkarten des Hauptfensters ermöglichen folgende Aktionen:

- *Components*: Erlaubt die Anzeige sowie das Entfernen, Hinzufügen und Austauschen von Komponenten. Zu jeder Komponente können die Leistungsstufe, die Loggingstufe und die Properties geändert werden.
- *Knowledge*: Zeigt die Ontologien, Fakten, Ziele, Intentionen, Aktionen, Planelemente, Dienste und Regeln des Agenten.
- *Log*: Zeigt Loggingmeldungen des Agenten an.
- *Messages*: Protokolliert den Nachrichtenaustausch zwischen Komponenten.

- *Edit Goal*: Erlaubt das Editieren, Aufstellen und Widerrufen von Zielen.
- *Log4j*: Erlaubt die Log4j-konforme Konfiguration des Loggings.

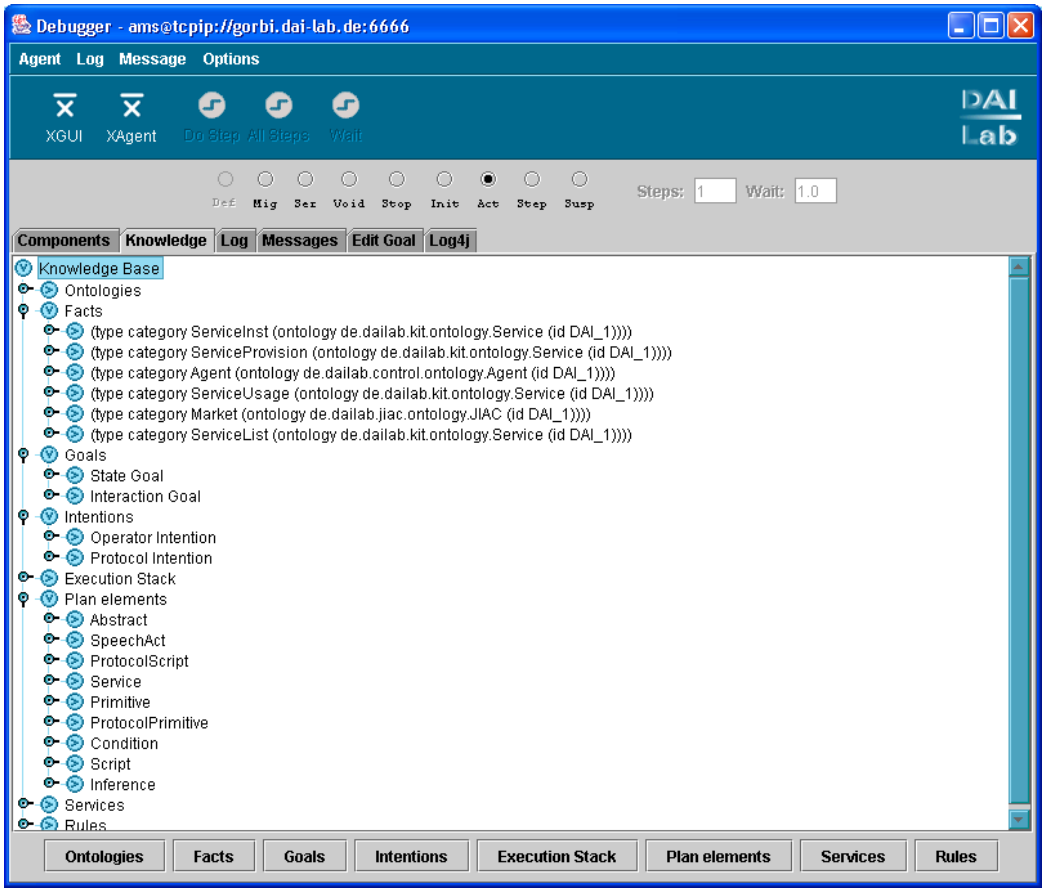

Abbildung 6.6: Benutzerschnittstelle des Agentenmonitors

Über die Menüleiste des Werkzeuges können noch weitere Einstellungen vorgenommen werden.

6.7.2 Plattformmonitor

Der Plattformmonitor ist Teil einer speziellen Komponente jedes Manageragenten. Sie visualisiert über eine Benutzerschnittstelle die auf jeweils dieser Plattform existierenden Agenten entsprechend ihres Typs und Zustands. Die Kommunikation zwischen Agenten wird über kurzzeitig eingeblendete und beschriftete Linien angezeigt. Dadurch sind die Agenten und ihre Beziehungen untereinander jederzeit auf einen Blick zu sehen (siehe Abbildung 6.7). Eine gleichzeitige Überwachung mehrerer Plattformen ist aufgrund der vielen Fenster sehr unübersichtlich und die Überwachung einer entfernten Plattform ist mit diesem Werkzeug nicht möglich. Desweiteren sind keine Details der Agenten administrierbar, da die Kontextmenüs der Agenten und der Plattform zurzeit nur wenige Aktionen zulassen.

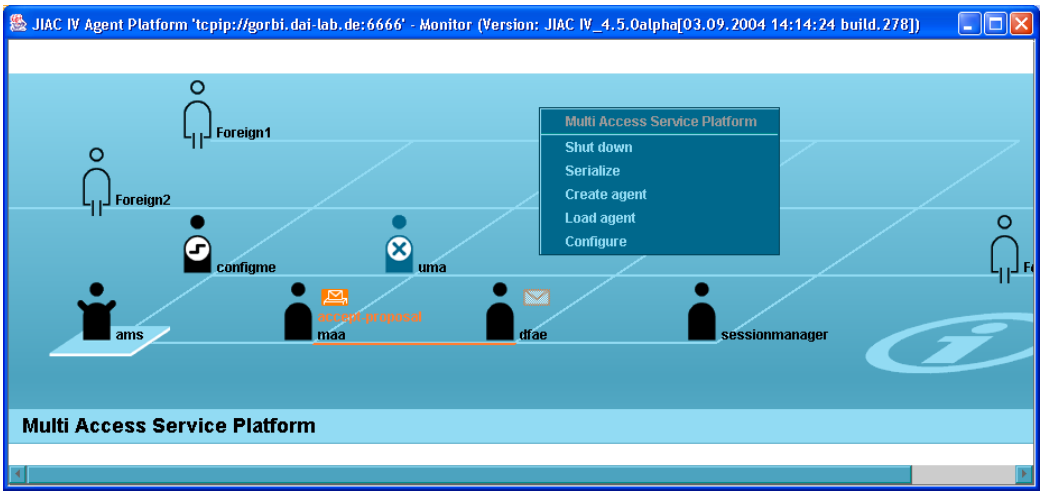

Abbildung 6.7: Benutzerschnittstelle des Plattformmonitors

Das Kontextmenü der Plattform erlaubt das Herunterfahren sowie das Serialisieren und Speichern der Plattform, das Ändern der Anzeigedauer der Kommunikationslinien und das Laden eines serialisierten sowie das Erzeugen eines neuen Agenten. Die Agenten können über Kontextmenüs beendet, serialisiert und gespeichert sowie migriert werden. Ebenso lassen sich der Lebenszykluszustand verändern und neue Komponenten hinzufügen.

6.7.3 Systemkonfigurator

Der Systemkonfigurator ist ein von Menschen nutzbarer Dienst, der über eine geräteunabhängige Benutzerschnittstelle bereitgestellt wird (siehe Abbildung 6.8). Er erlaubt die Administration entfernter Plattformen, Agenten und Dienste über verschiedene Endgeräte (z.B. PC, PDA, Handy oder Telefon) ohne Installation zusätzlicher Software. Die Dienstbasiertheit dieses Werkzeuges ermöglicht zwar nur Zugriffe auf die internen Daten der aktiven Agenten, bietet aber weitere JIAC-spezifische Merkmale wie Authentisierung, Autorisierung, Verschlüsselung und Abrechnung. Aufgrund der weniger performanten Dienstnutzung und der einfach gehaltenen Benutzerschnittstelle sind keine dynamischen oder ereignisbasierten Daten verfügbar.

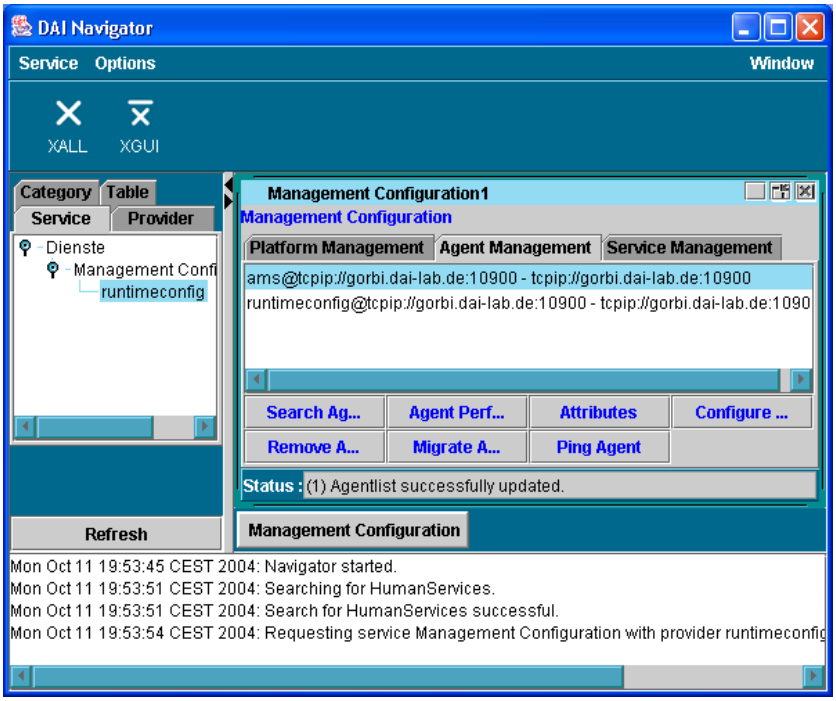

Abbildung 6.8: Benutzerschnittstelle des Systemkonfigurators

Der Systemkonfigurator ist in Plattform-, Agenten- und Dienstmanagement unterteilt. Das Plattformmanagement erlaubt die Suche nach Plattformen und für ausgewählte Plattformen die Änderung der Beschreibung, die Anzeige der Agenten sowie die Erzeugung neuer Agenten. Über das Agentenmanagement können Agenten gesucht, beendet, migriert und konfiguriert sowie deren Beschreibungen geändert und deren Leistungsfähigkeit und Erreichbarkeit überwacht werden. Zur Konfiguration eines Agenten gehört das An-

zeigen, Hinzufügen und Entfernen von Komponenten, Fakten, Zielen, Planelementen und Diensten. Das Dienstmanagement ermöglicht die Suche nach Diensten und die Änderung von Dienstbeschreibungen. Die Suche nach Plattformen, Agenten und Diensten kann bezüglich bestimmter Merkmale eingeschränkt werden.

6.7.4 Anwendungsmonitor

Der Anwendungsmonitor ist ein agentenbasiertes Werkzeug, das über eine Benutzerschnittstelle die ausführliche Administration einer verteilten Anwendung bestehend aus Plattformen, Agenten und Diensten erlaubt (siehe Abbildung 6.9). Um eine Überlastung zu vermeiden, werden nur die durch den Benutzer ausgewählten Agenten und Informationen überwacht. Prinzipiell können aber alle Agenten im Detail administriert werden, wobei auch dynamische Änderungen berücksichtigt werden. Der Zugriff auf agenteninterne Daten ist zwar auch im inaktiven Zustand des Agenten möglich, allerdings werden diese Daten nur unverschlüsselt übertragen.

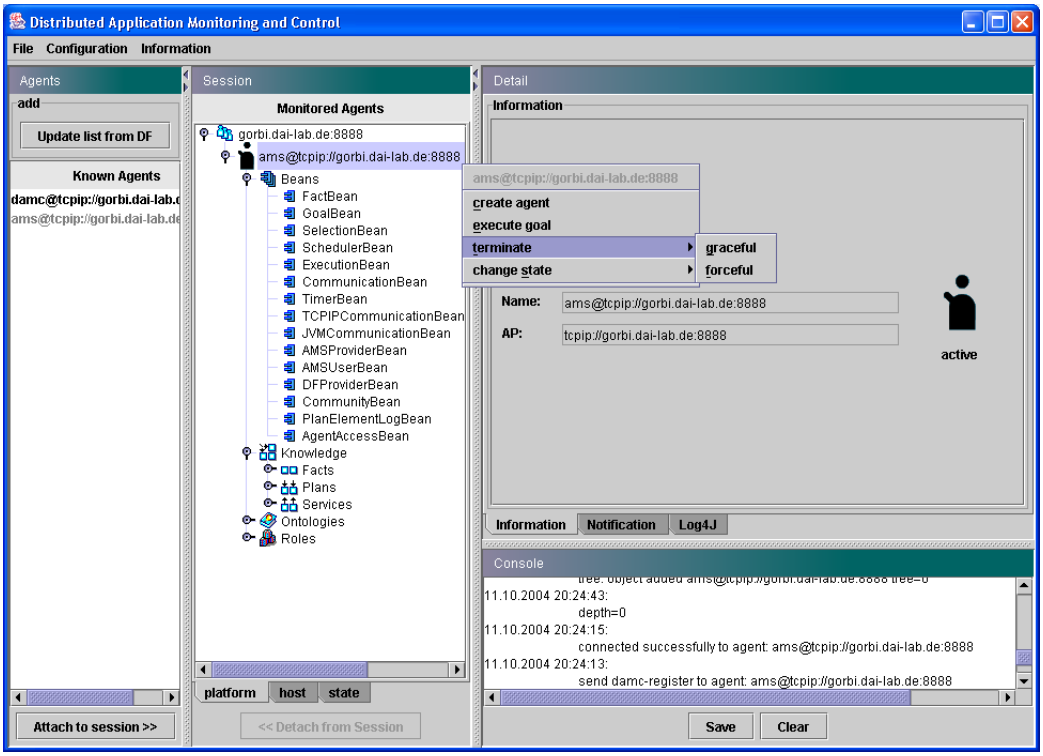

Abbildung 6.9: Benutzerschnittstelle des Anwendungsmonitors

Der im Werkzeug integrierte DF kann jederzeit umkonfiguriert werden, so dass im Prinzip alle Agenten der verschiedensten Anwendungen gefunden werden können. Die ausgewählten Agenten können nach Plattformen, Rechner oder Lebenszykluszustand gruppiert werden. Zu jedem Agenten werden die Komponenten, Fakten, Planelemente, Dienste, Ontologien und Rollen angezeigt. Über Kontextmenüs können diese Agenten beendet, ein Ziel aufgestellt, der Lebenszykluszustand geändert oder ein gleicher Agent erzeugt werden. Regelmäßige Pings testen die Erreichbarkeit aller Agenten. Der Administrator kann sich über SMS oder Email informieren lassen, wenn sich die Erreichbarkeit eines bestimmten Agenten geändert hat. Auch die Anzeige der Log4j-Meldungen einzelner Agenten ist möglich.

6.8 Zusammenfassung

In diesem Kapitel wurden alle Funktionen der mittleren Ebene der Managementinfrastruktur detailliert beschrieben. Diese Basismechanismen basieren auf entsprechenden Schnittstellen der unterstützten Agentenarchitektur und definieren eine einheitliche Schnittstelle in Form von Diensten für die höherwertigen Managementfunktionen. Dabei können allgemeine Funktionen, die alle Agenten zur Verfügung stellen, und agentenübergreifende Funktionen, die spezielle Agenten bereitstellen, unterschieden werden. Alle beschriebenen Basismechanismen wurden am Beispiel der Agentenentwicklungsumgebung JIAC implementiert.

Zu den allgemeinen Funktionen gehören ein Zeitgeber sowie die Introspektion und Manipulation einzelner Agenten. Mit Hilfe des Zeitgebers ist es möglich sich asynchron über Zeitpunkte oder abgelaufene Zeitspannen informieren zu lassen. Die Ausführung zeitgesteuerter Aktionen ist dadurch sehr einfach umzusetzen. Läuft ein Agent zum Zwecke der Fehlersuche im Schrittmodus, so haben die Stillstandszeiten keine Auswirkung auf die überwachten Zeitspannen und somit auf die Arbeitsweise des Agenten. Die Introspektionsfunktion ermöglicht Bestandteile eines Agenten zu beobachten. Dabei können sowohl Zustandsinformationen abgefragt als auch Änderungen durch Angabe eines Ereignismusters zeitnah überwacht werden. Ereignisse und Ereignismuster sind in einer Ontologie definiert, die sich zwar vorerst auf BDI-Elemente von JIAC beschränkt aber noch um andere Elemente erweitert oder verfeinert werden kann. Mit Hilfe der Manipulationsfunktion können Änderungen an einem Agenten vollzogen oder das Ausführen von Aktionen durch einen Agenten veranlasst werden. Die Agentenarchitektur sollte allerdings dafür

sorgen, dass keine Inkonsistenzen innerhalb des Agenten durch fehlerhafte oder unvollständige Manipulationen entstehen.

Zu den agentenübergreifenden Basismechanismen, die ein ausgezeichneter Manageragent auf jeder Agentenplattform anbietet, gehören Dienste zur Verwaltung der Agentenplattformen (hier Agenteninfrastruktur genannt) sowie zur Persistenz und zum Tracing von Agenten. Typische Dienste der Agenteninfrastruktur, wie sie beispielsweise auch FIPA definiert, sind die Registrierung und die Suche von Agenten und Diensten, die Zustellung von Nachrichten zwischen Agenten sowie das Erzeugen, die Migration und das Beenden von Agenten. Zur Gewährleistung der Persistenz von Agenten können diese in regelmäßigen Abständen oder vor dem Beenden gespeichert und beim nächsten Start wieder geladen werden, so dass sie ihre Aufgaben genau in dem zuletzt gespeicherten Zustand fortführen. Das Tracing umfasst die Protokollierung, Weitergabe und Darstellung aller Aktionen eines (mobilen) Agenten innerhalb eines bestimmten Zeitraumes, um den Ablauf dieser Aktionen auch später noch außerhalb des Agenten nachvollziehen zu können.

Die beschriebenen Basismechanismen werden beispielhaft von vier entwickelten Werkzeugen zur Administration eines Agentensystems durch Menschen genutzt. Dazu gehören ein Agentenmonitor, Plattformmonitor, Systemkonfigurator und Anwendungsmonitor. Der Agentenmonitor bietet eine Benutzerschnittstelle für die Introspektions- und Manipulationsfunktion eines einzelnen, lokalen Agenten. Der Plattformmonitor erlaubt den Zugriff auf die Agenteninfrastruktur und die Persistenzfunktion einer einzelnen, lokalen Agentenplattform. Der Systemkonfigurator fasst ein Großteil der Funktionalität von Agenten- und Plattformmonitoren unter einer einfachen geräteunabhängigen Benutzerschnittstelle zusammen. Allerdings kann nur bei aktivierten Agenten eine Introspektion und Manipulation ihrer internen Daten erfolgen. Der Anwendungsmonitor erlaubt die permanente Überwachung und Steuerung einer agentenbasierten Anwendung bestehend aus einer angegebenen Menge von Plattformen und Agenten. Bei Änderung der Erreichbarkeit eines Agenten ist eine Benachrichtigung über SMS oder Email möglich.

