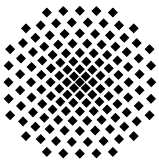
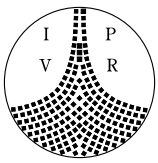


Universität Stuttgart
Fakultät Informatik



Projektbeschreibung: MELODY - Management Environment for Large Open Distributed sYstems

Ernö Kovacs / Cora Burger

Fakultätsbericht 8 / 95

CR-Klassifikation: C.2.4, D.4.4

Fakultät Informatik
Institut für Parallele und
Verteilte Höchstleistungsrechner
Universität Stuttgart
Breitwiesenstraße 20 - 22
D-70565 Stuttgart

Kurzfassung

Mit der zunehmenden Größe und Komplexität offener verteilter Systeme wächst der Bedarf an einer geeigneten systemseitigen Unterstützung bei der Nutzung von Diensten. Eine solche Unterstützung muß sowohl Komponenten zur Vermittlung als auch zur Verwaltung von Diensten enthalten. Da davon auszugehen ist, daß sich verteilte Systeme über verschiedene, administrativ getrennte Domänen erstrecken, die jeweils eigene Strategien verfolgen, müssen die einzelnen Komponenten in Abhängigkeit von diesen Strategien über Domänengrenzen hinweg zusammenarbeiten können.

Zur Lösung dieses Problems kann auf den bereits existierenden Ansätzen für Trading- und Managementdienste aufgesetzt werden. Der vorliegende Bericht stellt ein neues Modell für die Dienstnutzung vor und liefert darauf aufbauend eine ausführliche Klassifikation von Varianten eines Trading-Dienstes. Außerdem werden die Grundprinzipien des kooperativen Tradings eingeführt und die Bandbreite der möglichen Lösungen aufgezeigt. Abschließend wird das bisher im MELODY-Projekt (Management Environment for Large Open Distributed sYstems) verwirklichte System präsentiert, dessen Schwerpunkte auf der Zusammenarbeit zwischen Trading und Management zur Behandlung dynamischer Attribute und auf der Reservierung von Serverressourcen liegen. Der Managementdienst wird nur soweit erläutert, wie er für das Gesamtverständnis notwendig ist.

Abstract

As distributed systems are getting larger and more complex, the importance of supporting service users and providers is increasing enormously. Such support has to contain mediating and managing components. Because distributed systems in general are composed of different administrative domains with specific policies, supporting components have to cooperate across domain boundaries by taking into account these policies.

Suited tools are trading and management mechanisms. Up to now, these approaches have been treated more or less isolated. In the following we introduce a comprehensive approach, based on a general model for service usage and its support. This model is used to classify existing trading approaches and derive concepts for the interworking of all supporting components, especially for the cooperation of traders of different domains. These ideas have been partially implemented in the so-called MELODY system (Management Environment for Large Open Distributed sYstems). Special features of this system concern the interworking of trading and management to handle dynamic properties and the reservation of resources from servers.

Inhaltsverzeichnis

Abbildungsverzeichnis

Tabellenverzeichnis

1. Einleitung

Der über Rechnergrenzen hinweg angebotene Dienst zeichnet sich heutzutage als ein wichtiges Konstruktionsprinzip für offene, verteilte Systeme ab. Die Trennung zwischen dem Dienstanbieter und dem Dienstnutzer gibt beiden Seiten mehr Autonomie und ermöglicht die Schaffung ökonomischer Strukturen im elektronischen Bereich. Man spricht in diesem Zusammenhang von einem *offenen Markt von Diensten*, in dem Anbieter und Nutzer Kontrakte über Bedingungen für Dienstnutzungen aushandeln. Wie in realen Märkten können auf der Basis der bereits verfügbaren Dienste weitere, komplexere Dienste realisiert werden. Insgesamt wird also angestrebt, verteilte Systeme konstruktiv und modular aus existierenden Diensten zusammenzusetzen ([Brod94]).

Infolge der immer größeren und immer enger gekoppelten Netzwerke steht zu erwarten, daß sich der bereits erwähnte offene Markt von Diensten möglicherweise über die ganze Welt erstrecken wird. In naher Zukunft kann also weltweit eine Fülle von Diensten über die Netzwerke angeboten werden. Damit wird der Markt aus vielen konkurrierenden Dienstanbietern bestehen, die gleiche oder ähnliche Dienstleistungen verkaufen. Eine Vielzahl von Benutzern wird den Dienstmarkt nutzen, wobei individuelle und dementsprechend sehr unterschiedliche Kriterien die Dienstauswahl bestimmen. Außerdem kann von einem hochdynamischen, ständigen Änderungen unterworfenen Verhalten von Anbietern und Nutzern ausgegangen werden. Aus allen diesen Gründen muß Anwendern und Anbietern systemseitig eine geeignete Unterstützung bei der Dienstnutzung zur Verfügung gestellt werden.

Dies geschieht durch den im folgenden als *Marktinfrastruktur* bezeichneten, eigenständigen Dienst. Rein funktional sind die wichtigsten Komponenten dieser Marktinfrastruktur ein *Trading-Dienst* zur Dienstvermittlung und ein *Managementdienst*, der die Konfiguration, Überwachung und Steuerung des Marktes und der auf ihm angebotenen Dienste ermöglicht.

Ein weiterer Aspekt des offenen Dienstmarktes, der orthogonal zur Aufteilung in Funktionen steht, ist die auf organisatorischen Gegebenheiten beruhende Strukturierung des Marktes in verschiedene Bereiche. Jeder dieser Bereiche verfolgt eigene Strategien, u.a. hinsichtlich der Dienstnutzungen innerhalb seines Bereiches und über seine Grenzen hinaus. Dementsprechend müssen auch die Komponenten der Marktinfrastruktur aus den jeweils beteiligten Bereichen strategiegesteuert miteinander kooperieren können, um sich gegenseitig Zugang zu den jeweiligen Marktteilen zu gewähren und Steuer- oder Überwachungsinformationen auszutauschen (vgl. Abrechnungen im Mobilfunk über Nationengrenzen hinweg).

Angesichts der Komplexität des Problems beschränkt sich der vorliegende Bericht auf einen ersten Ansatz zur Realisierung einer solchen Marktinfrastruktur, die sich zusammensetzt aus mehreren kooperierenden, durch unterschiedliche Strategien gesteuerten Komponenten unterschiedlicher Funktionalität. Die Basis für diesen Ansatz bildet ein neues Systemmodell für die Dienstnutzung und die Interaktionen zwischen Dienstnutzern, -anbietern und der Marktinfrastruktur (Kapitel 2 und 3). Gleichzeitig dient das Modell der Klassifikation der unterschiedlichen Konzepte, die bei der Realisierung von Tradern und Trader-Kooperationen eingesetzt werden können (Kapitel 4 bzw. 5). Wo möglich, werden auch vorhandene Realisierungen in

die Klassifikation eingeordnet. Außerdem setzen die in Kapitel 5 beschriebenen, weitergehenden Überlegungen zur Kooperation zwischen Tradern auf dem Systemmodell auf.

Nach den im ersten Teil des Berichtes beschriebenen Konzepten zur Unterstützung der Teilnehmer in einem offenen Markt von Diensten wird auf die Realisierung im Projekt MELODY (Management Environment for Large Open DIstributed sYstems) genauer eingegangen. MELODY besitzt die folgenden vier thematischen Schwerpunkte:

Trading in offenen Systemen
beschäftigt sich mit den für das Trading typischen Aufgabenstellungen der Dienstvermittlung zwischen Dienstnutzer (Klient) und Diensterbringer (Server) in unterschiedlichen Szenarien. Bei der Vermittlung kann dabei auf dynamische, über das Netzwerk abrufbare Information zurückgegriffen werden. Weiterhin werden bei der Vermittlung unterschiedliche Interessen berücksichtigt - beispielsweise der Wunsch des Dienstnutzers nach einem optimalen Dienst im Unterschied zu dem Wunsch des Systemverwalters nach einer gleichmäßigen Auslastung. Neben diesen Tätigkeiten zur Ermittlung eines geeigneten Dienstes werden in die Vermittlung insbesondere Verhandlungen zur Festlegung einer bestimmten Dienstgüte integriert.

Management verteilter Anwendungen und Querbezug zum Trading
untersucht einerseits die Modellierung von Informationen, die ein Managementsystem über verteilte Anwendungen benötigt, und die Kommunikationsmechanismen, die zum Zugriff auf diese Information erforderlich sind. Andererseits werden die Querbezüge zwischen Trading und Management aufgedeckt. Beispielsweise verwenden der Trading-Dienst und das Management teilweise die gleichen Informationen, um sie zur Erfüllung ihrer jeweiligen Aufgaben auszuwerten.

Effiziente Verbreitung von Zustandsinformation
Einen besonderen Schwerpunkt bilden Untersuchungen, Managementinformation über das System effizient zu verbreiten. Dies ist für das Trading und das Managementsystem in gleicher Weise notwendig, falls der aktuelle Zustand der an einer Dienstnutzung beteiligten Komponente berücksichtigt werden muß.