Kapitel 7
Abrechnungsmanagement

Mit Blick auf die kommerzielle Verwertung agentenbasierter Dienste in einem liberalisierten Telekommunikationsmarkt muss dem Bereich des Abrechnungsmanagements besondere Aufmerksamkeit geschenkt werden. Aufbauend auf der in Abschnitt 2.2 durchgeführten Analyse und Bestandsaufnahme wird in diesem Kapitel ein generischer Abrechnungsmechanismus vorgestellt (vgl. [KHA07]). Insbesondere soll dieser Mechanismus den hohen Anforderungen bezüglich der Dienstorientierung, Kundenindividualität, Änderungsdynamik und Integrationsfähigkeit gerecht werden. Eine Berücksichtigung von Subdienstnutzungen ist wie in üblichen Multiprovider-Umgebungen zwingend notwendig. Bestehende agentenbasierte Systeme sollen um diese Abrechnungsfähigkeit erweitert werden können, ohne Änderungen an der Implementierung vornehmen zu müssen. Ein System ohne Abrechnung von Diensten muss dennoch möglich bleiben.

Auch beim Abrechnungsmanagement muss seitens der funktionalen Verteilung zwischen der externen und der agenteninternen Sicht differenziert werden. Die Abrechnung der Dienste wird durch spezialisierte Agenten unterstützt. Ein solcher Agent enthält unter anderem Wissen über die Gebührenmodelle, die für die Dienste angewendet werden können. Zu einem solchen Modell zählt insbesondere auch das Wissen über die Art und Granularität, den Ort der Bereitstellung und die Art der Verarbeitung der relevanten Informationen. In einem Agenten ist daher eine Komponente vorzusehen, die diese Informationen während eines Dienstablaufs sammelt, verarbeitet und nach außen weiterreicht. Für eine permanente Verarbeitung und Visualisierung der Abrechnungsinformationen müssen sie bei jeder Änderung oder in definierbaren Zeitintervallen zur Verfügung gestellt werden können. Dabei muss auf das Mitführen eindeutiger Dienstsitzungskennungen geachtet werden. Diese

Kennungen sind notwendig, um Dienstnutzungen unterschiedlicher Benutzer voneinander unterscheiden zu können. Im Falle einer nicht-anonymen Dienstnutzung muss die Dienstnutzungskennung einem realen Benutzer zugeordnet werden können.

7.1 Grobkonzept

Das Grobkonzept definiert die am Abrechnungsprozess beteiligten Rollen und deren Aufgaben sowie die zwischen den Rollen ablaufenden Interaktionen. Wie in Abbildung 7.1 dargestellt können die einzelnen Rollen jeweils einer der folgenden vier Organisationen (Rollengruppen) zugeordnet werden:

- *Kunde*: Ein Kunde besitzt einen Vertrag mit einem Anbieter, der es angegebenen Benutzern erlaubt bestimmte Dienste zu den ausgehandelten Konditionen zu nutzen. Er ist für die Begleichung der in Rechnung gestellten Gebühren verantwortlich.
- *Anbieter*: Ein Anbieter erbringt für seine Kunden Dienste zu den im Vertrag ausgehandelten Konditionen. Dabei kann er auf Dienste von sogenannten Drittanbietern zurückgreifen. Für die Ermittlung der angefallenen Gebühren stehen verschiedenste von Tarifentwicklern angebotene Tarifmodelle zur Verfügung.
- *Drittanbieter*: Ein Drittanbieter ist ein Anbieter, dessen Kunden keine Endanwender sondern selbst Anbieter von höherwertigen Diensten sind.
- *Tarifentwickler*: Ein Tarifentwickler bietet unterschiedlichste Tarifmodelle an, die von verschiedenen Dienstanbietern für deren Abrechnung verwendet werden können.

Die verschiedenen im Zusammenhang mit der Abrechnung von Diensten stehenden Aufgaben lassen sich in die folgenden drei Bereiche aufteilen:

1. *Konfiguration*: Zur Konfiguration der Abrechnung gehören zum einen die Kunden- bzw. Benutzerverwaltung und zum anderen die Tarifverwaltung. Zur Unterstützung der Kundenindividualität können jedem registrierten Kunden eigens zwischen den beteiligten Personen oder Agenten ausgehandelte Verträge zugeordnet werden. Der Kunde hat

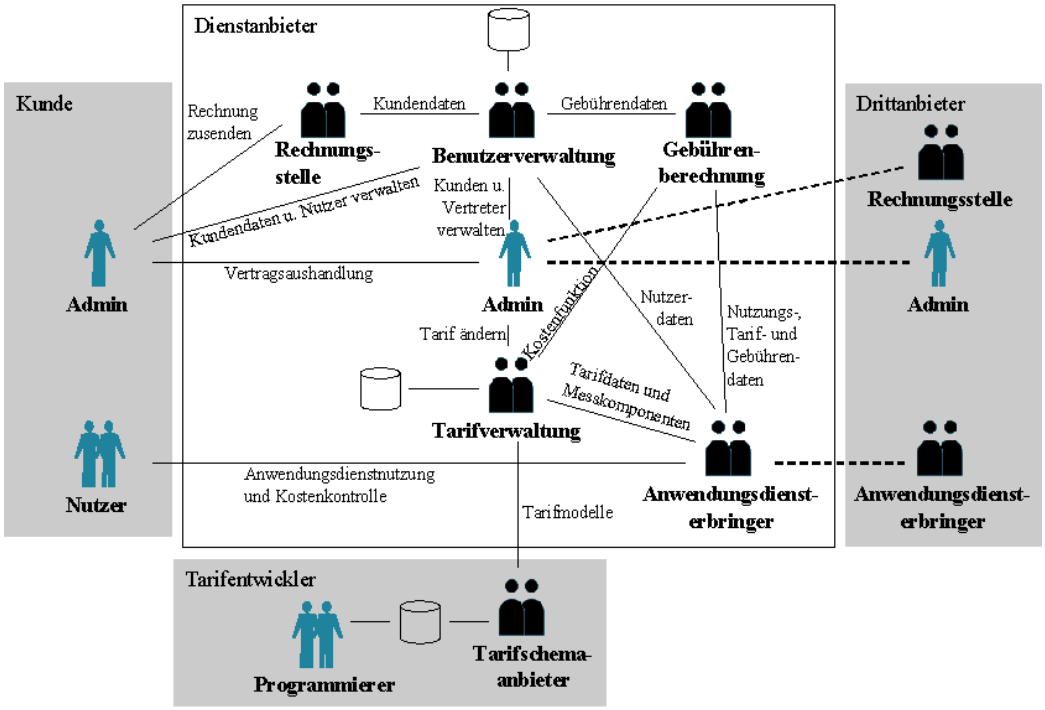

Abbildung 7.1: Rollen und Interaktionen beim Abrechnungsmanagement

anschließend die Möglichkeit einzelne Benutzer anzulegen und entsprechend des Vertrages zugehörige Einstellungen vorzunehmen (siehe Abschnitt 7.6). Für eine nutzungsabhängige und dienstorientierte Abrechnung können Tarife nicht nur auf allgemeingültigen sondern auch auf dienstspezifischen Modellen basieren. Ein Tarifmodell besteht u.a. aus einer Liste von gebührenrelevanten Ereignistypen, einem Parametrisierungsschema und einer auf Ereignissen und Parametrisierung beruhenden Kostenfunktion (siehe Abschnitt 7.5). Zur Sammlung der zu einer Dienstnutzung gehörenden Ereignisse müssen entsprechende Messkomponenten zur Verfügung stehen (siehe Abschnitt 7.2). Der Dienstanbieter hat nun die Möglichkeit verschiedene Tarife durch Parametrisierung von Tarifmodellen zu definieren, die von Tarifentwicklern angeboten werden. Für jedes zu verwendende Tarifmodell müssen den gebührenberechnenden Agenten die Kostenfunktion und den diensterbringenden Agenten die Messkomponenten hinzugefügt werden. Um mit einer hohen Änderungsdynamik zurechtzukommen, sollte dieser Prozess durch geeignete Werkzeuge unterstützt werden und weitestgehend automatisiert erfolgen.

2. *Dienstnutzung*: Zu Beginn jeder Dienstnutzung muss der für den Benutzer und Dienst gültige Tarif ermittelt und das Sammeln der im Tarifmodell angegebenen Ereignisse in den beteiligten Messkomponenten angestossen werden. Dadurch wird die Menge der gesammelten Daten auf ein Mindestmaß reduziert. Die Übermittlung der gemessenen Ereignisse kann über verschiedene Strategien erfolgen, wobei hier die ereignis- oder zeitabhängige Zustellung (Push) favorisiert wird. Optional wird die Kostenkontrolle aktiviert, die nach jedem durch eine Messkomponente übermittelten Ereignis die momentanen Kosten berechnen lässt und diese an den Dienstnutzer weiterleitet (siehe Abschnitt 7.4). Mit dem Ende der Dienstnutzung müssen die vollständigen Gebühren an den Rechnungssteller übergeben und alle beteiligten Messkomponenten aufgefordert werden, die Sammlung der relevanten Nutzungsdaten zu beenden. Um einen reibungslosen Ablauf zu garantieren muss dieser Prozess automatisiert erfolgen (siehe Abschnitt 7.7).

3. *Rechnungsstellung*: Alle in einem gewissen Zeitraum angefallenen Gebühren müssen den entsprechenden Kunden in Rechnung gestellt werden. Dazu müssen die Häufigkeit oder der Zeitpunkt für die Rechnungsstellung, der Tarif für die Ermittlung des Rechnungsbetrages, die Art der Zustellung und die Zahlungsmodalitäten im Vertrag angegeben sein (siehe Abschnitt 7.8). Um eine pünktliche Rechnungsstellung zu gewährleisten, sollte dieser Prozess möglichst automatisiert erfolgen. Ein Pre- oder Micro-Payment, bei dem auch anonyme Benutzer kostenpflichtige Dienste nutzen können, wird hier nicht betrachtet.

Die Konfiguration ist somit eine vorbereitende Managementphase für den eigentlichen Abrechnungsprozess, der in Abbildung 7.2 vollständig von der Sammlung der relevanten Daten bis hin zur Rechnungsstellung dargestellt ist.

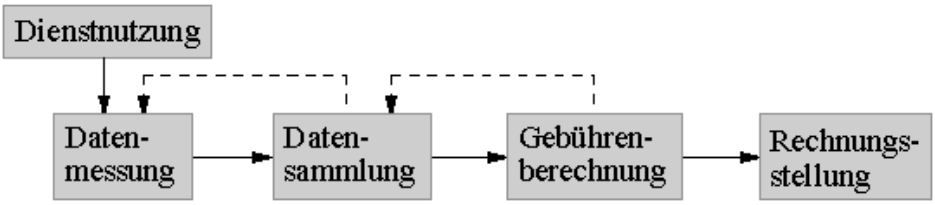

Abbildung 7.2: Aufgaben des Abrechnungsprozesses

Im folgenden Abschnitt wird ein Konzept zur Messung und Sammlung abrechnungsrelevanter Informationen entworfen. Die mit der Sammlung einhergehende Zusammenfassung und Gruppierung der Daten vereinfacht die

Gebührenberechnung, die die in Abschnitt 7.5 beschriebene Definition von Tarifen berücksichtigt. Die in den Tarifen verwendeten Gebührenmodelle geben die Rahmenbedingungen für die Sammlung der abrechnungsrelevanten Informationen vor. Die berechneten Gebühren sind die Voraussetzung für die in Abschnitt 7.8 beschriebene Rechnungsstellung. Die Datenmessung, Datensammlung und Gebührenberechnung kann iterativ[1] erfolgen. Die Rechnungsstellung sollte hingegen nur am Ende der Dienstnutzung durchgeführt werden.

7.2 Messung und Sammlung von Dienstnutzungsdaten

Bei der Messung gebührenrelevanter Ereignisse zu einzelnen Dienstnutzungen kann auf die Introspektionsfähigkeit der Agenten zurückgegriffen werden (siehe Abschnitt 6.2). Eine Erweiterung der Introspektion um beispielsweise dienstspezifische Ereignisse ist dabei problemlos möglich. Es muss nur sichergestellt sein, dass der Anbieteragent alle benötigten Vermittlungskomponenten besitzt, um sich über Ereignisse zu den im Tarifmodell angegebenen Ereignismustern informieren lassen zu können. Die Messung und Verarbeitung der Ereignisse gehört also zum agenteninternen Managementanteil. Damit ergibt sich für die Anbieteragenten der in Abbildung 7.3 dargestellte Aufbau.

Zu Beginn einer Dienstnutzung wird ein Dienstnutzungsobjekt erzeugt, das unter anderem eine eindeutige Kennung und die für die Berechnung der Gebühren relevanten Ereignismuster enthält. Gemäß dieser Ereignismuster wird unter Angabe der Nutzungskennung bei den zugehörigen Vermittlerkomponenten die Messung aktiviert. Eine Dienstnutzung besteht in der Regel aus einer Menge von Aktionen, die verschiedene Ereignisse auslösen können. Diese Ereignisse werden von den Messkomponenten an die angemeldeten Vermittlerkomponenten übergeben, die daraufhin die Abrechnungskomponente informiert, wenn sich das Ereignis einem Ereignismuster und einer der angegebenen Dienstnutzungen zuordnen lässt. Die Abrechnungskomponente speichert alle relevanten Ereignisse geordnet nach Dienstnutzung und Ereignismuster in einer internen Datenstruktur und informiert gegebenenfalls die

[1] Bei der Iteration werden die Aktionen während der Dienstnutzung mehrmals mit den bis dahin angefallenen Daten ausgeführt.

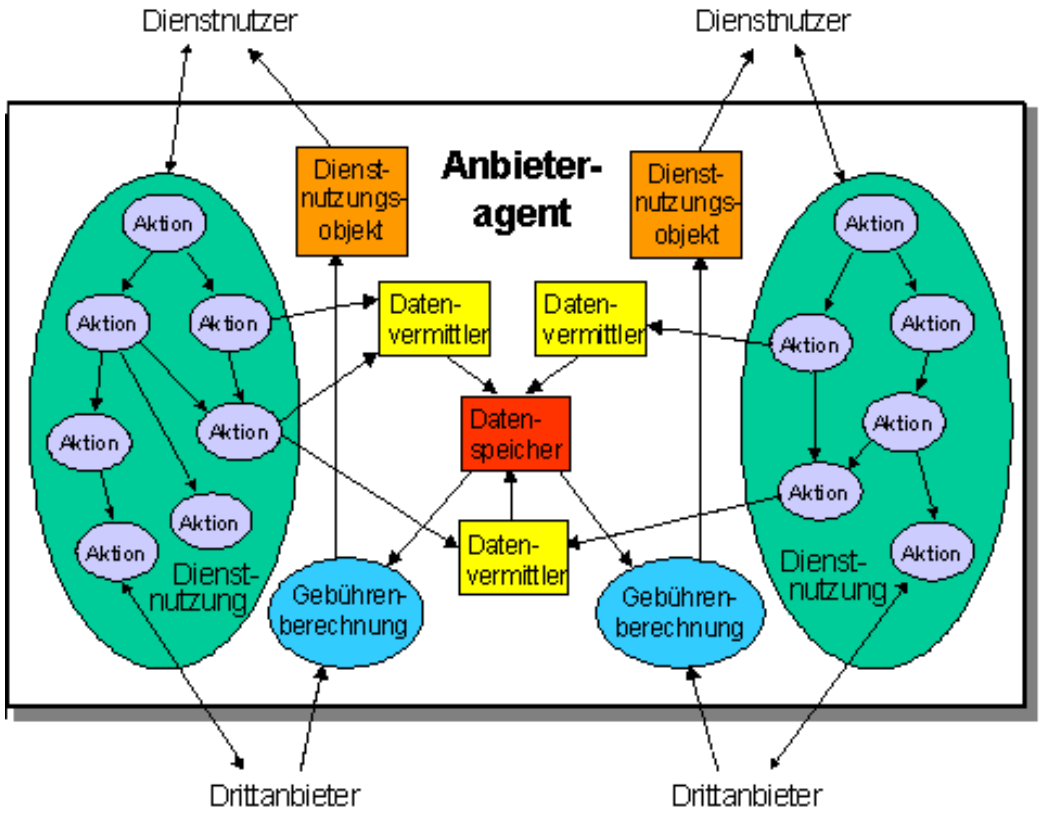

Abbildung 7.3: Aufbau eines Anbieteragenten zur Messung von Dienstnutzungen

zugehörigen Gebührenberechnungsprozesse. Die Deaktivierung der Messung erfolgt sofort nach Ende der Dienstnutzung.

7.3 Gebührenberechnung

Wie in Abbildung 7.3 bereits dargestellt existiert zeitgleich zu jeder kostenpflichtigen Dienstnutzung ein Prozess, der aus den gespeicherten Ereignissen und eventuell vorhandener Kosten für Subdienstnutzungen die aktuelle Gebühr unter Verwendung des gültigen Tarifs berechnet und im Dienstnutzungsobjekt ablegt (siehe Abschnitt 7.5). Für diesen Prozess wurde das in Abbildung 7.4 dargestellte Aktivitätsdiagramm entworfen, das nun genauer erläutert wird.

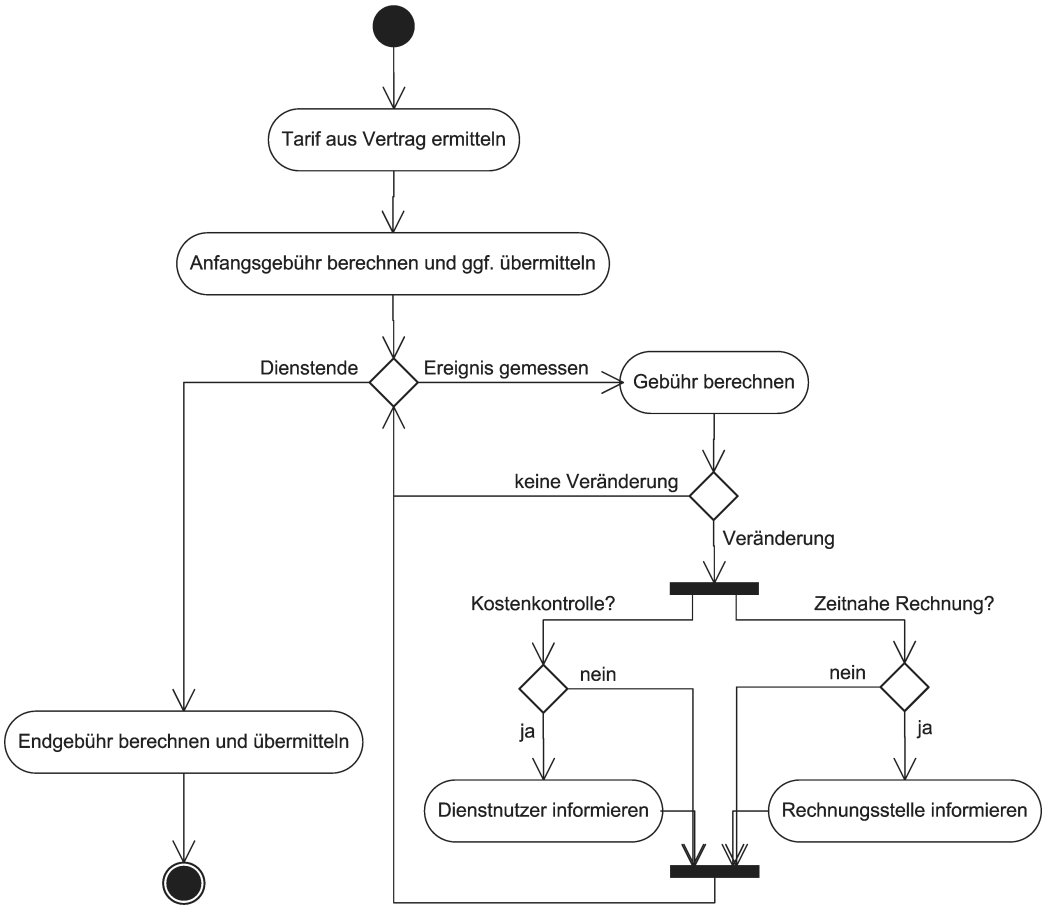

Abbildung 7.4: Aktivitätsdiagramm zum Prozess der Gebührenberechnung

Zu Beginn der Dienstnutzung wird aus dem Abrechnungsvertrag der gültige Tarif ermittelt und damit die Anfangsgebühr der Dienstnutzung berechnet. Sofern laut Vertrag eine Kostenkontrolle vom Benutzer gewünscht ist, wird der Dienstnutzeragent sofort über den aktuellen Stand der Nutzungsgebühr informiert (siehe Abschnitt 7.4). Parallel dazu wird auch der in Abschnitt 7.8 beschriebene Agent für die Rechnungsstellung informiert, wenn die vom Kunden angeforderten Rechnungen auch laufende Dienstnutzungen enthalten sollen.