Kooperative Dienstvermittlung
Wie bereits erwähnt, werden zukünftige verteilte Systeme nicht aus einem einzigen Gebilde mit einer einheitlichen Infrastruktur bestehen sondern aus mehreren autonomen und häufig auch heterogenen Zellen, die nur in genau definierter Weise zusammenarbeiten können. Dabei können Protokolle unterschiedlicher Komplexität zum Einsatz kommen. Gesteuert wird die Zusammenarbeit durch geeignete Kooperationsstrategien, wobei die Möglichkeiten von feststehenden, statischen Kooperationen bis hin zu dynamischen, fall- und situationsbezogenen Kooperationen reichen.

Die Anforderungen an den MELODY-Trader und die daraus abgeleiteten Entwurfsideen werden in Kapitel 6 aufgelistet. In Kapitel 7 werden die Konzepte des Management und ihrer Integration mit dem Trading geschildert, während Kapitel 8 den aktuellen Stand der Implementierung beschreibt. Das anschließende neunte Kapitel faßt den Bericht zusammen und gibt einen Ausblick auf anstehende Arbeiten.

Im Anhang werden die bisher bekannten Ansätze in tabellarischer Form klassifiziert. Ein umfangreiches Literaturverzeichnis gibt einen Überblick über Trading und das Management verteilter Anwendungen.

2. Modelle für die Nutzung verteilter Systeme

Im folgenden Kapitel wird das Modell vorgestellt, auf dem die Forschungsarbeiten des MELODY-Projektes beruhen. Das Systemmodell für die Marktinfrastruktur definiert unter Verwendung des Dienstbegriffes die bei der Nutzung verteilter Systeme beteiligten Komponenten. Es beschreibt, wer welche Aufgaben und Interessen hat und wie diese durch Zusammenarbeit aller zu verwirklichen sind. Dadurch wird einerseits die enge Verzahnung von Trading und Management deutlich. Andererseits zeigt es sich, daß im Rahmen von Dienstvermittlungen auch eine Unterstützung beim Aushandeln von Kontrakten zur Dienstnutzung benötigt wird. Dieser Aspekt wurde in herkömmlichen Vermittlungssystemen nicht berücksichtigt. Neben der funktionalen Aufteilung werden Strukturierungskonzepte und Möglichkeiten zur Steuerung des Systems durch Strategien vorgestellt.

2.1 Systemmodell von Dienstnutzung und Marktinfrastruktur

Die Grundlage des Systemmodells bildet der Dienst. Der Dienst ist eine *Funktionalität*, die erbracht und genutzt werden kann. Er besitzt *Eigenschaften* wie Leistungsparameter oder Kosten. Der *Diensterbringer* arbeitet unter Nutzung eines Rechnernetzes mit dem *Dienstnutzer* zusammen, die zwei unterschiedliche Rollen darstellen. Vor der Nutzung wird zwischen dem Diensterbringer und dem Dienstnutzer eine Vereinbarung (*Dienstkontrakt*) geschlossen, die die Dienstnutzung regelt. Solche Vereinbarungen können die Kosten einer Dienstleistung, die zur Verfügung stehenden Ressourcen, die Qualität des Dienstes, u.v.m. betreffen. Der Dienstkontrakt kann a priori vorhanden sein oder durch Verhandlung dynamisch zur Laufzeit etabliert werden.

Da die hier betrachtete Diensterbringung und -nutzung im Rahmen verteilter Systeme erfolgt, wird außer den beiden bereits geschilderten Rollen als drittes die Rolle des Systemverwalters zur Bereitstellung, Überwachung und Steuerung der Infrastruktur benötigt. Jeder menschliche Benutzer, der eine dieser drei Rollen innehat, ist durch eine geeignete Komponente zu repräsentieren, die sich im System befindet und die Interaktionen zwischen Benutzer und System bewerkstelligt (entspricht dem Benutzer-Account, auf dem z.B. Benutzer-Shells oder persönliche Agenten gemäß [PYF+92] gestartet werden können). Entsprechend der zugehörigen Rolle sind diese Repräsentanten vom Typ „Dienstnutzer “, „Diensterbringer“ oder „Marktinfrastruktur“ und sind mit den jeweiligen typspezifischen Fähigkeiten ausgerüstet.

Zur Unterstützung der Dienstnutzung in verteilten Systemen werden zwei spezielle Diensttypen verwendet. Davon wird derjenige zur Vermittlung zwischen Diensterbringern und -nutzern, der sogenannte Trading-Dienst, grundsätzlich bei Dienstnutzungen benötigt, d.h. auch außerhalb verteilter Systeme (vgl. z. B. Beratungsunternehmen im realen Markt). Der Trading-Dienst hilft bei der Auswahl eines geeigneten Diensterbringers und kann im Auftrag des Dienstnutzers die Verhandlungen zur Etablierung des Dienstkontraktes durchführen. Die Über-

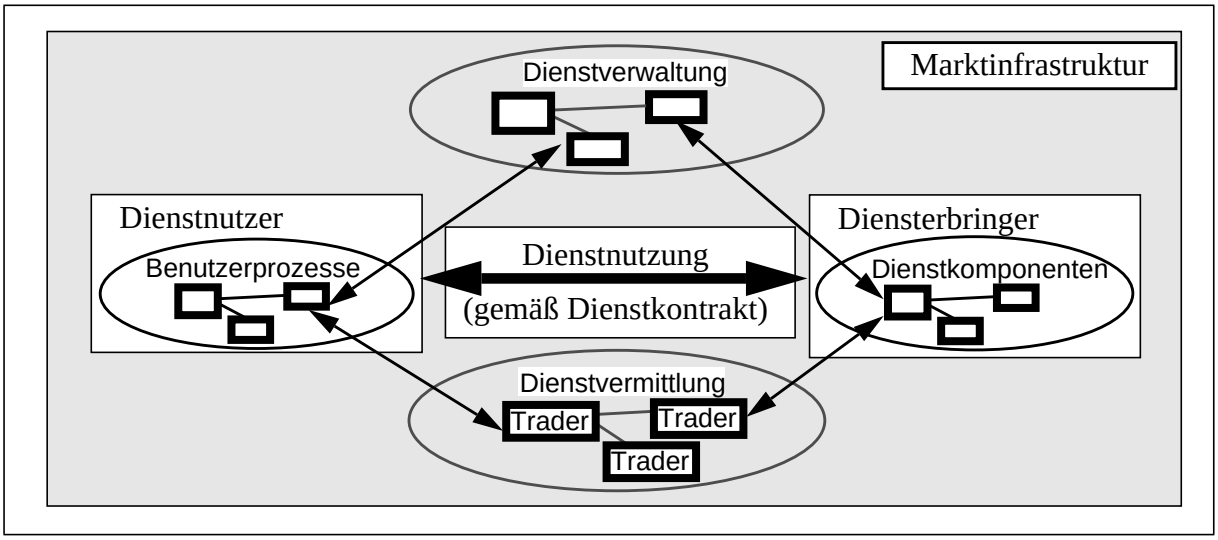


Abbildung 1. Modell für die Dienstnutzung in verteilten Systemen

wachung des Systems sowie die Bereitstellung von Abrechnungs- und Kontrollinformation in verteilten Systemen wird durch spezielle Verwaltungsdienste (auch als Management bezeichnet) geleistet. Diese gehören zur Infrastruktur des verteilten Systems, können jedoch durch Konfiguration an die Erfordernisse des jeweiligen Nutzers (Dienstnutzer, -erbringer, -vermittler) angepaßt werden. Dienstvermittlung und Managementdienste werden zum allgemeinen Dienst “Marktinfrastruktur” zusammengefaßt. Er ermöglicht den sogenannten offenen Markt von Diensten, in dem potentiell jeder Dienste anbieten und jeder Dienste nutzen kann. Insgesamt basiert der rein funktionale Teil des Systemmodells also auf den in Abbildung 1 gezeigten Komponenten. Auf strategiebedingte unterschiedliche Ausprägungen dieser Komponenten wird im Zusammenhang mit ihrer Zugehörigkeit zu unterschiedlichen Organisationen eingegangen (vgl. Abschnitt 2.3).

Eine Dienstnutzung wird erst durch das Zusammenspiel aller dieser Komponenten ermöglicht. Um dieses detaillierter beschreiben zu können, wird das folgende Kooperationsmodell (vgl. [BuSe94]) verwendet, wobei die Komponenten als aktiv handelnde Subjekte aufgefaßt werden.

2.1.1 Dienstnutzung als Kooperation

In welcher Form sich ein Subjekt an einer Kooperation beteiligt, hängt einerseits von seinen Fähigkeiten und Möglichkeiten, die sein Kooperationspotential festlegen, und andererseits von seinen Zielen ab. Dementsprechend lassen sich Subjekte im Hinblick auf ihr Kooperationsverhalten durch folgendes Tupel modellieren

Subjekt = ((Potential, Kosten), (Ziel, Gewichtung))

wobei alle Terme von der Zeit abhängen können. Dabei umfaßt der Begriff des Subjektpotentials die folgenden unterschiedlichen Typen:

- Aktionen, die ein Subjekt ausführen kann.
- Werkzeuge und andere Ressourcen, die zur Ausführung von Aktionen benötigt werden. Diese Ressourcen stehen i.a. nur in einer bestimmten Menge zur Verfügung, deren Wert durch die Ressourcenkapazität vorgegeben ist.
- Wissen über sich selbst, andere Subjekte und die Umgebung allgemein.