Anschließend wird auf eine Nachricht von der Abrechnungskomponente gewartet, die das Ende der Dienstnutzung oder die Messung eines gebührenrelevanten Ereignisses mitteilt. Dadurch kann auf ein permanentes Abfragen verzichtet werden. Es reicht allerdings nicht immer aus nur über dienstinterne Ereignisse informiert zu werden, sondern es müssen auch zwei Spezialfälle berücksichtigt werden. Zum einen ist im Falle einer von der Nutzungsdauer abhängigen Tarifierung das Zeitintervall ausschlaggebend. Zum anderen ist bei Berücksichtigung von Subdienstnutzungen deren Gebührenänderung durch eine Kostenkontrolle zu überwachen. Nach jedem Ereignis wird die aktuelle Nutzungsgebühr erneut berechnet. Eine eventuelle Mitteilung an den Dienstnutzer oder Rechnungssteller erfolgt aber nur wenn sich die Gebühr geändert hat. Nach Ende der Dienstnutzung wird die endgültige Gebühr berechnet und in jedem Fall an die Rechnungsstelle übermittelt, bevor der Prozess beendet wird.

7.4 Kostenkontrolle

Die Kostenkontrolle ermöglicht einem Dienstnutzer schon während der Dienstnutzung über den aktuellen Gebührenstand informiert zu werden. Neben der bereits in Abschnitt 7.3 erwähnten Kostenkontrolle, die ausschließlich für den dienstnutzenden Agenten durch den Anbieteragenten angestoßen wird, stehen zusätzlich drei Dienste der Anbieteragenten zur Verfügung, die unter den folgenden Bedingungen die Gebühr einer angegebenen Dienstnutzung mitteilen:

- Auf einmalige Anfrage
- Bei jeder Änderung
- In angegebenen Zeitabständen

Bei der ersten Variante wird die aktuelle Nutzungsgebühr einmalig übertragen und die Kostenkontrolle damit sofort beendet. Bei den letzten beiden Diensten läuft jeweils ein Protokoll ab, das erst mit Ende der kontrollierten Dienstnutzung beendet wird.

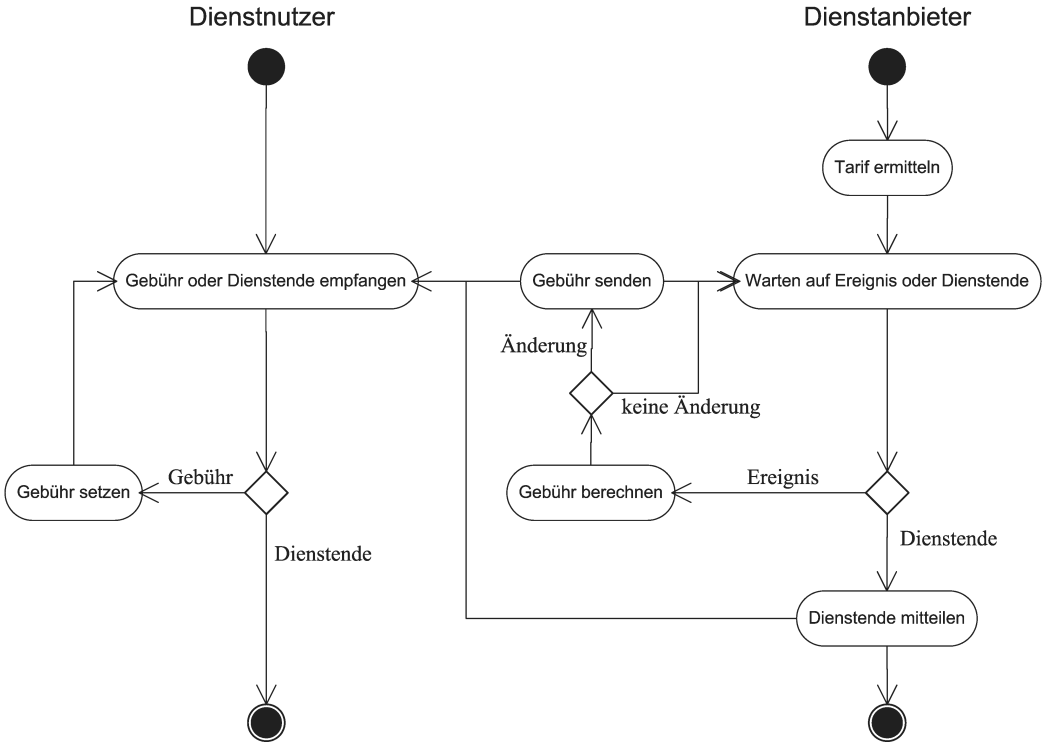

Abbildung 7.5: Protokoll der änderungsbasierten Kostenkontrolle

In Abbildung 7.5 ist das für die änderungsbasierte Kostenkontrolle implementierte Protokoll dargestellt. Ähnlich wie bei der durch den Anbieteragenten angestoßenen Kostenkontrolle müssen alle Ereignisse vom Dienstanbieter überwacht werden, die die Gebühr beeinflussen können. Nur bei Änderung wird die aktuell berechnete Gebühr übermittelt. Sobald die überwachte Dienstnutzung beendet ist, wird über das Ende der Kostenkontrolle informiert.

7.5 Tarifverwaltung

Damit die Abrechnung den Anforderungen an Dienstorientierung und Kundenindividualität genügt, müssen verschiedene Tarife mit unterschiedlichen Gebührenmodellen verwaltet werden können. Diese Gebührenmodelle sollen einerseits beliebige dienstspezifische Parameter und Ereignisse berücksichtigen können und andererseits möglichst einfach von den Dienstanbietern an deren Bedürfnisse anpassbar sein. Aus diesem Grund wurde eine Parametrisierungsmöglichkeit von Gebührenmodellen eingeführt, die es Tarifentwicklern erlaubt beliebige Kostenfunktionen zu implementieren, die neben den gemessenen Nutzungsdaten auch weitere Parameter berücksichtigen. Diese in einer Ontologie zu beschreibenden Parameter können dann von den Dienstanbietern instanziiert werden, um konkrete Tarife zu definieren, die anschließend in Dienstnutzungsverträgen Verwendung finden. Es wurde bereits ein Parametrisierungsschema definiert, mit dem z.B. lineare Kostenfunktionen implementiert werden können. Insgesamt ergibt sich daraus die in Abbildung 7.6 dargestellte Ontologie zur Definition von Tarifen.

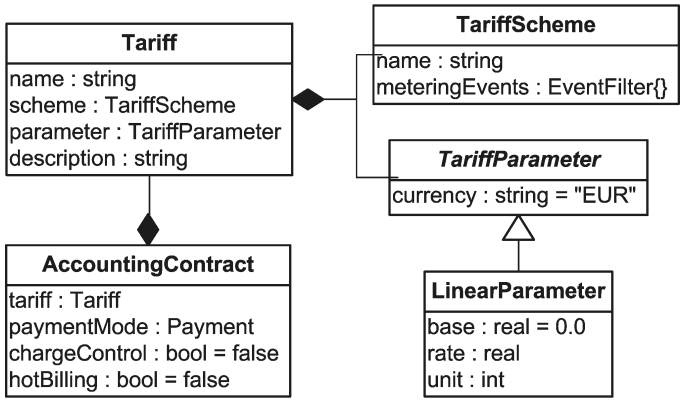

Abbildung 7.6: Ontologie zur Definition von Tarifen

Ein Gebührenmodell wird anhand des Namens des Tarifschemas und der Art der Parametrisierung eindeutig identifiziert. Die zu einem Gebührenmodell implementierte Kostenfunktion kann dabei auf die gemessenen Ereignisse zu den im Tarifschema definierten Ereignismustern (siehe Abschnitt 7.2) und auf das Parameterobjekt mit den vom Dienstanbieter angegebenen Werten zugreifen um die aktuelle Dienstnutzungsgebühr zu berechnen. Tarife bestehen demzufolge aus einem Tarifschema und einer Parametrisierung und werden gewöhnlich durch einen Tarifagenten verwaltet. Es muss nur sicher-

gestellt sein, dass die diensterbringenden Agenten zu allen gültigen Tarifen sowohl die zugehörige Kostenfunktion als auch die benötigten Vermittler- und Messkomponenten besitzen.

Zur Verwaltung der Tarife bietet der Tarifagent folgende vier Dienste an:

- *GetTariffs*: Liefert die Liste aller registrierten Tarife.
- *SearchOneTariff*: Sucht nach einem Tarif zum angegebenen Namen.
- *AddTariff*: Fügt einen neuen Tarif hinzu oder ersetzt einen bereits registrierten Tarif.
- *RemoveTariff*: Entfernt einen registrierten Tarif.

Jeder Tarifschemaanbieter sollte folgende zwei Dienste zur Verfügung stellen:

- *GetTariffSchemes*: Liefert Beschreibung, Tarif- und Parametrisierungsschema sowie Ontologien mit den verwendeten Ereignistypen zu allen angebotenen Tarifmodellen.
- *LoadTariffComponents*: Sucht nach der Kostenfunktion und den Vermittler- sowie Messkomponenten zum angegebenen Tarifmodell.

Am Beispiel der JIAC-Agentenarchitektur wurden die Kostenfunktionen als Skripte implementiert, um eine größtmögliche Flexibilität bei der Tarifierung zu gewährleisten. Diese Skripte müssen einen bereits vordefinierten Effekt besitzen, damit sie vom Abrechnungsprozess automatisch für die Gebührenberechnung ausgeführt werden können. Ein Skript zur Berechnung einer Dienstnutzungsgebühr, die von der Anzahl aller zu den Ereignismustern des Tarifschemas gesammelten Ereignisse abhängig ist, ist für den Fall eines linearen Parametrisierungsschemas in Abbildung 7.7 beispielhaft angegeben.

Neben diesem Tarifmodell sind noch zwei weitere Kostenfunktionen beispielhaft implementiert worden. Zum einen handelt es sich dabei um eine von der Nutzungsdauer und zum anderen um eine von den Gebühren aller Subdienstnutzungen abhängigen Tarifierung. Für die Ermittlung der Nutzungsdauer brauchen keine Ereignisse berücksichtigt werden, da Beginn und Ende jeder Dienstnutzung bereits in dem entsprechenden Dienstnutzungsobjekt angegeben sind. Für die Ermittlung aller zu einer Diensterbringung gehörenden Subdienstnutzungen wird der Ereignistyp „SubService“ verwendet. Die Abfrage der Gebühren zu jeder der gesammelten Subdienstnutzungen erfolgt mit Hilfe der in Abschnitt 7.4 beschriebenen Kostenkontrolle.

```
(act myCharging
   (var ?servInst:ServiceInst ?time:Date ?account:ServiceAccount ?date:Date
        ?tariff:Tariff ?scheme:TariffScheme ?parameter:LinearParameter
        ?currency:string ?base:real ?rate:real ?unit:int)

   // zusätzliche Vorbedingungen des Tarifmodells
   (pre (and
      (att TariffParameter.currency ?parameter ?currency)
      (att base ?parameter ?base)
      (att rate ?parameter ?rate)
      (att unit ?parameter ?unit)
   ))

   // nur Name des Tarifschemas und Parametertyp anpassen
   (eff (and
      (att account ?servInst ?account)
      (att name ?scheme \"myTariffScheme\")
      (att parameter ?tariff ?parameter)
      (att lastUpdate ?account ?date)
      (comp lessEquDate ?time ?date)
   ))

   (script
      (var ?chargedEvents:class:java.util.HashMap ?conv:string ?amount:real ?now:Date
           ?number:int ?filters:EventFilter{} ?filtername:string ?money:Money)
      (seq
         // Aktualisierungsdatum der Dienstnutzungsgebühr setzen
         (bind ?now (fun currentDate))
         (update (att lastUpdate ?account ?now))

         // gesammelte gebührenrelevante Ereignisse abrufen
         (getAccount (var ?servInst ?chargedEvents ?conv))

         // Ereignisse zu allen Ereignismustern zusammenzählen
         (bind ?number 0)
         (eval (att meteringEvents ?scheme ?filters))
         (iseq ?filters (var ?filter:EventFilter)
            (seq
               (eval (att name ?filter ?filtername))
               (bind ?number (fun Int.add ?number (fun getLength
                  (fun getCharges ?chargedEvents ?filtername))))
               (unbind ?filtername)
            )
         )

         // Dienstnutzungsgebühr berechnen
         (bind ?amount (fun Real.add ?base (fun Real.mul ?rate
            (fun int2real (fun Int.div ?number ?unit)))))

         // Dienstnutzungsgebühr aktualisieren
         (bind ?money (obj Money (amount ?amount) (Money.currency ?currency)))
         (update (att price ?account ?money))
      )
   )
)
```

Abbildung 7.7: Beispiel für die Implementierung einer linearen Kostenfunktion

7.6 Benutzerverwaltung

Für eine kundenindividuelle Abrechnung von nicht-anonymisierten Dienstnutzungen ist eine Benutzerverwaltung durch den Dienstanbieter unerlässlich. Hierbei werden Informationen über die Kunden (z.B. die in den Verträgen enthaltenen Daten wie Dienste und Tarife sowie Anschrift und sonstige Daten für die Rechnungsstellung) als auch über deren Nutzer verwaltet. Zusätzlich können vom Anbieter auch alle kostenpflichtigen Dienste mit den jeweils möglichen Tarifen eingetragen werden, um sie von den kostenlosen Diensten unterscheiden zu können. Dabei sollten auch Abweichungen zwischen einzelnen Vertretern eines Anbieters möglich sein.

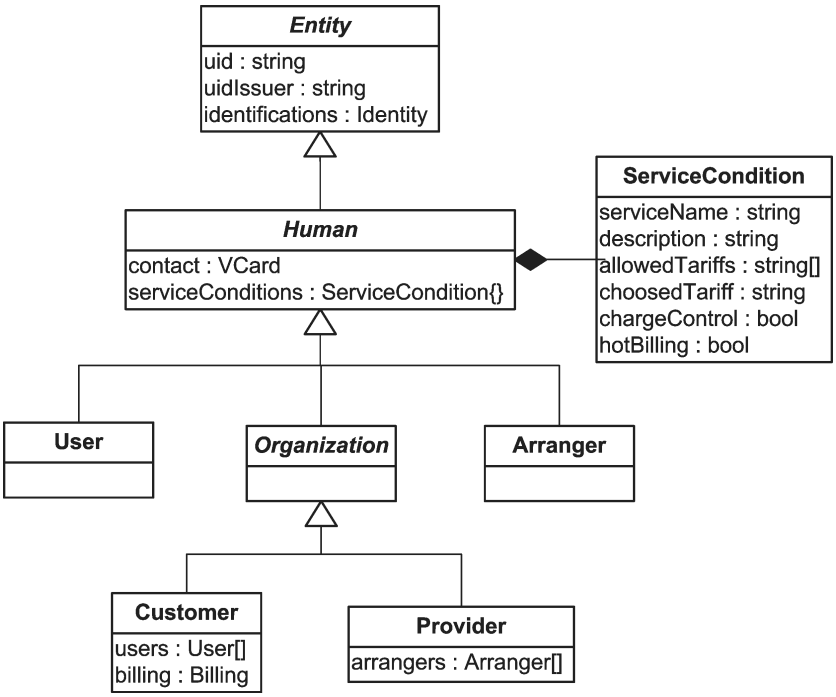

Abbildung 7.8: Ontologie zur Beschreibung von Kunden, Benutzern und Anbietern

Abbildung 7.8 zeigt die für die Benutzerverwaltung entworfene Datenstruktur, die auf der in Abschnitt 5.4.1 beschriebenen Basisontologie für die Verwaltung beliebiger organisatorischer Einheiten aufsetzt. Bei Personen (Human) kann es sich sowohl um Kunden (Customer) oder Anbieter (Provider) in Gestalt einer Organisation (Organization) als auch um einzelne Nutzer (User) oder Vertreter (Arranger) handeln. Alle besitzen Kontaktinformationen (con-

tact) in Form einer „vCard“ [DH98] und abrechnungsspezifische Informationen zu den jeweils relevanten Diensten (serviceConditions). Jedem Anbieter ist zusätzlich eine Liste von Vertretern (arrangers) und jedem Kunden eine Liste von Benutzern (users) sowie die Art der Rechnungsstellung (billing) zugeordnet. Zu den abrechnungsspezifischen Informationen eines nutzbaren oder angebotenen Dienstes gehören der Dienstname (serviceName), eine kurze Dienstbeschreibung (description), die verfügbaren Tarife (allowedTariffs), der standardmäßige oder ausgewählte Tarif (choosedTarif) und der Wunsch oder das Angebot einer Kostenkontrolle (chargeControl) sowie einer Rechnungsstellung mit Angabe laufender Dienstnutzungen (hotBilling). Dabei ist zu beachten, dass einem Benutzer oder Vertreter immer nur eine Teilmenge der für den zugehörigen Kunden bzw. Anbieter angegebenen Dienste und Tarife zugeordnet ist.

Damit die Kunden, Benutzer, Anbieter und Vertreter ihre Daten administrieren und der Abrechnungsprozess diese Informationen abfragen kann, muss der Benutzerverwaltungsagent entsprechende Dienste bereitstellen. Über geeignete Sicherheitsmechanismen ist festzulegen, wer welche Daten einsehen und gegebenenfalls ändern darf. So sollten Benutzerdaten nicht nur vom Anbieter sondern auch vom zugehörigen Kunden selbst angepasst werden können. Sämtliche Änderungen müssen zuvor auf Zulässigkeit geprüft werden. Zu den Diensten der Benutzerverwaltung gehören im Einzelnen:

- *GetCustomers*: Gibt die Liste aller Kunden zurück.
- *SearchOneCustomer*: Liefert die Eigenschaften des angegebenen Kunden.
- *AddCustomer*: Fügt einen neuen Kunden mit den angegebenen Eigenschaften hinzu.
- *ChangeCustomerPermissions*: Erlaubt dem Anbieter die Liste der nutzbaren Dienste und erlaubten Tarife eines Kunden zu ändern.
- *ChangeCustomerSettings*: Erlaubt einem Kunden seine Kontaktinformationen zu ändern.
- *SearchOneUser*: Liefert die Eigenschaften des angegebenen Benutzers.
- *AddUser*: Fügt dem Kunden einen neuen Benutzer mit den angegebenen Eigenschaften hinzu.
- *ChangeUserPermissions*: Erlaubt einem Kunden die Liste der nutzbaren Dienste und erlaubten Tarife für einen seiner Benutzer zu ändern.