Teile des Potentials können anderen Subjekten umsonst oder gegen Anrechnung von Kosten zur Verfügung gestellt werden. Bei Zielen kann grundsätzlich unterschieden werden zwischen denen, die sich auf die Erfüllung einer Aufgabe beziehen, und denen, die die Qualität der Aufgabenerfüllung betreffen. Die Gewichtung von Zielen wird durch die Strategie des Subjektes festgelegt (vgl. Abschnitt 2.3). Sie wird einerseits bei der Planung von Aktivitäten und andererseits bei der Lösung von Konflikten benötigt.

Wenn bei einem Subjekt zwischen den durch das Potential vorgegebenen Möglichkeiten und den durch seine Ziele gestellten Anforderungen eine Diskrepanz besteht, so kann es unter Abwägung mit seinen sonstigen Zielen bestrebt sein, diesen Unterschied durch eine Kooperation auszugleichen. Dementsprechend kooperiert eine Menge von Subjekten während eines bestimmten Zeitintervalles, wenn es im fraglichen Zeitraum unter den Zielen dieser Subjekte mindestens ein Ziel gibt, das nur durch die Vereinigung aller Subjektpotentiale erreicht werden kann, und wenn alle während dieses Zeitintervalles möglicherweise auftretenden Konflikte behoben werden können. Konfliktbehebung erfolgt durch Abschwächen oder Aufgabe widersprüchlicher Ziele in Abhängigkeit von deren Gewichtung.

Um dieses Kooperationsmodell auf die in Abbildung 1 eingeführten Komponenten anwenden zu können, sind sowohl das Potential als auch die Ziele von Diensterbringern, -nutzern und Marktinfrastrukturkomponenten im Hinblick auf ihre Kooperationsbereitschaft zu untersuchen. Tabelle 1 gibt einen Überblick über die dabei beteiligten Aspekte, wobei die Zusammenarbeit aller drei unterschiedlichen Komponententypen im Vordergrund steht. Dagegen wird die Kooperation zwischen Marktinfrastrukturkomponenten, speziell zwischen Dienstvermittlern erst in Kapitel 5 behandelt.

Komponente	Verfügbare Fähigkeiten	Benötigte Fähigkeiten	Qualitative Ziele
Dienstnutzer		Dienstleistung, Unterstützung bei Nutzung	Hohe Dienstqualität, geringe Kosten
Diensterbringer	Dienstleistung	Installation, Konfiguration, Überwachung	Hoher Durchsatz, hohe Einnahmen
Marktinfrastruktur	Management und Dienstvermittlung		Geordneter Ablauf, hohe Ressourcennutzung
Managementdienste	Installation, Konfiguration, Ausführung, Beobachtung, Einfluß auf System	Vermittlung und Plazierung von Management-komponenten	
Trading-Dienst	Dienstvermittlung, Kontraktaushandlung	Information über Dienste	

Tabelle 1: An der Dienstnutzung beteiligte Komponenten als kooperative Subjekte

Wie Tabelle 1 zeigt, benötigen Komponenten Fähigkeiten von anderen und stellen anderen welche zur Verfügung. So benötigt der Dienstnutzer einen bestimmten Dienst und Unterstüt-

zung in Form von Dienstvermittlung und Information über die Nutzung. Dabei ist er in jedem Fall an einer qualitativ hohen Dienstausführung zu niedrigen Kosten interessiert. Im Vergleich zum Dienstnutzer verfügt der Diensterbringer über die komplementäre Fähigkeit des Dienstes. Andererseits benötigt er Unterstützung bei der Installation und Konfiguration seines Dienstes im verteilten System und bei der Überwachung und Abrechnung der Dienstnutzung. Indem er den Dienst über das Rechnernetz anbietet, ist er insbesondere daran interessiert, daß dieser Dienst in geordneter Weise genutzt wird, gut ausgelastet ist und möglichst hohe Einnahmen einbringt. Dadurch steht er potentiell im Konflikt zum Dienstnutzer. Außerdem können unterschiedliche Dienstanbieter in Konkurrenz zueinander stehen.