- *ChangeUserSettings*: Erlaubt einem Benutzer seine Kontaktinformationen zu ändern oder für seine Dienste andere Tarife auszuwählen.
- *RemoveUser*: Ermöglicht einem Kunden einen seiner Benutzer zu löschen.
- *RemoveCustomerAndUsers*: Entfernt den angegebenen Kunden und alle seine Benutzer.
- *GetProvider*: Gibt die Eigenschaften (z.B. kostenpflichtige Dienste mit den zulässigen Tarifen) des Anbieters zurück.
- *ChangeProvider*: Ändert die Eigenschaften des Anbieters.
- *GetArrangers*: Gibt die Liste aller Vertreter an.
- *SearchOneArranger*: Liefert die Eigenschaften des angegebenen Vertreters.
- *AddArranger*: Fügt einen neuen Vertreter mit den angegebenen Eigenschaften hinzu.
- *ChangeArrangerPermissions*: Erlaubt dem Anbieter die Liste der in seinem Namen maximal anzubietenden Dienste und zugehörigen Tarife eines Vertreters zu ändern.
- *ChangeArrangerSettings*: Erlaubt einem Vertreter seine Kontaktinformationen und die Liste der angebotenen Dienste und unterstützten Tarife zu ändern.
- *RemoveArranger*: Entfernt den angegebenen Vertreter.
- *SearchUserTariff*: Liefert zum angegebenen Benutzer und Dienst den zugehörigen Kunden und eingestellten Tarif.
- *SearchArrangerTariffs*: Liefert zum angegebenen Vertreter und Dienst den Anbieter und die unterstützten Tarife.

Für die am Beispiel von JIAC implementierte Benutzerverwaltung wurde ein Werkzeug entwickelt, das die Definition von Tarifen, kostenpflichtigen Diensten, Kunden, Nutzern, Anbietern und Vertretern unter Berücksichtigung der oben genannten Restriktionen erlaubt und daraus entsprechende Datenobjekte generiert. Diese Objekte können dann später dem Benutzerverwaltungsagenten hinzugefügt werden. Abbildung 7.9 zeigt einen Ausschnitt aus

der Benutzerschnittstelle dieses Werkzeuges, in dem die Definition eines Kunden „DAI-Labor“ mit dem Dienst „TravelService“ und zugeordnetem Tarif „Standard“ zu sehen ist.

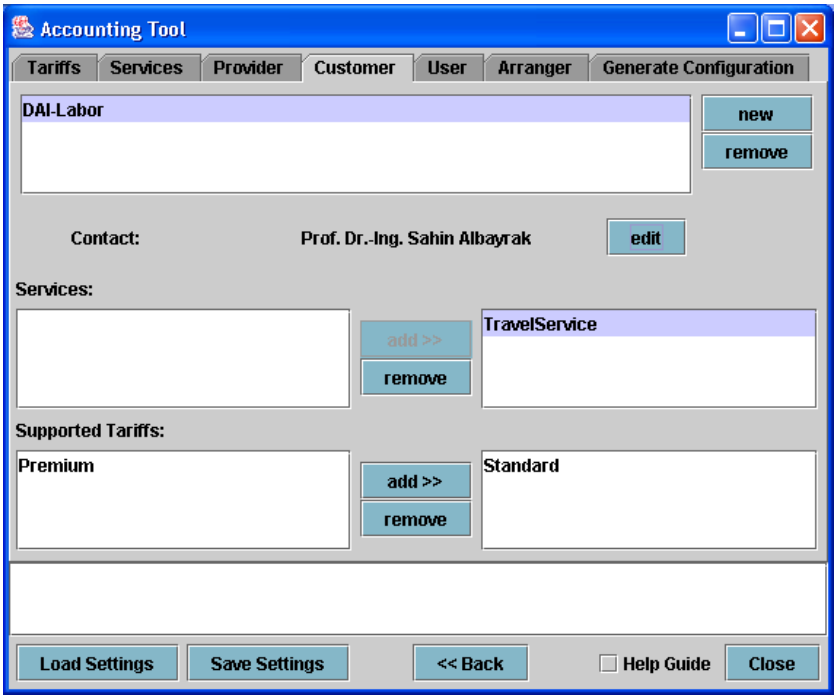

Abbildung 7.9: Benutzerschnittstelle eines Werkzeuges zur Definition von Kundenbeziehungen

7.7 Integration in den Dienstablauf

Damit der Abrechnungsprozess für jede Dienstnutzung automatisch ausgeführt wird, muss er von einer dienstübergreifenden Instanz gestartet und auch wieder beendet werden. Am Beispiel der JIAC-Architektur musste hierfür die Kommunikationskomponente erweitert werden, da sie für den Ablauf des Metaprotokolls zuständig ist.

Abbildung 7.10 zeigt den kompletten Ablauf des Abrechnungsprozesses während einer Dienstnutzung. Nach erfolgreicher Aushandlung der Dienstnutzung durch das Metaprotokoll sendet die Kommunikationskomponente des Anbieteragenten eine Nachricht an die Abrechnungskomponente (sofern vor-

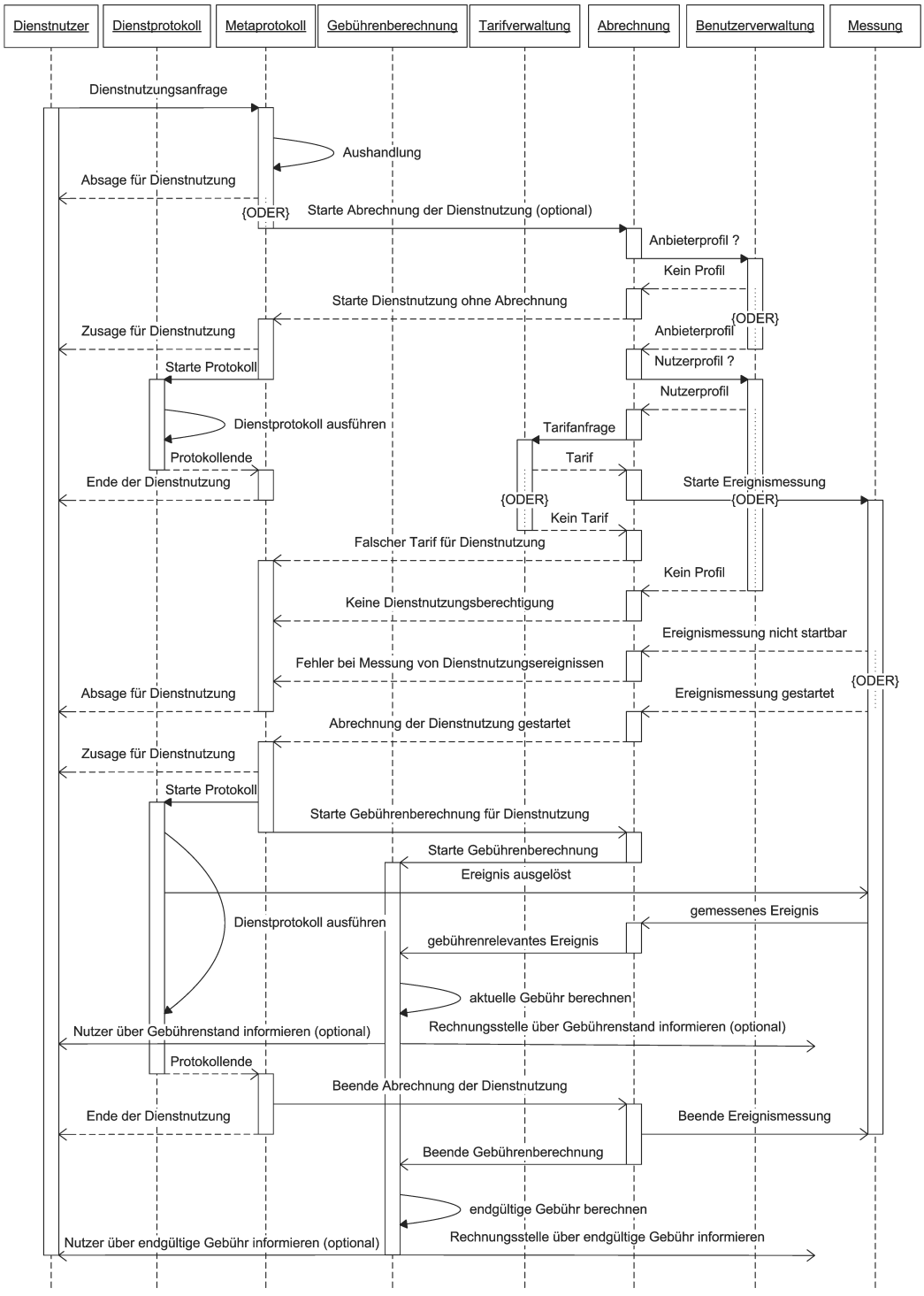

Abbildung 7.10: Sequenzdiagramm für den Ablauf des Abrechnungsprozesses innerhalb einer Dienstnutzung

handen), die daraufhin die Ausführung des Skriptes zum Starten der Ereignismessung anstößt. Innerhalb dieses Skriptes wird zunächst geprüft, ob es sich um eine kostenpflichtige Dienstnutzung handelt, indem beim Benutzerverwaltungsagent nach einem entsprechenden Dienstprofil des Anbieters bzw. Vertreters[2] gesucht wird (siehe Abschnitt 7.6). Ist kein Profil für diesen Dienst vorhanden, so endet das Skript und es findet die Dienstnutzung ohne Abrechnung statt. Anderenfalls wird zusätzlich noch das Dienstprofil des Benutzers[3] abgefragt, dieses Profil mit dem des Vertreters verglichen und für den im Profil angegebenen Tarifnamen eine Anfrage an den Tarifverwaltungsagenten gestellt (siehe Abschnitt 7.5). Kann für den zu nutzenden Dienst kein passendes Dienstprofil des Benutzers ermittelt werden oder erfolgt keine Unterstützung der im Profil angegebenen Einstellungen durch den Vertreter, so scheitert das Skript und die fehlende Berechtigung des Nutzers wird der Kommunikationskomponente mitgeteilt, die anschließend die Dienstnutzung ablehnt. Gleiches gilt, wenn der angegebene Tarif nicht bekannt ist. Im Erfolgsfall wird für jedes im Tarifschema angegebene Ereignismuster das Sammeln der zugehörigen Ereignisse durch Senden einer Nachricht an die angegebene Vermittlerkomponente gestartet, die wiederum entsprechende Messkomponenten aktivieren kann (siehe Abschnitt 7.2). Wurde die Überwachung für alle Ereignismuster erfolgreich angestoßen, so wird die Kommunikationskomponente über das Ende des Skriptes informiert, die daraufhin die Dienstnutzung akzeptiert, das dienstspezifische Protokoll startet und die Abrechnungskomponente zum Starten eines Skriptes für die ereignisgesteuerte Gebührenberechnung auffordert (siehe Abschnitt 7.3).

Sämtliche während des Dienstprotokolls ausgelösten und durch die Messkomponenten festgestellten Ereignisse werden an die zugehörige Vermittlerkomponente weitergeleitet. Die Vermittlerkomponenten prüfen alle gesammelten Ereignisse auf Relevanz für die Dienstnutzung und leiten sie gegebenenfalls mit einer Nachricht an die Abrechnungskomponente weiter, die daraufhin das Skript für die Gebührenberechnung informiert. Dieses Skript teilt auf Wunsch die aktuelle Nutzungsgebühr dem dienstnutzenden und dem rechnungsstellenden Agenten mit, sofern eine Änderung vorliegt.

Mit dem Ende des Dienstprotokolls sendet die Kommunikationskomponente eine Nachricht an die Abrechnungskomponente, die daraufhin die Überwachung für alle Ereignismuster durch Senden einer Nachricht an die entsprechenden Vermittlerkomponenten beendet. Diese Vermittlerkomponenten

[2]Der Vertreter ist der Eigentümer des dienstanbietenden Agenten.

[3]Der Benutzer ist der Verursacher der Dienstnutzung und somit der erste Eintrag in der Dienstnutzungskette, der entweder über die Eigentümerbeziehung oder durch Authentifizierungsmechanismen ermittelt wurde.

informieren ihrerseits alle beteiligten Messkomponenten. Anschließend wird das Ende der Dienstnutzung auch dem Skript für die Gebührenberechnung mitgeteilt, das die endgültige Nutzungsgebühr an den rechnungsstellenden Agenten überträgt (siehe Abschnitt 7.8) bevor es sich selbständig beendet.

7.8 Rechnungsstellung

Für eine automatisierte Rechnungsstellung muss bei jeder Registrierung eines neuen Kunden die Überwachung der im Vertrag angegebenen rechnungsbedingenden Ereignisse beim rechnungsstellenden Agenten angestoßen werden. Bei diesen Ereignissen kann es sich beispielsweise um eine Zeitspanne (z.B. ein Monat), um eine Anzahl von Dienstnutzungen oder um eine Mindestgebühr handeln. Bei Eintreten eines der angegebenen Ereignisse werden dann sofort sämtliche zum Kunden gehörenden Rechnungsdaten ermittelt. Der Gesamtbetrag wird über eine dem Tarif entsprechende Kostenfunktion berechnet, die gewöhnlich auf den seit der letzten Rechnungsstellung angefallenen Dienstnutzungs- bzw. Gebührendaten basiert. Diese Daten wurden von den diensterbringenden Agenten spätestens nach jeder Dienstnutzung mitgeteilt (siehe Abschnitt 7.3). Mit Deregistrierung eines Kunden wird die Überwachung der Ereignisse beendet und eine letzte Rechnung gestellt. Jede Rechnung muss dabei dem im Vertrag angegebenen Detaillierungsgrad genügen und auf dem angegebenen Weg (Post, Fax, E-Mail oder SMS) zugestellt werden.

Zusätzlich zur automatischen Rechnungsstellung wird ein Dienst angeboten, der es einem Kunden, Benutzer, Vertreter oder Anbieter ermöglicht eine Aufstellung aller durch ihn genutzten oder erbrachten Dienste zu bekommen. In dieser Aufstellung sind neben dem Auftraggeber und der aktuellen Uhrzeit zu jeder Dienstnutzung das Datum und die Uhrzeit, der Dienstname, der Tarif und die Gebühr, gegebenenfalls der Anbieter, Vertreter, Kunde oder Benutzer und die Gesamtgebühr über alle Dienstnutzungen angegeben. Abbildung 7.11 zeigt die geräteunabhängige Benutzerschnittstelle dieses Dienstes.

Der rechnungsstellende Agent bietet also folgende zwei Dienste an:

- *GetCharges*: Erlaubt einem Kunden, Benutzer, Anbieter oder Vertreter eine ausführliche Liste aller von ihm genutzten oder erbrachten Dienste anzufordern.

- *UpdateCharge*: Ermöglicht einem diensterbringenden Agenten den aktuellen Gebührenstand und weitere Informationen zu einer Dienstnutzung mitzuteilen.

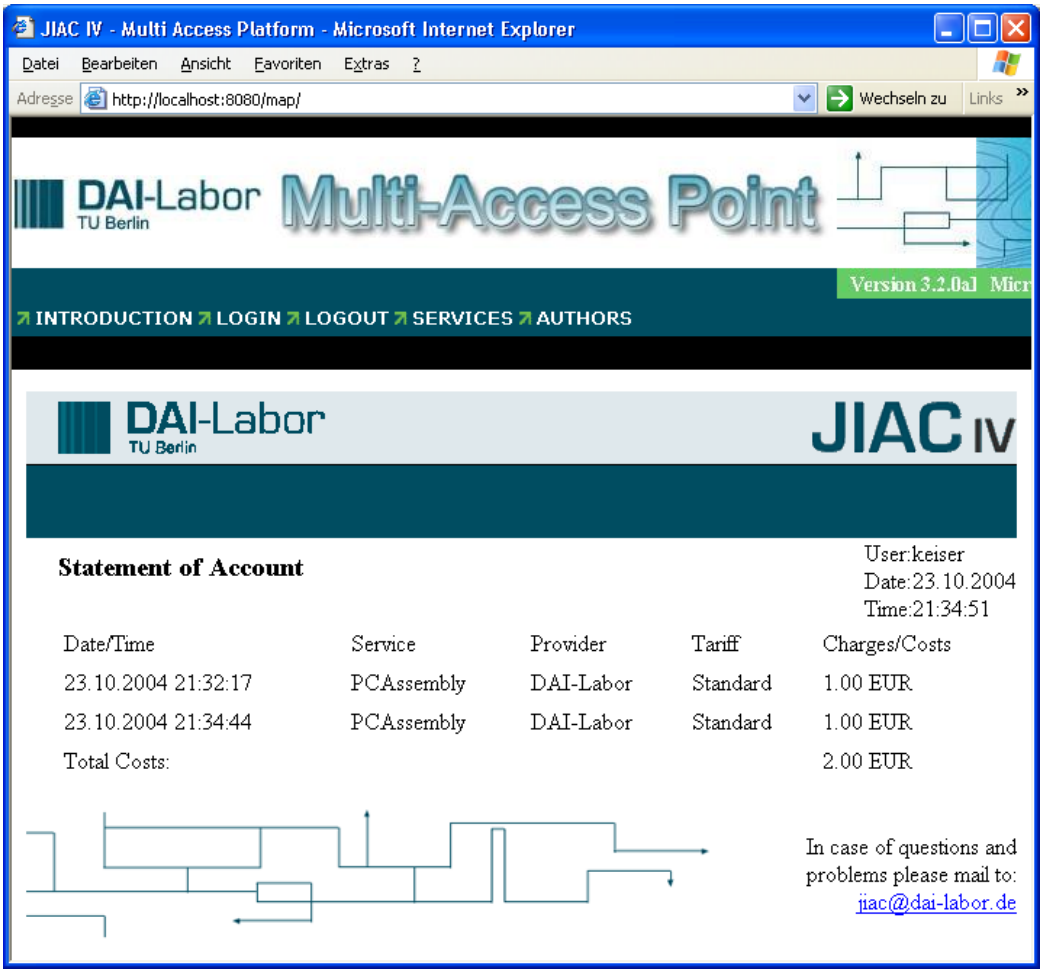

Abbildung 7.11: Benutzerschnittstelle für die Rechnungsstellung

Wünschenswert wäre auch, dass bei der Rechnungsstellung das im Vertrag angegebene Zahlungsverfahren Berücksichtigung findet. In Abhängigkeit vom Zahlungsverfahren könnte ein entsprechender Inkasso-Mechanismus realisiert werden. Die Einbindung bestehender Zahlungssysteme war jedoch nicht Bestandteil dieser Arbeit.

7.9 Zusammenfassung

In diesem Kapitel wurde basierend auf den Basismechanismen der Managementinfrastruktur ein Abrechnungsmechanismus entwickelt, der den Anforderungen an Dienstorientierung, Kundenindividualität, Änderungsdynamik und Integrationsfähigkeit gerecht wird. Dienstorientierung meint, dass potentiell jede agentenbasierte Dienstnutzung abgerechnet werden kann. Die Tarife können jederzeit individuell an die Gegebenheiten jedes einzelnen Benutzers oder Kunden angepasst werden. Die Berücksichtigung beliebiger messbarer Ereignisse (z.B. auch Subdienstnutzungen) ist bei der Abrechnung von Diensten möglich. Dieser Abrechnungsmechanismus lässt sich unter Verwendung der Basismechanismen prinzipell in jede dienstbasierte Agentenarchitektur integrieren ohne die Architektur oder die Implementierung der abzurechnenden Dienste ändern zu müssen.

Ausgehend von einem Grobkonzept, das die am Abrechnungsprozess beteiligten Rollen definiert, wurden anschließend die Messung und Sammlung von Dienstnutzungsdaten, die Berechnung der Gebühren einer Dienstnutzung, die Kostenkontrolle durch den Dienstnutzer, die Verwaltung der Tarife und Benutzer sowie die Rechnungsstellung genauer beschrieben. Bei der Messung der Dienstnutzungsdaten wird die Introspektionsfähigkeit der diensterbringenden Agenten genutzt, um die in den Tarifmodellen spezifizierten gebührenrelevanten Ereignisse zu überwachen. Damit die Tarife vom Dienstanbieter einfach definiert werden können, bestehen sie aus einem konfigurierbaren Tarifmodell, das von einem Tarifentwickler bereitgestellt wird, und einer zugehörigen Konfiguration. Einem Tarifmodell sind sowohl die gebührenrelevanten Ereignismuster als auch die Implementierung einer Kostenfunktion basierend auf den gemessenen Daten und der Konfiguration zugeordnet. In der Benutzerverwaltung sind für jeden Kunden unter anderem die Vertragsdaten und für dessen Benutzer auch die erlaubten und der gewählte Tarif angegeben, die in der Tarifverwaltung abgelegt sind. Entsprechende Werkzeuge unterstützen die verschiedenen Rollen bei der Definition der Tarife, bei der Verwaltung der Kunden und Benutzer sowie bei der Abfrage der Rechnungsdaten.

Das Abrechnungsmanagement wurde prototypisch mit der Agentenentwicklungsumgebung JIAC IV implementiert. Das bedeutet, dass alle Datenstrukturen als Ontologien modelliert wurden und für die Kostenkontrolle, Benutzer- und Tarifverwaltung wie auch die Rechnungsstellung entsprechende Managementdienste spezifiziert wurden, die durch verschiedene Managementagenten bereitgestellt werden.

Kapitel 8
Konfigurationsmanagement

Das Konfigurationsmanagement agentenbasierter Anwendungen umfasst alle Aktivitäten, die zur Erhebung, Inventarisierung und Kontrolle sowie zur Initialisierung, Änderung und Anpassung relevanter Systemelemente wie Plattformen, Agenten und Dienste notwendig sind. Diese Elemente müssen dabei eindeutig identifizierbar sein. Die Konfigurationen vor und während des Betriebs einer Anwendung unterscheiden sich nicht wesentlich voneinander. Bei Wartungsarbeiten müssen nur zusätzlich die eventuell laufenden Prozesse berücksichtigt und erst in einen konsistenten Zustand überführt werden. Unter Umständen ist es auch notwendig zu definieren, welche Personen welche Änderungen vornehmen dürfen. Im Folgenden erfolgt ausschließlich eine Unterscheidung zwischen der Konfiguration

- eines einzelnen Agenten sowie
- einer kompletten Anwendung inkl. der Dienste.

Zur Agentenkonfiguration gehört die Zuordung der innerhalb einer Anwendung zu spielenden Rollen und den damit verbundenen Fähigkeiten auf einzelne Agenten oder Gruppen von Agenten. Die Fähigkeiten und Verhaltensweisen einer Rolle werden durch die zugehörigen Bestandteile (z.B. Wissenselemente, Komponenten, Skripte) und eventuell notwendiger externer Software (z.B. Datenbanken) bestimmt, die den Agenten als Ganzes oder einzeln über die in Abschnitt 6.3 beschriebene Manipulationsfunktion hinzugefügt und entfernt werden können (siehe Abbildung 8.1). Es werden nur die statischen Elemente, die den Aufbau eines Agenten beschreiben, und keine kurzfristigen Ziele oder Intentionen betrachtet. Bei der Konfiguration müssen die zwischen den Rollen und Elementen bestehenden Abhängigkeiten und Konflikte berücksichtigt und aufgelöst werden.

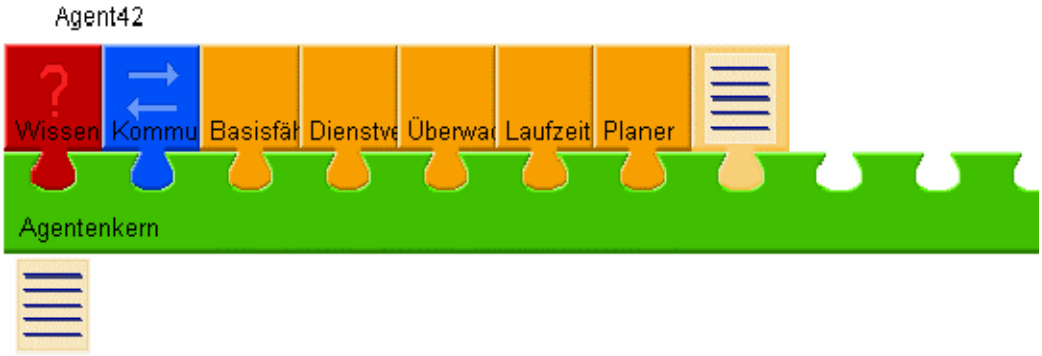

Abbildung 8.1: Konfiguration eines Agenten

Zur Konfiguration einer agentenbasierten Anwendung gehört die Zuordnung neuer oder vorhandener Agenten auf einzelne Plattformen, die die Laufzeitumgebung der Agenten darstellen und sich auf unterschiedlichen Rechnern befinden können. Hierfür stehen verschiedene Funktionen der in Abschnitt 6.4 beschriebenen Agenteninfrastruktur zur Verfügung. Es muss betrachtet werden, welche Agenten zwingend zur Diensterbringung erforderlich sind, welche Rollen sie erfüllen müssen und wie sie über die Plattformen verteilt werden sollen. Zur Installation eines neuen Dienstes kann es notwendig sein, nicht nur neue Agenten zu erzeugen sondern auch im operativen Betrieb stehende Agenten mit zusätzlichen Eigenschaften auszurüsten. Auch bei der Konfiguration ganzer Anwendungen müssen die Abhängigkeiten zwischen den Rollen und Agenten berücksichtigt werden. Zur Vereinfachung der Konfiguration oder aus organisatorischen Gründen können Agenten zu Gruppen zusammengefasst werden. Eine Plattform stellt dabei ein Spezialfall einer Gruppe dar. Zur Vervielfältigung und Verteilung von Fähigkeiten steht das Konzept des Klonens zur Verfügung, bei dem Agenten mit gleichen Fähigkeiten erzeugt werden. Es handelt sich dabei nicht um identische Agenten.

Die Unterscheidung zwischen der Plattform- und der Agentenkonfiguration führt ähnlich wie beim allgemeinen Managementmodell zu einem agenteninternen Konfigurationsanteil und einem übergeordneten Konfigurationsmanager. Hierzu wird im nächsten Abschnitt das Konzept für das Konfigurationsmanagement grob beschrieben, bevor in den nachfolgenden Abschnitten konkret auf die einzelnen identifizierten Aspekte eingegangen wird.

8.1 Grobkonzept

Das in Abbildung 8.2 dargestellte Grobkonzept zeigt die an der Konfiguration beteiligten Rollen und deren Aufgaben sowie die zwischen den Rollen ablaufenden Interaktionen.

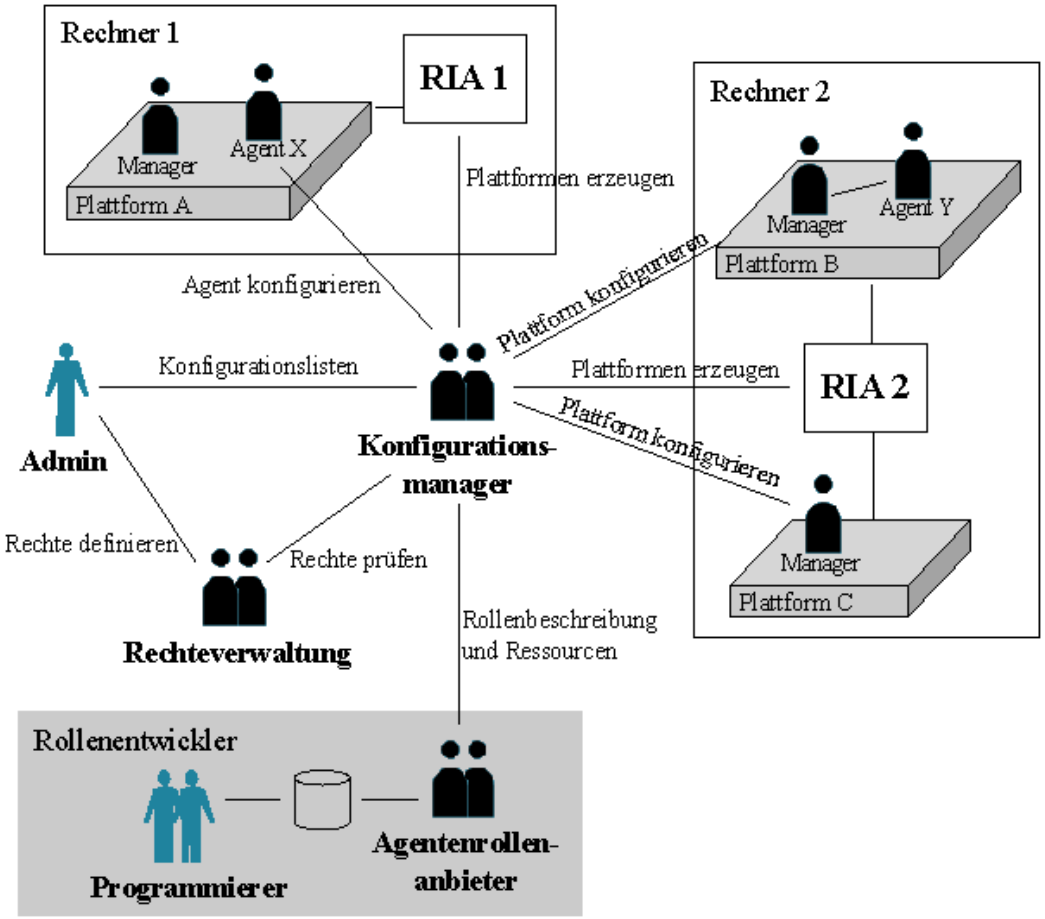

Abbildung 8.2: Rollen und Interaktionen beim Konfigurationsmanagement

Der Administrator eines agentenbasierten Systems fordert vom Konfigurationsmanager die aktuellen Konfigurationslisten an, deren Struktur in Abschnitt 8.2 ausführlich beschrieben ist. Für die Konstruktion[1] können bereits bestehende Listen geladen oder auf Anfangslisten zugegriffen werden. In der

[1]Unter Konstruktion ist die Definition bzw. Zusammenstellung von Agenten gemeint und nicht die Erzeugung von Laufzeitobjekten.

Wartungsphase, d.h. während der Laufzeit des Systems, werden diese Listen durch Abfrage der zugehörigen Plattformen und Agenten erstellt (siehe Abschnitt 8.4.1). Um eine Änderung der Konfiguration geeignet zu unterstützen, werden von verschiedenen Entwicklern unterschiedlichste Agentenrollen angeboten, die eine bestimmte Fähigkeit oder Verhaltensweise kapseln (siehe Abschnitt 8.3). Der Zugriff auf selbst implementierte Rollen ist dabei inbegriffen. Die veränderten Konfigurationslisten können nun für die Anpassung einer laufenden oder zum Starten einer neuen Anwendung verwendet werden. Für das Starten eines neuen Prozesses (beispielsweise einer Agentenplattform) auf einem entfernten Rechner muss dort ein in Abschnitt 8.5 beschriebener RIA-Server (Remote Invocation Architecture) zur Verfügung stehen, dem die benötigten Ressourcen und die entsprechende Konfiguration übergeben werden. Der Manageragent einer Plattform ist in der Lage seine Plattform und eventuell auch die dort existierenden Agenten zu konfigurieren (siehe Abschnitt 8.4.2). Alternativ dazu kann auch die Konfigurationsmöglichkeit der einzelnen Agenten direkt genutzt werden. Um sicherzustellen, dass nur ausgewählte Personen Zugriff auf die Konfiguration haben, können sowohl der Konfigurationsmanager als auch die Agenten selbst auf Informationen der in Abschnitt 8.6 beschriebenen Rechteverwaltung zugreifen. Über diese Rechteverwaltung ist es dem Administrator möglich zu definieren, welche Personen von welchen Agenten oder Gruppen welche Daten auslesen und welche Änderungen vornehmen dürfen.

8.2 Konfigurationslisten

Konfigurationslisten ermöglichen die Konstruktion und darauf basierend die anschließende Erzeugung oder Veränderung einer agentenbasierten Anwendung. Sie beschreiben den Aufbau eines solchen Systems inklusive der zugehörigen Plattformen und Agenten mit ihren (statischen) Bestandteilen und technischen Werten. Dabei werden einzelne Merkmale, Fähigkeiten und Verhaltensweisen der Anwendung unter Berücksichtigung vorhandener Abhängigkeiten und Einschränkungen verschiedenen Agenten zugeordnet. Eine Konfiguration besteht in der Regel sowohl aus allgemeingültigen (z.B. FIPA-konformen) Angaben als auch aus Anteilen, die von der verwendeten Laufzeitumgebung bzw. Agentenarchitektur abhängig sind. Abbildung 8.3 zeigt eine Ontologie, die für den architekturunabhängigen Teil der Konfigurationslisten definiert wurde.

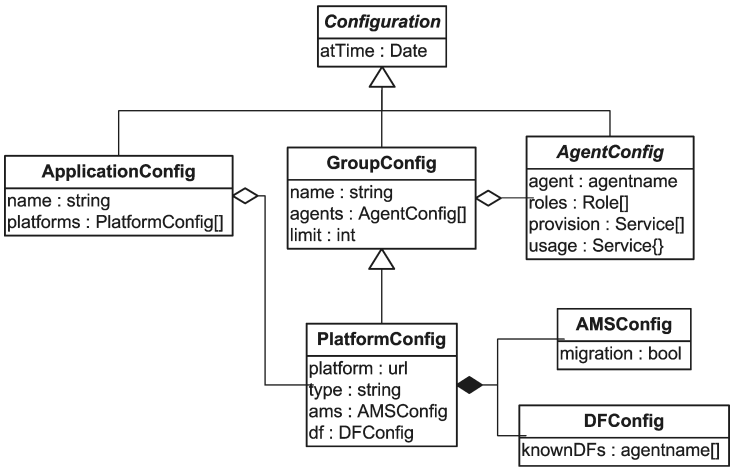

Abbildung 8.3: Ontologie zur Konfiguration agentenbasierter Systeme

Es können gesamte Anwendungen aber auch Plattformen und einzelne Agenten konfiguriert werden. Zu jeder Konfiguration (Configuration) ist angegeben, wann sie laufzeittechnisch umgesetzt oder gemäß einer laufenden Anwendung aktualisiert wurde (atTime). Eine agentenbasierte Anwendung (ApplicationConfig) besteht aus einer Liste von Agentenplattformen (platforms). Eine Agentenplattform (PlatformConfig) ist eine spezielle Gruppe von Agenten (GroupConfig), die zusätzlich eine eindeutige Adresse (platform), eine Bezeichnung der verwendeten Laufzeitumgebung (type) sowie eine Beschreibung des AMS und DF besitzt. Auf einer Plattform können nur Agenten existieren, die zur angegebenen Laufzeitumgebung (z.B. JIAC IV) kompatibel sind. Die Konfiguration des AMS (AMSConfig) und des DF (DFConfig) geben Aufschluss darüber, ob eine Migration von Agenten möglich ist (migration) und mit welchen anderen DFs die Registrierungsdaten geteilt oder ausgetauscht werden (knownDFs). Zu jeder Gruppe kann neben dem Namen (name) und einer Agentenliste (agents) auch eine maximale Anzahl von Agenten (limit) angegeben werden. Ein Agent (AgentConfig) hat einen eindeutigen Namen (agent), spielt eine Menge von Rollen (roles) und erbringt (provision) bzw. nutzt (usage) in diesem Zusammenhang bestimmte Dienste. Die Definition der Dienstbeschreibung erfolgte bereits in Abschnitt 5.4.2 und der Aufbau der Agentenrollen wird in Abschnitt 8.3 beschrieben. Aufgrund dieser Angaben kann schon während der Konfiguration automatisch geprüft werden, ob zu jedem durch einen Agenten genutzten Dienst innerhalb der entsprechenden Region ein zugehöriges Dienstangebot existiert.

Sämtliche Konfigurationen können durch Spezialisierungen der beschriebenen Ontologie auch architekturabhängige Bestandteile aufweisen. Abbildung 8.4

zeigt beispielsweise eine Ontologie für die Konfiguration eines JIAC-Agenten. Ein JIAC-Agent (JIACAgentConfig) besitzt zusätzlich zu den allgemeinen Angaben einer Agentenkonfiguration eine Menge von Berechtigungen des Manageragenten (permissions), Loggingeinstellungen (logConf), eine Performanzstufe (performance) und eine Liste von Komponenten (components), Planelementen (plans), Fakten (facts) und Regeln (rules). Weiterhin sind die möglichen Effekte der angebotenen Dienste (effect), die potentiellen Ziele (goals) und eventuell von den Komponenten benötigte externe Software (neededSoftware) angegeben. Auch hier kann überprüft werden, ob die externen Softwarebestandteile und zu den Zielen zugehörige Dienste innerhalb der Region existieren.

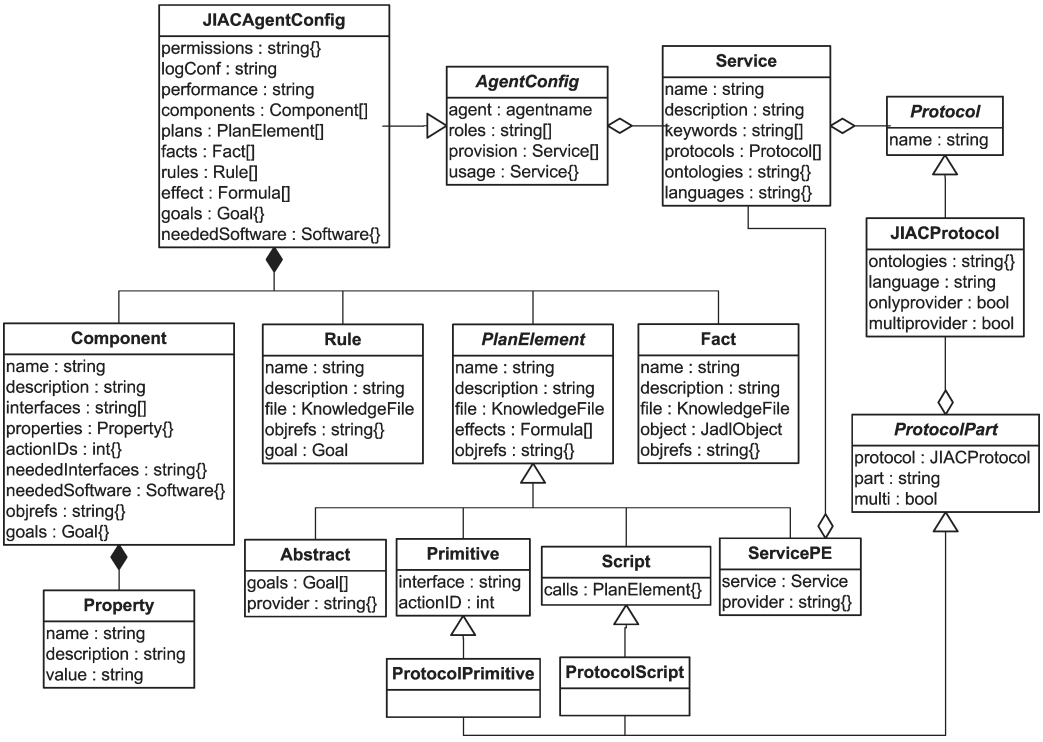

Abbildung 8.4: Ontologie zur Konfiguration von JIAC-Agenten

Eine Komponente (Component) bietet (interfaces) und nutzt (neededInterfaces) eine Menge von Schnittstellen, hat gewisse Einstellungen (properties), kann Aktionen ausführen (actionIDs), auf externe Software (neededSoftware) und Faktenbasisobjekte zugreifen (objrefs) sowie Ziele aufstellen (goals). Hierbei ist zu prüfen, ob andere Komponenten mit den zu nutzenden Schnittstellen im Agenten vorhanden sind und nicht die gleichen Schnittstellen verwenden. Regeln, Planelemente und Fakten können aus einer Datei geladen

werden (file) und auf Faktenbasisobjekte referenzieren (objrefs). Feuert eine Regel (Rule), so wird gewöhnlich ein Ziel aufgestellt (goal). Ein Fakt (Fact) wird durch ein Objekt einer beliebigen von „JadlObject“ abgeleiteten Kategorie repräsentiert (object). Ein Planelement (PlanElement) kann eine Menge an Effekten (effects) erreichen, wobei zwischen Dienstplanelementen, Skripte sowie abstrakte und primitive Planelemente unterschieden wird. Ein abstraktes Planelement (Abstract) stellt Ziele auf (goals), die durch einen der angegebenen Agenten (provider) zu erfüllen sind. Ein Skript (Script) hingegen ruft wiederum andere Planelemente auf (calls), ein Dienstplanelement (ServicePE) nutzt den angegebenen Dienst (service) bei einem der möglichen Agenten (provider) und ein primitives Planelement (Primitive) führt eine Aktion (actionID) bei einer Komponente mit angegebener Schnittstelle aus (interface). Das Konfigurationsmanagement muss dafür sorgen, dass dem Agenten die von den Skripten aufgerufenen Planelemente und für die primitiven Handlungen entsprechende Komponenten mit den notwendigen Schnittstellen und Aktionen zugeordnet sind. Zusätzlich ist zu prüfen, dass die in den Dienstplanelementen genannten Dienste und die in den abstrakten Handlungen definierten Effekte auch jeweils von einem der angegebenen Agenten angeboten bzw. erfüllt werden.