Die durch Dienstnutzer und -erbringer bei der Dienstnutzung benötigte Unterstützung wird durch die Komponenten der Marktinfrastruktur, d.h. die Management- und Vermittlungsdienste geleistet. Managementdienste in verteilten Systemen erfüllen eine Reihe von Aufgaben wie z. B. Installieren und Konfigurieren von Diensten sowie Ausführung, Überwachung, Steuerung und Verwaltung von Dienstkomponenten. Die bei der Überwachung gesammelten Daten können anderen Komponenten zur Verfügung gestellt werden. Sie werden allgemein zur Beantwortung von Informationsanfragen, z. B. über den aktuellen Zustand von Dienstkomponenten oder bei der Abrechnung von Dienstnutzungen eingesetzt.

Der Trading-Dienst unterstützt die Marktteilnehmer bei der Orientierung am Markt, indem er dem Dienstnutzer die Möglichkeit bietet, bedarfsgerecht einen Dienstleister auszuwählen und einen geeigneten Dienstkontrakt mit ihm auszuhandeln. Den Diensterbringern erlaubt er, ihr Angebot durch die Veröffentlichung einer Dienstbeschreibung bekannt zu geben und ermöglicht dadurch die verstärkte Nutzung der entsprechenden Dienstleistung. Für die Ermittlung geeigneter Diensterbringer benötigt der Trading-Dienst Informationen über Diensteigenschaften, die möglicherweise über die im Angebot spezifizierten hinausgehen oder sich auf dynamische Eigenschaften beziehen.

Wie bereits erwähnt, verfügen Managementdienste über die Fähigkeit der Informationsbeschaffung, stellen also das vom Trading-Dienst benötigte Gegenstück dar. Andererseits erfordert die Funktionalität des Managementdienstes die Aufteilung in einzelne Komponenten, die nach Bedarf im Netz zu verteilen sind. Die Dienstvermittlung erlaubt hierbei sowohl die geeignete Plazierung als auch das Auffinden und Nutzen bereits vorhandener Managementfunktionen. Es besteht also eine enge Verzahnung zwischen Trading und Management, die in bisherigen Arbeiten nicht berücksichtigt wurde.

Als Grundlage für die Behandlung existierender Trader-Konzepte reicht der oben eingeführte Detaillierungsgrad aus. Für weitergehende Überlegungen, speziell auch zu Trader-Kooperationen wird jedoch eine Verfeinerung des Systemmodells benötigt. Dazu soll im folgenden der Ablauf der Kooperation zwischen den einzelnen Komponenten detaillierter untersucht werden, um die Aufgaben der Marktinfrastrukturkomponenten genauer definieren zu können.

2.1.2 Koordination der Dienstnutzung

Koordination wird dazu eingesetzt, kooperative Aktionen und Interaktionen effizient zu organisieren und Konflikte zu lösen. Dementsprechend setzen sich Kooperationen i.a. aus mehreren, möglicherweise gleichzeitig oder überlappend ablaufenden Phasen, den Koordinations- und Realisierungsphasen zusammen (vgl. auch [BuSe94]). Zur Koordination können dabei außer einer Einigung bzgl. des kooperativen Vorgehens die Überwachung aller ausgehandelten Aspekte gehören. Derartige Aushandlungen können sowohl beim Aufbau von Kooperations-

beziehungen als auch während einer Kooperation zur Planung des weiteren kooperativen Vorgehens bzw. zum Abbau der Kooperationsbeziehung benötigt werden.

	Vorbereitung	Planung (externe Konfliktlösung)	Realisierung
Auslösendes Ereignis	Kooperationsbedarf entdeckt	Partner gefunden, Kooperations-/ Planungsbereitschaft oder Konflikt entdeckt	Planung erfolgreich beendet
Aktionen/ Interaktionen	Interne Konfliktlösung, Partnersuche	Interne Konfliktlösung, Verhandlung	Ausgehandelte Aktionen/ Interaktionen
Ergebnis	Partner gefunden oder Kooperation nicht möglich	Planung erfolgreich oder Kooperation nicht möglich	Ziele erreicht, von neuem Kooperations-/Planungs-bereitschaft oder Konflikt

Tabelle 2: Kooperationsphasen

Wie Tabelle 2 zeigt, wird jede Phase durch ein bestimmtes Ereignis ausgelöst, besteht aus Aktionen und Interaktionen zur Erfüllung phasenspezifischer Aufgaben und liefert ein Ergebnis eines bestimmten Typs. Dabei spielen sowohl strategische Gesichtspunkte als auch solche, die Mechanismen betreffen, eine Rolle. Eine detailliertere Behandlung der einzelnen Phasen erfolgt in Kapitel 5 im Zusammenhang mit der Behandlung der Kooperation zwischen Trader-Komponenten. Im folgenden wird genauer erläutert, wie sich das Koordinationsmodell auf die Dienstnutzung in verteilten Systemen anwenden läßt.