Da es sich bei JIAC-Protokollen (JIACProtocol) um eingebettete Protokolle mit internen Aktionen handelt, kann nur die Anbieterseite oder aber auch die Nutzerseite involviert sein (onlyprovider). Im zweiten Fall muss sichergestellt werden, dass neben dem dienstanbietenden auch der dienstnutzende Agent ein zugehöriges Protokollplanelement (ProtocolPart) besitzt. Bei Verhandlungsprotokollen sind gleichzeitig mehrere Instanzen von Anbietern möglich (multiprovider).

8.3 Verwaltung von Agentenrollen

Agentenrollen kapseln eine Menge von zusammengehörigen Fähigkeiten, Eigenschaften und Verhaltensweisen für Agenten um die Konfiguration überschaubarer und weniger fehleranfällig zu gestalten. Abbildung 8.5 zeigt eine Ontologie zur Definition solcher Agentenrollen. Eine Agentenrolle (Role) kann dabei aus mehreren Rollen aggregiert sein (subroles) und mit anderen Rollen über Dienste interagieren.

Ähnlich wie bei den Konfigurationslisten können auch den Agentenrollen architekturabhängige Bestandteile über entsprechende Spezialisierungen zu-

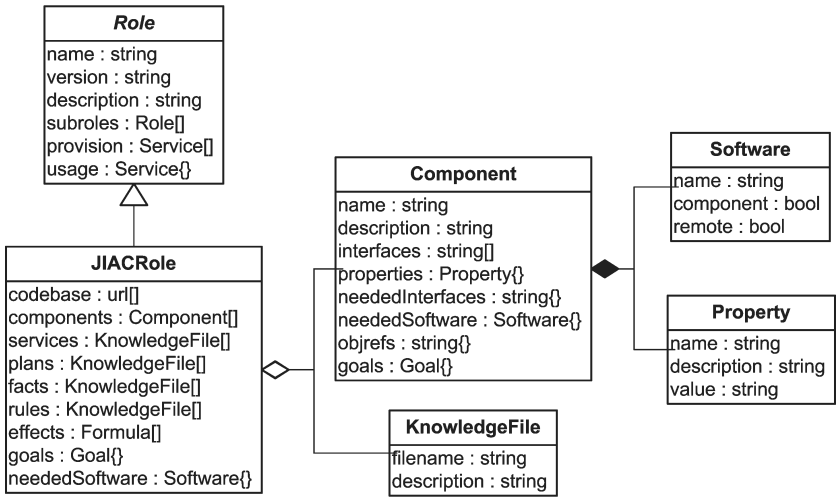

Abbildung 8.5: Ontologie zur Beschreibung von Agentenrollen

geordnet werden. Am Beispiel von JIAC kann zu einer Rolle (JIACRole) zusätzlich der verwendete Code in Form von URLs (codebase), eine Menge von Komponenten (components) sowie Dateien mit enthaltenen Diensten (services), Planelementen (plans), Fakten (facts) und Regeln (rules) angegeben werden. Eine Agentenrolle in JIAC erzielt üblicherweise eine Reihe an Effekten (effects) und muss eventuell dafür bestimmte Ziele erfüllt bekommen (goals) oder auf externe Software zugreifen können (neededSoftware). Externe Software (Software) kann durch eine JIAC-Komponente verwaltet (component) und eventuell auf einem entfernten Rechner gestartet werden (remote).

Um die Wiederverwendung implementierter Agentenrollen zu verbessern können vom Konfigurationsmanager verschiedene Dienste bei unterschiedlichen Rollenanbietern genutzt werden. Folgende Dienste stehen zur Verfügung:

- *SearchRoles*: Liefert eine Liste von Rollenbeschreibungen zu angegebenen Schlüsselworten oder Ontologien.
- *LoadArchives*: Liefert zu angegebenen Rollennamen und Agentenarchitekturen die kompletten Archive mit Code, Dokumentation und Rollenbeschreibungen.

Das Werkzeug zur Konfiguration neuer und laufender Agentensysteme enthält für den Administrator zusätzlich eine Benutzerschnittstelle mit der eine Suche nach Agentenrollen angestoßen und die Ergebnisse inklusive der Doku-

mentation angezeigt werden können. Die ausgewählten Rollen können anschließend einzelnen Agenten oder Gruppen hinzugefügt werden.

8.4 Konfigurationsdienste

Für die Ermittlung der aktuellen Konfigurationslisten und die Änderung von agentenbasierten Anwendungen stellt der Konfigurationsmanager zwei Dienste zur Verfügung, die in den folgenden beiden Abschnitten genauer beschrieben werden. Diese beiden Dienste greifen dabei auf folgende Basismechanismen zurück:

- Agentenintrospektion (siehe Abschnitt 6.2) zur Ermittlung agenteninterner Konfigurationen.
- Agentenmanipulation (siehe Abschnitt 6.3) zur Änderung agenteninterner Konfigurationen.
- Agenteninfrastruktur (siehe Abschnitt 6.4) zur Ermittlung und Änderung agentenübergreifender Konfigurationen.

Die Erstellung neuer Agentenplattformen ist mit den agentenbasierten Mechanismen nicht möglich. Aus diesem Grund wird mit RIA (Remote Invocation Architecture) ein neues Konzept eingeführt, das in Abschnitt 8.5 ausführlich beschrieben wird.

8.4.1 Lesen der Konfiguration

Der Dienst „getConfiguration“ des Konfigurationsmanagers nutzt entsprechende Infrastrukturdienste des AMS (Agent Management System) und des DF (Directory Facilitator) um die agentenübergreifende Konfiguration wie Plattformen und Gruppen einer verteilten Anwendung zu ermitteln. Bei allen zur Anwendung gehörenden Agenten wird der Dienst „showAgentConfiguration“ für die Abfrage der agenteninternen Konfiguration wie Agentenrollen sowie einzelne Fähigkeiten und Verhaltensweisen in Anspruch genommen, der auf der Introspektion von Agenten basiert.

Die so erstellten Konfigurationslisten enthalten alle Elemente der Anwendung und der zugehörigen Agenten, die für den Dienstnutzer laut Berechtigung

(siehe Abschnitt 8.6) auch sichtbar sind. Sie können für die Rekonstruktion eines verschwundenen Agenten oder zur Vervielfältigung eines existierenden Agenten (Klonen) verwendet werden.

8.4.2 Ändern der Konfiguration

Um während der Laufzeit Einfluss auf die Agentenkonfiguration nehmen zu können stellt der Konfigurationsmanager den Dienst „setConfiguration“ zur Verfügung. Dieser Dienst nutzt entsprechende Infrastrukturdienste der Agentenplattformen um die agentenübergreifende Konfiguration durch Erzeugung, Migration, Klonen und Terminierung von Agenten zu verändern. Bei allen zur Anwendung gehörenden Agenten wird der Dienst „changeAgentConfiguration“ für die Änderung der agenteninternen Konfiguration wie Agentenrollen sowie einzelne Fähigkeiten und Verhaltensweisen in Anspruch genommen, der auf der Manipulation von Agenten basiert.

Bei der Veränderung von Konfigurationen muss vorher überprüft werden, inwieweit bestimmte Manipulationen erlaubt oder sinnvoll sind. Dies umfasst einerseits die bereits in Abschnitt 8.2 erwähnten Konsistenzprüfungen als auch das Abprüfen der Berechtigungen des Benutzers. Somit könnte beispielsweise einem Administrator das Recht eingeräumt werden, dem Agenten neue Kommunikationskomponenten hinzuzufügen, aber jedem beliebigen anderen Nutzer diese Aktion verboten werden.

8.5 Starten von Agentenplattformen

Für die Erzeugung von Agentenplattformen wird auf den entsprechenden Rechnern ein sogenannter RIA-Server (Remote Invocation Architecture) bereitgestellt [Cha04], der beliebige Java-Programme unter Angabe einer XML-basierten Konfiguration starten kann. Der RIA-Client, der im Konfigurationsmanager enthalten ist, überträgt oder aktualisiert vorher gegebenenfalls benötigte Bibliotheken, falls sie in der angegebenen Version noch nicht auf dem Server vorhanden sind. Nach dem Starten der Anwendung bekommt der Client das Ergebnis mitgeteilt. Dieser Ansatz kann zwar auch für andere Programmiersprachen eingesetzt werden, jedoch müssen bei plattformabhängigen Sprachen für die Auswahl der relevanten Bibliotheken zusätzlich die Besonderheiten der einzelnen Zielrechner berücksichtigt werden.

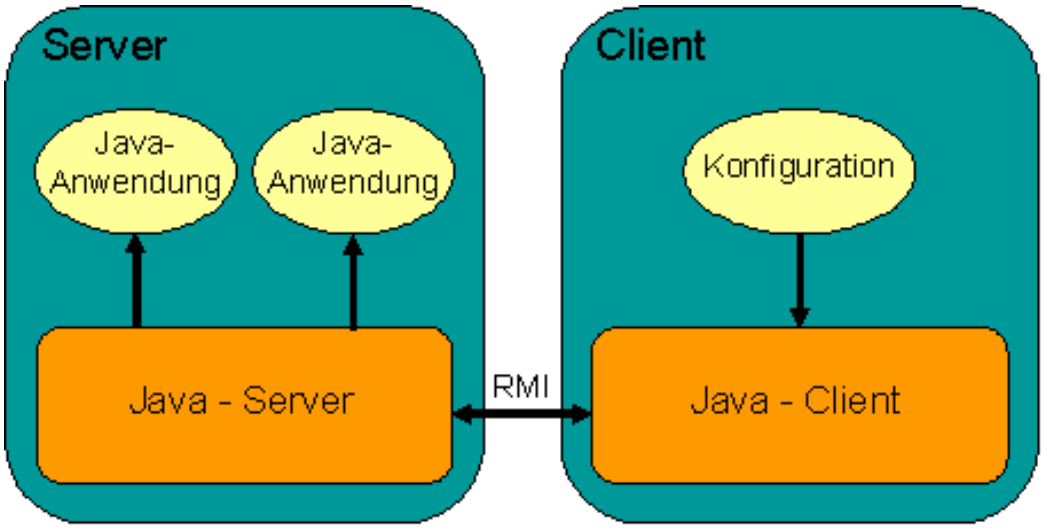

Abbildung 8.6: Remote Invocation Architecture [Cha04]

Die Konfiguration enthält unter anderem die Adresse des Servers, den Namen der Anwendung, die Klasse mit der die Anwendung (Agentenplattform) gestartet wird, den vollständigen Klassenpfad und die unterstützten Java-Versionen.

8.6 Rechteverwaltung

Bei der Konfiguration von agentenbasierten Anwendungen sollen nicht alle Details für jeden Teilnehmer sichtbar oder auch änderbar sein. Um je nach Dienstnutzer Teile der Liste ein- oder ausblenden zu können, ermöglicht eine Rechteverwaltung die Definition von Konfigurationspolitiken. Damit können Rechte einzelner Benutzer oder Gruppen von Benutzern bezüglich der Sichtbarkeit und der Änderungsmöglichkeit festgelegt werden. Ein Plattform-Administrator sollte beispielsweise die technischen Werte einer Plattform lesen und einschränken können, da er für deren Ressourcen zuständig ist, muss aber hierzu nicht unbedingt die Fähigkeiten eines Agenten sehen. Dagegen sollte der Besitzer eines Agenten das Recht haben, sämtliche Angaben zum Agenten lesen und verändern zu können.

Die in Abbildung 8.7 dargestellte Ontologie zeigt den Aufbau einer Konfigurationspolitik (ConfigPolicy), bei der einer Menge von Benutzern (owners) bestimmte Berechtigungen (settings) für eine Menge von Agenten (agents)

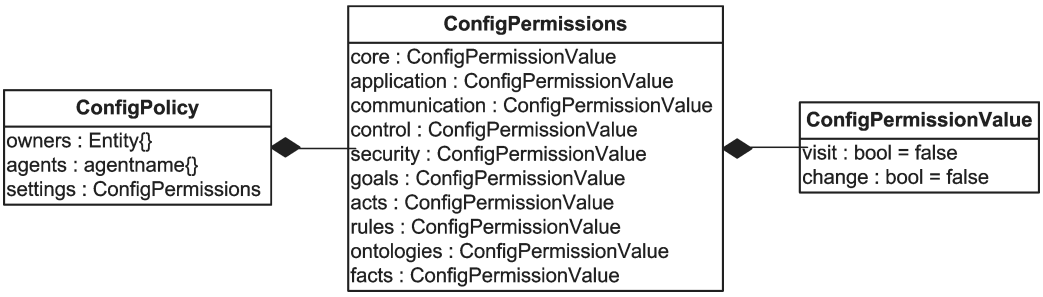

Abbildung 8.7: Ontologie zur Definition von Konfigurationspolitiken

zugeordnet werden können. Bei den Berechtigungen (ConfigPermissions) wird beispielsweise zwischen der Anwendungs- (application), Kommunikations- (communication) und Sicherheitskonfiguration (security) sowie der Konfiguration von Zielen (goals), Aktionen (acts) und Ontologien (ontologies) unterschieden. Weitere Unterscheidungen können durch Spezialisierungen dieser Kategorie vorgenommen werden. Für jede dieser Konfigurationen kann angegeben werden, ob sie sichtbar (visit) und änderbar (change) sein sollen.

Für die Verwaltung der Konfigurationspolitik stehen folgende Dienste zur Verfügung:

- *searchConfigurationPolicy*: Liefert die Konfigurationspolitiken eines Benutzers aus einer Rechteverwaltung.
- *getConfigurationPolicies*: Liefert alle Konfigurationspolitiken einer Rechteverwaltung.
- *setConfigurationPolicies*: Aktualisiert alle Konfigurationspolitiken innerhalb einer Rechteverwaltung.

Die Definition der Konfigurationspolitiken ist durch entsprechende Werkzeuge zu unterstützen, die geeignete Benutzerschnittstellen anbieten.

8.7 Zusammenfassung

In diesem Kapitel wurde basierend auf den Basismechanismen der Managementinfrastruktur ein Konfigurationsmanagement entwickelt, das die Erhebung, Inventarisierung, Kontrolle, Initialisierung, Änderung und Anpassung aller relevanten Systemelemente wie Plattformen, Agenten und Dienste umfasst. Hierbei wird zwischen der Konfiguration einzelner Agenten und der Konfiguration einer agentenbasierten Anwendung unterschieden. Die Agentenkonfiguration nutzt die Introspektions- und Manipulationseigenschaft der Agenten während die Anwendungskonfiguration auf der Agenteninfrastruktur aufbaut.

Ausgehend von einem Grobkonzept, das die an der Konfiguration beteiligten Rollen und deren Aufgaben definiert, wurden anschließend das Konzept der Konfigurationslisten, die Verwaltung von Agentenrollen, die einzelnen Konfigurationsdienste, das Starten einer Initialkonfiguration sowie die Verwaltung der Konfigurationsrechte genauer beschrieben. Die Konfigurationslisten ermöglichen die Konstruktion und die anschließende Erzeugung und Veränderung eines Agentensystems, indem sie den Aufbau des Systems mit seinen Bestandteilen definieren. Eine Konfiguration kann dabei neben den allgemeingültigen Angaben auch architekturspezifische Elemente enthalten. Sämtliche Angaben in den Konfigurationslisten werden für eine Konsistenzprüfung genutzt, um schon frühzeitig Fehler oder mögliche Probleme ausschließen zu können. Agentenrollen erhöhen die Übersichtlichkeit und Wiederverwendbarkeit einer Konfiguration indem sie eine Menge von zusammengehörigen Fähigkeiten, Eigenschaften und Verhaltensweisen von Agenten kapseln. Zum initialen Starten einer Agentenplattform auf einem entfernten Rechner wurde die Remote Invocation Architecture (RIA) entworfen. Mit Hilfe der Rechteverwaltung können Konfigurationspolitiken definiert werden, die die Rechte einzelner Benutzer oder Gruppen von Benutzern bezüglich der Sichtbarkeit und Änderungsmöglichkeit von Elementen einer Konfiguration beschreiben. Die Rechteverwaltung könnte aber auch in ein umfassenderes Sicherheitskonzept integriert werden, das allerdings in dieser Arbeit nicht betrachtet wird.

Auch das Konfigurationsmanagement wurde prototypisch mit der Agentenentwicklungsumgebung JIAC IV implementiert. Das bedeutet, dass die Konfigurationslisten als Ontologien modelliert wurden und die Konfigurationsdienste wie auch die Rechteverwaltung durch verschiedene Managementagenten bereitgestellt werden.

Kapitel 9
Zusammenfassung und Ausblick

Im abschließenden Kapitel wird die komplette Arbeit noch einmal kurz zusammengefasst, eine Bewertung der entwickelten Managementinfrastruktur vorgenommen und einen Ausblick auf mögliche Erweiterungen gegeben.

9.1 Zusammenfassung

In dieser Arbeit wurde ein generischer Ansatz für die Integration von Managementmechanismen in agentenbasierte Systeme vorgestellt. Generisch bedeutet, dass der hier beschriebene Lösungsansatz MIAS unabhängig von konkreten Anwendungsfeldern verwendet werden kann und nur die Besonderheiten der Agententechnologie berücksichtigt.