Die Vorbereitungsphase besteht beim Dienstnutzer im wesentlichen darin, einen Dienstleistungsbedarf zu entdecken, aus dem sich ein Vermittlungsbedarf ergeben kann. Im Gegensatz dazu verfügt ein Diensterbringer per definitionem über den Bedarf der Nutzung, der seinerseits einen Vermittlungsbedarf enthält. Zu unterscheiden ist die Form der Partnersuche. Während der Dienstnutzer sich von einer ihm bekannten Trader-Komponente über potentielle Partner unterrichten läßt und durch eine Verhandlung mit ihnen die Planungsphase initiiert, verhält sich der Diensterbringer in bisherigen verteilten Systemen relativ passiv und überläßt es dem Trading-Dienst, geeignete Partner zu informieren, die auf den Erbringer zukommen. Abhilfe kann hier eine Nachbildung der in realer Umgebung eingesetzten Werbung schaffen. Die Behandlung der dadurch im Trading-Dienst entstehenden Aufgaben würde jedoch den Rahmen des vorliegenden Berichtes sprengen.

Bei der Kontaktaufnahme zwischen Dienstnutzer und -erbringer ist eine Übereinkunft auszuhandeln, die die Konditionen der Dienstnutzung regelt (externe Konfliktlösung). Dabei ist außer einem gemeinsamen Verständnis über die Funktion des Dienstes festzulegen, welche Qualitätsmerkmale er erbringt und welche Kosten dabei entstehen. Bei geeigneter Angabe des Verhandlungsspielraumes kann diese Aushandlung durch den Trading-Dienst übernommen werden. Die Überwachung der vereinbarten Bedingungen erfolgt entweder direkt gegenseitig oder sie wird an die Martkinfrastruktur delegiert. Wird eine Verletzung durch einen der Partner erkannt oder ergeben sich Änderungen der ursprünglichen Ziele bzw. im Verhalten der Umgebung, so wird die Kooperation entweder abgebrochen oder es kann durch eine neue Verhandlung versucht werden, sie mit geänderten Bedingungen fortzuführen. Wie bereits erwähnt,

bietet es sich an, die herkömmliche Trader-Funktionalität zu erweitern und derartige Verhandlungen im Rahmen von Vermittlungen mit durchzuführen.

Während der Realisierungsphase der Kooperation, d. h. während der Dienstnutzung und im nachhinein kann sowohl beim Dienstnutzer als auch beim -erbringer Bedarf an Information über die Dienstnutzung entstehen, die durch den Managementdienst befriedigt werden kann. Genauso wie im Fall des Vermittlungsbedarfs kann davon ausgegangen werden, daß hierbei keine Partnersuche erforderlich ist. Jeder Dienstnutzer und -erbringer sollte zumindest eine Komponente der Marktinfrastruktur als Zugangspunkt zu einem geeigneten Management- und Trading-Dienst kennen, die dafür sorgen kann, daß der entsprechende Vermittlungs- bzw. Informationsbedarf befriedigt wird.

2.2 Modellierung der beteiligten Komponenten

Um die in den vorangegangenen Abschnitten identifizierten Aufgaben der Marktinfrastruktur umsetzen zu können, muß es möglich sein, die Komponenten des Systemmodells im verteilten System ansprechen und mit ihnen umgehen zu können. Daher soll nun auf die einzelnen Komponenten näher eingegangen werden. Das Modell enthält die drei Rollen „Dienstnutzer“, „Dienstanbieter“ und „Marktinfrastruktur“. Jede Instanz einer Rolle verfügt über eine *Identität*, die z.B. über einen Namen ausgedrückt wird, und über einen *Repräsentanten* im System. Zu einem solchen Repräsentanten existiert mindestens ein *Zugangspunkt*, d.h. eine definierte Stelle wie z.B. eine Adresse, über die Interaktionen zwischen Repräsentanten abgewickelt werden. Durch diese Trennung in Identität und Zugangspunkt von Repräsentanten wird die Mobilität der Rolleninstanzen ermöglicht und damit auch der Mobilität der repräsentierten Benutzer Rechnung getragen. Je nach Realisierung des Repräsentanten (z. B. einzelner Prozeß oder aus mehreren zusammengesetzt) kann der Zugangspunkt ein eindeutiger sein oder es können mehrere Zugangspunkte an unterschiedlichen Stellen im System auftreten (wie z. B. bei Replikaten). Jeder Repräsentant verfolgt individuelle Ziele, die sich in rollenabhängigen *Strategien* ausdrücken (vgl. Abschnitt 2.1.1 und 2.3.3). Außerdem besitzen die Repräsentanten systemtechnisch dargestellte *Eigenschaften*, die z.B. bei der Dienstauswahl berücksichtigt werden.