Auch wenn Agenten ihre Ziele gewöhnlich autonom verfolgen, ist es wichtig sicherzustellen, dass sie stets im Sinne des Eigentümers handeln. Somit sind nicht zuletzt für den industriellen und kommerziellen Einsatz von Agentensystemen umfangreiche Managementfunktionen notwendig, die es den verschiedenen beteiligten Rollen ermöglichen auf hoher Abstraktionsebene die entsprechenden Teile des System sowohl manuell als auch automatisiert zu überwachen und gegebenenfalls anzupassen. Diese Anforderung konnte anhand eines Beispielszenarios aus dem Bereich der Verkehrstelematik verdeutlicht werden, bei dem agentenbasierte Dienste in ein Fahrzeug geladen werden, die im Falle einer Nutzung auf verschiedenen Marktplätzen nach Leistungen externer Anbieter suchen.

Basierend auf dem Ziel nach einer universell einsetzbaren Managementinfrastruktur für agentenbasierte Systeme wurde der Stand der Technik sowohl im Bereich des Managements verteilter Systeme inklusive Web Services als auch im Bereich der Agententechnologie analysiert und bewertet. Insbesondere erfolgte eine Untersuchung von Konzepten, Standards und existierenden Lösungen beider Bereiche. Hierbei konnte festgestellt werden, dass sich die Managementstandards überwiegend mit der Netzwerkebene beschäftigen und somit höherwertige Managementfunktionen nicht weit genug spezifiziert sind, um sie auch auf Agentensysteme anwenden zu können. Die im Bereich der Agententechnologie entwickelten Managementspezifikationen decken nur die Ebene der Multiagentensysteme ab und berücksichtigen nicht die Überwachung einzelner Agenten. Existierende Frameworks bieten nur rudimentäre und proprietäre Schnittstellen für das Management von Agenten an. Aus diesen Gründen wurde ein neuer Ansatz entwickelt, der zwar bestehende Ansätze teilweise integriert aber auch die besonderen Eigenschaften der Agententechnologie berücksichtigt und nutzt.

Da sämtliche höherwertige Managementfunktionen (z.B. Diagnose und Behebung von Fehlern, Bewertung und Beseitigung von Engpässen, Abrechnung und Steuerung von Diensten) auf Basismechanismen wie Introspektion und Manipulation von Agenten beruhen, wurde ein dreischichtiges Managementmodell entworfen, bei dem die mittlere Ebene als Adapter zwischen der spezifischen Managementschnittstelle der verwendeten Agentenarchitektur und den möglichst universell einsetzbaren höherwertigen Managementfunktionen agiert. Dadurch ist ein Einsatz für eine große Anzahl von Agentenarchitekturen möglich. Dieser Ansatz wurde prototypisch am Beispiel der BDI-basierten und dienstorientierten Architektur von JIAC umgesetzt und hat sich anschließend in verschiedenen Anwendungsprojekten (z.B BerlinTainment und BIB3R) des DAI-Labors bewährt. Die mittlere Ebene mit den Basismechanismen wurde noch ergänzt um eine Agenteninfrastruktur für die Verwaltung ganzer Agentensysteme und der angebotenen Dienste, einen Persistenzmechanismus für die Speicherung von Agentenzuständen und das Tracing von Agenten um deren Handlungen auch später noch nachvollziehen zu können. Bereits auf dieser niedrigen Abstraktionsebene wurden verschiedene Werkzeuge zur Administration bereitgestellt.

Die oberste Ebene des Managementmodells orientiert sich an den FCAPS genannten Funktionsbereichen, die innerhalb der Spezifikationen zum OSI-Management beschrieben sind. In dieser Arbeit wurden die Bereiche Abrechnungs- und Konfigurationsmanagement genauer betrachtet.

Die entwickelte Infrastruktur für das Abrechnungsmanagement erlaubt eine individuelle und sehr flexible Vergebührung agentenbasierter Dienste. Jedem Dienstnutzer oder Kunden können individuelle Tarife zugeordnet werden. Da in den Tarifmodellen die gebührenrelevanten Ereignisse definiert sind und deren Sammlung während der Dienstnutzung auf der Introspektionsfunktion basiert, können alle messbaren Ereignisse bei der Gebührenberechnung berücksichtigt werden. Auch die Integration des Abrechnungsmechanismus in verschiedene dienstbasierte Agentenarchitekturen ist im Prinzip möglich. Die Konfigurierbarkeit von Tarifmodellen gewährleistet eine einfache Definition neuer Tarife durch den Dienstanbieter, wofür auch ein Werkzeug bereitgestellt wurde. Durch entsprechende Regeln im Abrechnungsprozess, die auf die Manipulationsfunktion der Agenten zurückgreifen, kann beispielsweise zusätzlich Einfluss auf die Erbringung der Dienste genommen oder eine Kostenkontrolle durch den Dienstnutzer ermöglicht werden.

Das Konfigurationsmanagement erlaubt die sehr genaue Anpassung eines laufenden Systems an beispielsweise geänderte Bedürfnisse. Das wird erreicht durch Abfrage der aktuellen Konfigurationsliste des Systems und der anschließenden Umkonfiguration basierend auf der veränderten Liste. Die geänderte Konfigurationsliste kann zuvor teilweise auf Konsistenz geprüft werden, um schon frühzeitig mögliche Probleme ausschließen zu können. Agentenrollen innerhalb einer Konfiguration erlauben eine Kapselung und somit leichtere Wiederverwendung von zusammenhängenden Agentenbestandteilen. Die Konfiguration erfolgt sowohl auf Ebene einzelner Agenten durch Nutzung der Introspektions- und Manipulationsfunktion als auch auf Ebene kompletter Anwendungen durch Verwendung der Agenteninfrastruktur und einer Remote Invocation Architecture. Berechtigungen geben an, welche Elemente des Systems ein bestimmter Benutzer oder eine Benutzergruppe einsehen oder verändern darf.

9.2 Bewertung

Da keine vergleichbare Managementlösung für agentenbasierte Systeme existiert, bleibt nur die Möglichkeit MIAS gegenüber den zuvor definierten Anforderungen oder gegenüber Managementsystemen anderer Bereiche zu bewerten.

Eine der wesentlichen Anforderung war die beliebige Erweiterbarkeit von MIAS um zusätzliche höherwertige Managementaktivitäten auf Basis wie-

derverwendbarer Basismechanismen. Dies wird durch die mittlere Ebene des Managementmodells gewährleistet, die unter anderem generische Dienste zur Überwachung und Steuerung von Agenten und zur Administration der Agenteninfrastruktur bereitstellt. Diese Erweiterbarkeit wurde am Beispiel des Abrechnungs- und des Konfigurationsmanagements aufgezeigt.

Mit dem in dieser Arbeit entwickelten Abrechnungsmanagement lassen sich beliebige Geschäftsmodelle abdecken, da die Tarifmodelle und zugehörige Aktionen frei implementiert werden können. Eine Berücksichtigung beliebiger messbarer Ereignisse ist dabei möglich. Die Definition und Anpassung von Tarifen kann vom Dienstanbieter durch Konfiguration wiederverwendbarer Tarifmodelle unter Verwendung eines Werkzeuges einfach vorgenommen werden.

Das Konfigurationsmanagement ermöglicht die Anpassung des Agentensystems durch Veränderung der aktuellen Konfigurationslisten. Dabei werden Abhängigkeiten zwischen einzelnen Bestandteilen und Agentenrollen sowie notwendige Bedingungen überprüft.

Eine weitere Anforderung war eine möglichst große Unabhängigkeit von konkreten Agentenarchitekturen zu bewahren, so dass MIAS prinzipiell unterschiedlichste und sogar heterogene Agentensysteme verwalten kann. Dies wurde ebenfalls durch die mittlere Managementebene erreicht, die eine Vermittlung zwischen der spezifischen Managementschnittstelle der unteren Managementebene und den generischen Basismechanismen vornimmt. Da die prototypische Realisierung von MIAS vorerst nur für JIAC erfolgte, kann diese Anforderung noch nicht in der Praxis evaluiert werden.

Auch für die Implementierung der höherwertigen Managementfunktionen wurde prototypisch JIAC verwendet, wodurch einige Vorteile dieser Agententechnologie auch dem Management zu Gute kommen. Beispielsweise können aufgrund der Skalierbarkeit kritische Managementprozesse jederzeit auf mehrere Agenten verteilt werden. Um Managementaktionen auch bei starker Überlastung eines Agenten zeitnah ausführen zu können, besteht die Möglichkeit den Managementaktionen eine höhere Priorität als allen anderen Aktionen zu geben. Dieses Merkmal steht zwar seitens JIAC zur Verfügung, wird aber von anderen Agentenarchitekturen nicht angeboten.

Ein besonderes Merkmal agentenbasierter Systeme stellt die Autonomie der Agenten dar. Die Managementinfrastruktur sollte daher die Autonomie der verwalteten Agenten beibehalten und nicht verhindern. D.h. Agenten können auch Managementaktionen verweigern oder anders als erwartet erbringen. Um trotzdem dem Bedürfnis nach Einflussnahme gerecht zu werden, bietet

sich die Definition und Umsetzung von Autonomielevel an, die jeweils einen bestimmten Grad an Überwachung und Steuerung zulassen. Auf der obersten Autonomieebene wäre dann nur noch der Entzug von Ressourcen möglich bzw. garantiert.

9.3 Ausblick

Auf Ebene der FCAPS Funktionsbereiche wurden bereits das Abrechnungs- und das Konfigurationsmanagement prototypisch realisiert. Das zugrundeliegende Managementmodell erlaubt aber auch die Bereitstellung weiterer Mechanismen insbesondere zu den Bereichen Fehler-, Leistungs- und Sicherheitsmanagement. Zu diesen Bereichen wurde im Rahmen dieser Arbeit aber noch kein Konzept entwickelt.

Auch auf Ebene der Basismechanismen ist eine Zugriffskontrolle auf die Introspektions- und Manipulationsfunktion von Agenten sinnvoll. Diese Zugriffskontrolle sollte nicht nur für komplette Agenten sondern auch für einzelne Agentenelemente definiert werden können. Bei der Manipulation von BDI-basierten Agenten ist zusätzlich auf die Konsistenz ihres Wissens und ihrer Ziele zu achten, indem Widersprüche erkannt und entsprechend behandelt werden.

Des weiteren entstehen zur Zeit neue Forschungsaktivitäten im Bereich „autonomic computing“. Die Idee darin besteht die immer komplexer werdenden Systeme so intelligent zu gestalten, dass sie selbständig und möglichst ohne Eingriff des Menschen anstehende Probleme lösen können. Das bedeutet aber nicht, dass keine Steuerung mehr möglich sein wird, sondern nur dass der Umfang notwendiger Eingriffe reduziert und somit der Benutzer eines Systems von schwierigen, umfangreichen, fehleranfälligen oder gefährlichen Aufgaben zum großen Teil entlastet wird. Hier bietet sich insbesondere die Agententechnologie an, die Autonomie als eines der wichtigsten Merkmale angibt. Angewendet auf das Management agentenbasierter Systeme bedeutet dies, dass diese Systeme ein sogenanntes „self management“ unterstützen indem sie beispielsweise die Fähigkeit zur Selbstheilung, Selbstkonfiguration, Selbstoptimierung oder zum Selbstschutz besitzen.

Teil IV

Anhang

Abkürzungsverzeichnis

AAA Authentifizierung, Autorisierung, Accounting

ACC Agent Communication Channel

ACL Agent Communication Language

ADIF Accounting Data Interchange Format

ADS Agent Discovery Service

AID Agent Identifier

AMS Agent Management System

AOT Agent Oriented Technology

AP Agent Platform

ASE Application Service Element

ASN.1 Abstract Syntax Notation One

CASA Component Architecture for Service Agents

CCITT Comité Consultatif International Téléphonique et Télégraphique

CMIP Common Management Information Protocol

CMIS Common Management Information Service

CMISE Common Management Information Service Element

CORBA Common Object Request Broker Architecture

DAP Directory Access Protocol

DCN Data Communication Network

DF Directory Facilitator

DIT Directory Information Tree

DSA Directory System Agent

DSP Directory System Protocol

DUA Directory User Agent

FCAPS Fault-, Configuration-, Accounting-, Performance-, Securitymanagement

FCR Federated Charging and Rating Facility

FIPA Foundation for Intelligent Physical Agents

GDMO Guidelines for the Definition of Management Objects

GUI Graphical User Interface

HAP Home Agent Platform

HTTP Hypertext Transfer Protocol

ICCAS Internet Charge Calculation und Accounting System

ICL Interagent Communication Language

IDL Interface Definition Language

IETF Internet Engineering Task Force

IPDR Internet Protocol Detail Record

ISO International Organization for Standardization

ITIL Information Technology Infrastructure Library

ITU International Telecommunication Union

JADE Java Agent Development Framework

JIAC Java-based Intelligent Agent Componentware

JVM Java Virtual Machine

KQML Knowledge Query and Manipulation Language

LCN Local Communication Network

MF Mediation Function

MIB Management Information Base

MO Managed Object

MTS Message Transport Service

MASIF Mobile Agent System Interoperability Facility

NEF Network Element Function

NGOSS New Generation Operations Systems and Software

OMG Object Management Group

OSF Operation Systems Function

OSI Open Systems Interconnection

OSS Operational Support System

QAF Q-Adapter Function

QoS Quality of Service

RADIUS Remote Authentication Dial In User Service

RDN Relative Distinguish Name

RFC Request for Comment

RIA Remote Invocation Architecture

RMI Remote Method Invocation

RPC Remote Procedure Call

SET Secure Electronic Transaction

SGMP Simple Gateway Monitoring Protocol

SL Semantic Language

SMAE System Management Application Entity

SMFA Specific Management Functional Area

SMI Structure of Management Information

SMP System Management Process

SNMP Simple Network Management Protocol

TACACS+ Terminal Access Controller Access-Control System Plus

TCP Transmission Control Protocol

TINA Telecommunications Information Networking Architecture

TMN Telecommunications Management Network

TOM Telecom Operations Map

UDP User Datagram Protocol

URI Universal Resource Identifier

URL Uniform Resource Locator

WSF Workstation Function

WWW World Wide Web

XML Extensible Markup Language

Glossar

Abrechnungsmanagement Unter dem Abrechnungsmanagement wird die Sammlung von Daten über Ressourcennutzungen verstanden. Diese Daten können beispielsweise für die Abrechnung von *Diensten* verwendet werden, sofern sie sich den *Dienstnutzungen* eindeutig zuordnen lassen.

Agent Ein Agent ist eine virtuelle Einheit (Software), die in der Lage ist ohne direkten Eingriff von außen bestimmte Aufgaben zu erfüllen und dabei Kontrolle über ihre Handlungen besitzt.

Agentendienst Ein Agentendienst ist eine Leistung, die zwischen *Agenten* angeboten wird.

Agentenplattform ist eine Laufzeitumgebung für *Agenten*.

Agentensystem Ein Agentensystem ist eine Menge von *Agentenplattformen* und *Agenten*, die in Beziehung zueinander stehen.

Agent Communication Channel FIPA definiert einen Agent Communication Channel als ein Element einer *Agentenplattform*, das einen *Dienst* für den Transport von Nachrichten für *Agenten* bereitstellt. Es nutzt dabei auch Informationen vom *Agent Management System* und *Directory Facilitator*.

Agent Discovery Service Der Agent Discovery Service ist eine von FIPA spezifizierte Komponente, die basierend auf der JXTA-Technologie für die plattformübergreifende Bekanntmachung bzw. Registrierung von *Agenten* und *Diensten* auf entfernten Endgeräten in sogenannten Ad-Hoc-Netzwerken zuständig ist. Auch das Abonnieren von Informationen wird unterstützt.

Agent Management System FIPA definiert ein Agent Management System als notwendige Komponente einer *Agentenplattform*, die den Zugriff auf die *Agentenplattform* regelt und einen Weiße Seiten *Dienst* für *Agenten* anbietet. Dieser *Dienst* ist auch für die Vergabe eindeutiger Kennungen für *Agenten* verantwortlich.

Benutzerverwaltung Eine Benutzerverwaltung hält Informationen über alle registrierten Benutzer eines Systems bereit. Das können beispielsweise Kontaktdaten, Präferenzen, Verträge, *Tarife* oder Berechtigungsnachweise sein. Benutzer können unterschiedliche Rollen einnehmen (z.B. Kunde, Anbieter, Administrator) und verschiedenen Gruppen angehören.

Billing Unter Billing wird die Rechnungsstellung basierend auf den im Vertrag genannten Konditionen und den zuvor gesammelten Gebührendaten des Kunden verstanden.

Charging Unter Charging wird die Berechnung der Gebühren basierend auf dem im Vertrag angegebenen *Tarif* und den gesammelten Ressourcennutzungsdaten bzw. weiterer Kontextinformationen des Kunden verstanden.

Directory Facilitator FIPA definiert einen Directory Facilitator als optionale Komponente einer *Agentenplattform*, die einen Gelbe Seiten *Dienst* für *Agenten* anbietet.

Dienst Ein Dienst ist eine Leistung, die nach außen hin angeboten wird.

Dienstnutzung Eine Dienstnutzung umfasst die gesamte Phase der Inanspruchnahme eines *Dienstes*. Hierzu gehört u.a. die Kommunikation zwischen dem Anbieter und Kunden von der Dienstanfrage bis hin zur Übermittlung des Dienstendes und gegebenenfalls auch der Nachweis der Identität und die Aushandlung der Konditionen.

Faktenbasis Eine Faktenbasis ist eine Komponente innerhalb eines *Agenten*, in der das Wissen des *Agenten* über sich und seine Umgebung in Form von Fakten abgelegt ist.

Hot-Billing Dieses Merkmal erlaubt eine Rechnungsstellung zu jedem Zeitpunkt (d.h. unabhängig von definierten Intervallen) unter eventueller Berücksichtigung noch laufender gebührenpflichtiger Aktionen des Kunden.

Inkasso Unter Inkasso wird der Einzug der in den Rechnungen angegebenen Forderungen des Anbieters an den Kunden verstanden.

Kategorie Eine Kategorie ist ein Element einer JIAC *Ontologie*, das ähnlich zu Java-Klassen die Attribute mit zugehörigen Bedingungen und die Vererbungsbeziehung eines komplexen Datentyps beschreibt. Methoden werden in einer Kategorie allerdings nicht definiert.

Kostenfunktion Eine Kostenfunktion berechnet die aktuelle Gebühr beispielsweise einer *Dienstnutzung* basierend auf gegebenen Parametern und den zugehörigen Ressourcennutzungsdaten.

Kostenkontrolle Die Kostenkontrolle erlaubt einem Kunden schon während der Ausführung einer kostenpflichtigen Aktion den aktuellen Gebührenstand zu verfolgen.

Management Unter Management wird die Betreuung oder Leitung eines Systems über dessen gesamten Lebenszyklus hinweg verstanden. Meist werden dabei aber nur die Phasen ab der Einführung bzw. Auslieferung des Systems betrachtet. In dieser Arbeit umfasst der Begriff vorwiegend die Überwachung und Steuerung des Systems zur Laufzeit, mit dem Ziel einen möglichst optimalen Betrieb zu gewährleisten.

Managementinfrastruktur Eine Managementinfrastruktur ist die Gesamtheit aller Elemente und Funktionen, die für das *Management* eines Systems verantwortlich sind.

Mediationkomponente Eine Mediationkomponente ist eine Agentenkomponente, die Daten über Nutzung oder Änderung von Ressourcen vermittelt. Für die Registrierung bei der Komponente kann ein Ereignisfilter und ein Intervall für die Übermittlung angegeben werden. Die Komponente überwacht anschließend die zuständigen *Messkomponenten* und leitet die gemessenen und gefilterten Ereignisse weiter.

Message Transport Service FIPA definiert ein Message Transport Service als standardmäßigen Mechanismus zur Kommunikation von Nachrichten zwischen beliebigen *Agenten*.

Messkomponente Eine Messkomponente stellt Informationen über die Nutzung oder Änderung bestimmter Ressourcen bereit.

Metaprotokoll Ein Metaprotokoll ist ein generisches Protokoll für eine *Dienstnutzung*, in das beliebige Aushandlungs- und Dienstnutzungsprotokolle integriert werden können.

Metering Unter Metering wird die Messung der Nutzung oder Änderung beliebiger Ressourcen verstanden. Hierfür sind entsprechende *Messkomponenten* zuständig.

Monitoring Unter Monitoring wird die Überwachung eines Systems verstanden. Es umfasst sowohl des *Metering* als auch die Sammlung, Aggregation und Bereitstellung der gemessenen Daten.

Ontologie Ontologien beschreiben Dinge eines bestimmten Anwendungsbereiches und ihre Beziehungen zueinander. Für die Repräsentation von Ontologien stehen verschiedene Sprachen zur Verfügung, die jeweils unterschiedliche Konzepte bereitstellen.

Planelement Ein Planelement ist eine implementierte Handlung eines *Agenten*, die auch als Teil eines komplexen Ablaufes (Planes) ausgeführt werden kann. JIAC unterscheidet verschiedene Arten von Planelementen (z.B. Dienst-, Protokoll-, Skript-, Inferenz-, abstrakte und primitive Planelemente).

Subdienstnutzung Eine Subdienstnutzung ist eine *Dienstnutzung* durch einen Anbieter, die für die Erbringung seines eigenen *Dienstes* notwendig ist.

Tarif Ein Tarif definiert eine Sammlung von Preisen oder eine Berechnungsvorschrift für den Preis einer Leistung (z.B. für eine *Dienstnutzung*). Tarife können durch Parametrisierung von *Tarifmodellen* einfach definiert werden.

Tarifmodell Ein Tarifmodell implementiert eine parametrisierbare *Kostenfunktion*, die auf den unter Verwendung definierter Ereignisfilter gemessenen Ereignissen innerhalb einer Dienstnutzung basiert. Ein Tarifmodell kann sowohl allgemeingültiger als auch dienstspezifischer Art sein.

Literaturverzeichnis

[AAH00] Aboba, Bernard, Jari Arkko und D. Harrington: *Introduction to Accounting Management*. RFC 2975, Oktober 2000.

[ABH+01] Ankolekar, Anupriya, Mark Burstein, Jerry R. Hobbs, Ora Lassila, David L. Martin, Sheila A. McIlraith, Srini Narayanan, Massimo Paolucci, Terry Payne, Katia Sycara und Honglei Zeng: *DAML-S: A Semantic Markup For Web Services*. In: Committee, SWWS Program (Herausgeber): *First Semantic Web Working Symposium (SWWS)*, Seiten 411–430, August 2001.

[ABK+98] Albayrak, Sahin, Karsten Bsufka, Andreas M. Kirchwitz, Hans Schlenker, Kai Seidler und Ralf Sesseler: *Agent-based Traffic Telematics Services*. In: Albayrak, Sahin (Herausgeber): *Intelligent Agents for Telecommunications Applications*, Seiten 148–170. IOS Press, 1998.

[ACZ01] Arkko, Jari, Pat R. Calhoun und Glen Zorn: *DIAMETER Accounting Extension*. IETF work in progress, März 2001.

[AL00] Aboba, Bernard und Dave Lidyard: *The Accounting Data Interchange Format (ADIF)*. IETF work in progress, April 2000.

[Alb98] Albayrak, Sahin: *Introduction to Agent Oriented Technology for Telecommunications*. In: Albayrak, Sahin (Herausgeber): *Intelligent Agents for Telecommunications Applications*, Seiten 1–18. IOS Press, 1998.

[BB00] Brownlee, N. und A. Blount: *Accounting Attributes and Record Formats*. RFC 2924, September 2000.

[BHM+04] Booth, David, Hugo Haas, Francis McCabe, Eric Newcomer, Michael Champion, Chris Ferris und David Orchard: *Web Services Architecture*. W3C Working Group Note, Februar 2004.

[BIP91] Bratman, Michael E., David Israel und Martha Pollack: *Plans and Resource-Bounded Practical Reasoning*. In: Cummins, Robert und John L. Pollock (Herausgeber): *Philosophy and AI: Essays at the Interface*, Seiten 1–22. The MIT Press, Cambridge, Massachusetts, 1991.

[BL94] Berners-Lee, Tim: *Universal Resource Identifiers in WWW*. RFC 1630, Juni 1994.

[BLFM98] Berners-Lee, Tim, R. Fielding und L. Masinter: *Uniform Resource Identifiers (URI): Generic Syntax*. RFC 2396, August 1998.

[BLHL01] Berners-Lee, Tim, James Hendler und Ora Lassila: *The Semantic Web*. Scientific American, 284(5):34–43, Mai 2001.

[BLMM94] Berners-Lee, Tim, L. Masinter und M. McCahill: *Uniform Resource Locators (URL)*. RFC 1738, Dezember 1994.

[Bon02] Bon, Jan Van (Herausgeber): *The Guide to IT Service Management*, Band 1. Addison Wesley, 2002.

[BPSM+04] Bray, Tim, Jean Paoli, C. M. Sperberg-McQueen, Eve Maler, Francois Yergeau und John Cowan: *Extensible Markup Language (XML) 1.1*. W3C Recommendation, Februar 2004.

[Bro91] Brooks, R.: *How to build complete creatures rather than isolated cognitive simulators*, 1991.

[BV06] Bullard, V. und W. Vambenepe: *Web Services Distributed Management: Management using Web Services*. Technischer Bericht, August 2006.

[Cai04] Caire, Giovanni: *JADE: the new kernel and last developments*. http://jade.cselt.it/papers/Jade-the-services-architecture.pdf, September 2004.

[CFSD90] Case, J., M. Fedor, M. Schoffstall und J. Davin: *A Simple Network Management Protocol (SNMP).* RFC 1157, Mai 1990.

[CG97] Carrel, D. und L. Grant: *The TACACS+ Protocol Version 1.78.* IETF work in progress, Januar 1997.

[Cha04] Chamsi, Maher: *Konzeption und Realisierung einer Architektur zum remote Deployment und Management von verteilten Multiagentensystemen.* Diplomarbeit, Technische Universität Berlin, 2004.

[CHvRR04] Clement, Luc, Andrew Hately, Claus von Riegen und Tony Rogers: *UDDI Version 3.0.2.* UDDI Spec Technical Committee Draft, Oktober 2004.

[CLG+03] Calhoun, Pat R., J. Loughney, E. Guttman, Glen Zorn und Jari Arkko: *Diameter Base Protocol.* RFC 3588, September 2003.

[CMM98] Cheyer, Adam, David Martin und Douglas Moran: *The Open Agent Architecture™.* http://www.ai.sri.com/~oaa/oaaslides/, Februar 1998.

[CMRW06] Chinnici, Roberto, Jean-Jacques Moreau, Arthur Ryman und Sanjiva Weerawarana: *Web Services Description Language (WSDL) Version 2.0 Part 1: Core Language.* W3C Candidate Recommendation, Januar 2006.

[DH98] Dawson, F. und T. Howes: *vCard MIME Directory Profile.* RFC 2426, September 1998.

[FBK+01] Fricke, Stefan, Karsten Bsufka, Jan Keiser, Torge Schmidt, Ralf Sesseler und Sahin Albayrak: *Agent-based Telematic Services and Telecom Applications.* Communications of the ACM, 44(4):43–48, 2001.

[Fer99] Ferber, Jacques: *Multi-Agent Systems – An Introduction to Distributed Artificial Intelligence.* Addison-Wesley, 1999.

[FIP02a] FIPA: *FIPA Abstract Architecture Specification.* Foundation for Intelligent Physical Agents, 00001, Dezember 2002.

[FIP02b] FIPA: *FIPA ACL Message Structure Specification.* Foundation for Intelligent Physical Agents, 00061, Dezember 2002.

[FIP02c] FIPA: *FIPA Agent Management Specification*. Foundation for Intelligent Physical Agents, 00023, Dezember 2002.

[FIP02d] FIPA: *FIPA Agent Message Transport Service Specification*. Foundation for Intelligent Physical Agents, 00067, Dezember 2002.

[FIP02e] FIPA: *FIPA SL Content Language Specification*. Foundation for Intelligent Physical Agents, 00008, Dezember 2002.

[FIP02f] FIPA: *FIPA Subscribe Interaction Protocol Specification*. Foundation for Intelligent Physical Agents, 00035, Dezember 2002.

[FIP03a] FIPA: *FIPA Agent Discovery Service Specification*. Foundation for Intelligent Physical Agents, 00095, November 2003.

[FIP03b] FIPA: *FIPA JXTA Discovery Middleware Specification*. Foundation for Intelligent Physical Agents, 00096, November 2003.

[Fis93] Fischer, K.: *The Rule-based Multi-Agent System MAGSY*. In: *Proceedings of the CKBS'92 Workshop*, 1993.

[FOK02] FOKUS: *Federated Charging and Rating Facility*. Version 4.0, September 2002.

[GHM+03] Gudgin, Martin, Marc Hadley, Noah Mendelsohn, Jean-Jacques Moreau und Henrik Frystyk Nielsen: *SOAP Version 1.2 Part 1: Messaging Framework*. W3C Recommendation, Juni 2003.

[GK94] Genesereth, Michael R. und Steven P. Ketchpel: *Software agents*. Commun. ACM, 37(7):48–ff., 1994.

[GSTJ98] Ghanei, Azita, Michael Schmidt, Hermann Többen und Olaf Jentsch: *2. Zwischenbericht zum Projekt MIATA*. November 1998.

[Hes] Hessler, Axel: *MIAC: A Methodology for Intelligent Agent Componentware*. work in progress, Technische Universität Berlin.

[HPS04] He, Hao, Mark Potts und Igor Sedukhin: *Web Service Management: Service Life Cycle*. W3C Note, Februar 2004.

[HRDL99] Hellemans, Patrick, Cliff Redmond, Koen Daenen und Dave Lewis: *Accounting Management in a TINA-Based Service and Network Environment*. In: Zuidweg, H., M. Campolargo, J. Delgado und A. Mullery (Herausgeber): *Lecture Notes in Computer Science*, Seiten 13–24, Heidelberg, April 1999. Springer-Verlag.

[HTW04] Helsinger, Aaron, Michael Thome und Todd Wright: *Cougaar: A Scalable, Distributed Multi-Agent Architecture*. In: *2004 IEEE International Conference on Systems, Man & Cybernetics*, Seiten 1910–1917, The Hague, Oktober 2004. IEEE.

[ING99] ING Project: *D5.1: Accounting: State of the Art*. http://ing.ctit.utwente.nl/WU5/D5.1/index.html, Oktober 1999.

[IPD01] IPDR: *Network Data Management - Usage: For IP-based Service*. Version 3.0, November 2001.

[IT] ITU-T: *Information technology – Open Systems Interconnection – Systems Management:XXX Management Function*. ISO/IEC 10164-(1-22), ITU-T Recommendations X.730-753.

[IT89] ITU-T: *Information processing systems – Open Systems Interconnection – BasicReference Model – Part 4: Management framework*. ISO/IEC 7498-4, ITU-T Recommendation X.700, 1989.

[IT92a] ITU-T: *Information technology – Open Systems Interconnection – Structure of managementinformation: Definition of management information*. ISO/IEC 10165-2, ITU-T Recommendation X.721, 1992.

[IT92b] ITU-T: *Information technology – Open Systems Interconnection – Structure of managementinformation: Guidelines for the definition of managed objects*. ISO/IEC 10165-4, ITU-T Recommendation X.722, 1992.

[IT93] ITU-T: *Information technology – Open Systems Interconnection – Management InformationServices – Structure of management information: Management InformationModel*. ISO/IEC 10165-1, ITU-T Recommendation X.720, 1993.

[IT94] ITU-T: *Information technology – Open Systems Interconnection – Basic ReferenceModel: The Basic Model.* ISO/IEC 7498-1, ITU-T Recommendation X.200, 1994.

[IT95] ITU-T: *Information technology – Open Systems Interconnection – Systems management: Usage metering function for accounting purposes.* ISO/IEC 10164-10, ITU-T Recommendation X.742, April 1995.

[IT97] ITU-T: *TMN Management Functions.* ITU-T Recommendation M.3400, April 1997.

[IT98a] ITU-T: *Information technology – Open Systems Interconnection – Common managementinformation protocol: Specification.* ISO/IEC 9596-1, ITU-T Recommendation X.711, 1998.

[IT98b] ITU-T: *Information technology – Open Systems Interconnection – Common managementinformation service.* ISO/IEC 9595, ITU-T Recommendation X.710, 1998.

[IT98c] ITU-T: *Information technology – Open Systems Interconnection – Systems managementoverview.* ISO/IEC 10040, ITU-T Recommendation X.701, 1998.

[KHA06] Konnerth, Thomas, Benjamin Hirsch und Sahin Albayrak: *JADL — an Agent Description Language for Smart Agents.* In: Baldoni, M. und U. Endriss (Herausgeber): *Declarative Agent Languages and Technologies IV*, Band 4327 der Reihe *LNCS*, Seiten 141–155. Springer Verlag, 2006.

[KHA07] Keiser, Jan, Benjamin Hirsch und Sahin Albayrak: *Agents do it for Money — Accounting features in Agents.* In: Dastani, M., A. El Fallah Segrouchni, A. Ricci und M. Winikoff (Herausgeber): *ProMAS 2007 Post-Proceedings*, Band 4908 der Reihe *LNAI*, Seiten 44–58. Springer Verlag, 2007. to appear.

[LMSW05] Luck, M., P. McBurney, O. Shehory und S. Willmott: *Agent Technology Roadmap.* 2005.

[LS99] Lassila, Ora und R. Swick: *Resource Description Framework (RDF) model and syntax specification.* W3C Recommendation, Februar 1999.

[MBH+04] Martin, David, Mark Burstein, Jerry Hobbs, Ora Lassila, Drew McDermott, Sheila McIlraith, Srini Narayanan, Massimo Paolucci, Bijan Parsia, Terry Payne, Evren Sirin, Naveen Srinivasan und Katia Sycara: *OWL-S: Semantic Markup for Web Services*, November 2004.

[MBL+04] Martin, David, Mark Burstein, Ora Lassila, Massimo Paolucci, Terry Payne und Sheila McIlraith: *Describing Web Services using OWL-S and WSDL*, November 2004.

[MC00] M3I-Consortium: *Charging and Accounting System (CAS) Design*. Deliverable 4 Version 1.01, Juli 2000.

[MC01] M3I-Consortium: *CAS Implementation*. Deliverable 13 Version 1.1, Juli 2001.

[MFHS02] McGuinness, Deborah L., Richard Fikes, James Hendler und Lynn Andrea Stein: *DAML+OIL: An Ontology Language for the Semantic Web*. IEEE Intelligent Systems, 17(5):72–80, September 2002.

[MHR91] Mills, C., G. Hirsch und G. Ruth: *Internet Accounting: Background*. RFC 1272, November 1991.

[MR91] McCloghrie, K. und M. Rose: *Management Information Base for Network Management of TCP/IP-based internets:MIB-II*. RFC 1213, März 1991.

[Mül97] Müller, Jörg P.: *The Design of Intelligent Agents: A Layered Approach*. Springer-Verlag New York, Inc., Secaucus, NJ, USA, 1997.

[MvH03] McGuinness, Deborah L. und Frank van Harmelen: *OWL Web Ontology Language Overview*. World Wide Web Consortium (W3C) Candidate Recommendation, August 2003.

[OMG00] OMG: *Mobile Agent Facility Specification*. Object Management Group, Januar 2000.

[OPF03] Odell, James J., H. Van Dyke Parunak und Mitchell Fleischer: *The Role of Roles in Designing Effective Agent Organizations*. In: *Software Engineering for Large-Scale Multi-Agent Systems: Research Issues and Practical Applications*, Seiten 27–38. 2003.

[Rad03] Radisic, Igor: *Ein prozessorientierter, policy-basierter Ansatz für ein integriertes, dienstorientiertes Abrechnungsmanagement.* Doktorarbeit, Ludwig-Maximilians-Universität München, Januar 2003.

[RG91] Rao, A. S. und M. P. Georgeff: *Modeling Rational Agents within a BDI-Architecture.* In: Allen, J., R. Fikes und E. Sandewall (Herausgeber): *Principles of Knowledge Representation and Reasoning: Proc. of the*, Seiten 473–484. Morgan Kaufmann, San Mateo, CA, 1991.

[Rig00] Rigney, C.: *RADIUS Accounting.* RFC 2866, Juni 2000.

[RM90] Rose, M. und K. McCloghrie: *Structure and Identification of Management Information for TCP/IP-basedInternets.* RFC 1155, Mai 1990.

[RN95] Russell, Stuart und Peter Norvig: *Artificial Intelligence: A Modern Approach.* Prentice Hall, 1995.

[RRSW00] Rigney, C., A. Rubens, W. Simpson und S. Willens: *Remote Authentication Dial In User Service (RADIUS).* RFC 2865, Juni 2000.

[Sch85] Schaik, Edward A. Van: *A management system for the information business: organizational analysis.* Prentice-Hall, Inc., Upper Saddle River, NJ, USA, 1985.

[Sch02] Schmidt, Torge: *ASITA: Advanced Security Infrastructure for Multi-Agent-Applications in the Telematic Area.* Doktorarbeit, Technische Universität Berlin, Januar 2002.

[Sea69] Searle, John R.: *Speech Acts : An Essay in the Philosophy of Language.* Cambridge University Press, 1969.

[Ses02] Sesseler, Ralf: *Eine modulare Architektur für dienstbasierte Interaktionen zwischen Agenten.* Doktorarbeit, Technische Universität Berlin, Januar 2002.

[SL97] Schmid, B. und M. Lindemann: *Elemente eines Referenzmodells Elektronischer Märkte.* Universität St. Gallen, 1997.

[SRGF01] Stiller, Burkhard, Peter Reichl, Jan Gerke und Placi Flury: *A Generic and Modular Internet Charging System for the Cumulus Pricing Scheme*, 2001.

[Sta93] Stallings, William: *SNMP, SNMPv2, and CMIP: The Practical Guide to Network-Management Standards*. Addison-Wesley Publishing Company, 1993.

[STGB98] Schmidt, Michael, Hermann Többen, Azita Ghanei und Sigfried Ballmann: *1. Zwischenbericht zum Projekt MIATA*. Juli 1998.

[TC96] TINA-C: *Accounting Management Architecture*. Version 1.2, Dezember 1996.

[TC97] TINA-C: *Service Architecture*. Version 5.0, Baseline document, Juni 1997.

[TMF00] TMF: *Telecom Operations Map*. Guidebook 910, Version 2.1, März 2000.

[WJ95] Wooldridge, Michael und Nicholas R. Jennings: *Intelligent Agents: Theory and Practice*. Knowledge Engineering Review, 10(2):115–152, Juni 1995.

[WPHT02] Winikoff, M., L. Padgham, J. Harland und J. Thangarajah: *Declarative and procedural goals in intelligent agent systems*. In: *Proceedings of the Eighth International Conference on Principles of Knowledge Representation and Reasoning (KR2002)*, April 2002.

[WS06] Wilson, K. und I. Sedukhin: *Web Services Distributed Management: Management of Web Services*. Technischer Bericht, August 2006.

[ZPK+99] Zygourakis, Prevedourou, Kirikoglu, Stamoulis, Dramitinos, Kalopsikakis, Kind, Louta, Tzifa, Kaltabani, Demestichas, Liossis, Anagnostou, Raatikainen, Jormakka und Valtari: *Design of advanced agent-based accounting and charging and personal mobility support services*. MONTAGE Deliverable D25, April 1999.