2.2.1 Modellierung und Realisierung von Dienstnutzern

Dienstnutzer besitzen Repräsentanten, die ihnen den Zugriff auf Dienste ermöglichen (Benutzeragent, englisch User Agent). Die Spanne der verfügbaren Benutzeragenten reicht von einem generischen Klienten, mit dessen Hilfe eine Vielzahl unterschiedlicher Dienste genutzt werden kann (wie z.B. Klienten für die verschiedenen, über das World Wide Web abrufbaren Informationsdienste), über allgemein einsetzbare Standardagenten (z.B. ftp) bis zu Spezialkomponenten, die je nach Einsatzgebiet und Aufgabe bestimmte Eigenschaften eines Dienstes ausnutzen. Die Hauptaufgabe des Benutzeragenten ist die Abwicklung des beim Zugriff auf den Dienst benötigten Kommunikationsprotokolls. Es gibt verschiedene Implementierungsmöglichkeiten für Benutzeragenten, die jeweils bei verschiedenen Arten von Diensten verwendet werden.

Im einfachsten Fall ist der Benutzeragent mit dem Anwendungsprogramm für die Nutzung des Servers integriert. In diesem Fall kann ausschließlich das vorhandene Anwendungsprogramm zur Nutzung des Dienstes verwendet werden. Eine andere Variante realisiert den Benutzeragenten als ein Satz von Subroutinen, die in einer Library zusammengefaßt werden. Diese Library wird zum Programmcode des Anwendungsprogrammes hinzugelinkt. Diese Realisie-

rungsvariante ermöglicht die Nutzung des Dienstes in unterschiedlichen Anwendungen. Ein Benutzeragent kann aber auch als eine Sammlung von Dienstprogrammen realisiert sein, die dann zur Dienstnutzung im Batch- oder im Interaktiv-Betrieb aufgerufen werden, um die Dienstleistung zugänglich zu machen. Bei Diensten, die Betriebsystemfunktionen realisieren (beispielsweise ein verteiltes Dateisystem), wird die Benutzeragent-Funktionalität in den Betriebsystemkern integriert. Abbildung 2 illustriert verschiedene Benutzeragenten-Ansätze.

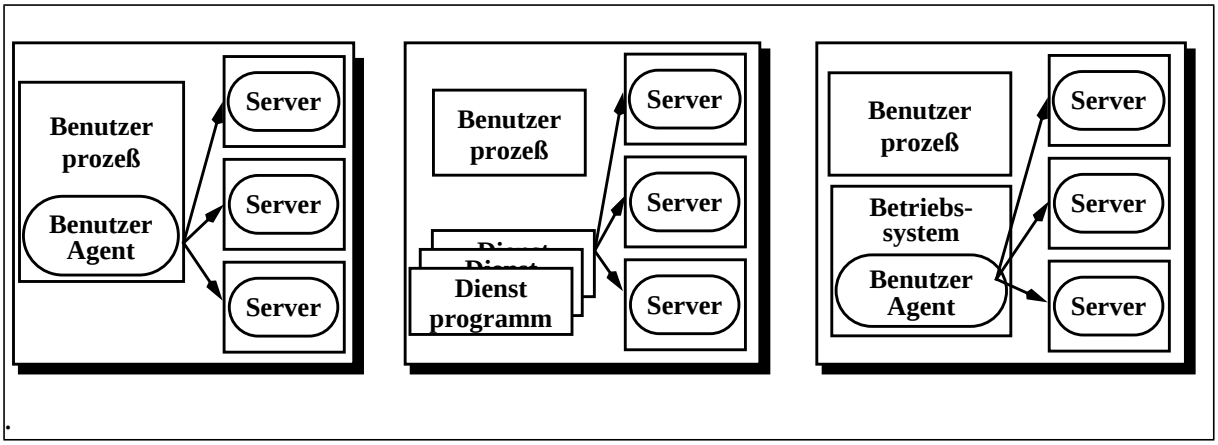


Abbildung 2. Varianten des Benutzeragenten in verteilten Systemen

Der Benutzeragent erbringt die Dienstleistung im Auftrag des Anwenders durch die Nutzung der entsprechenden Server. Daneben kann er, insbesondere im Zusammenhang mit komplexen Aufträgen, verschiedene weitere Aufgaben übernehmen, beispielsweise

- Aufträge durch parallele Anfragen auf unterschiedliche Server verteilen.
- Serverprozesse im verteilten System starten, um sie die gestellte Aufgabe durchführen zu lassen.
- Transparent für den Nutzer an Transaktionen teilnehmen und somit eine konsistenzerhaltende Durchführung der gestellten Aufgabe gewährleisten.
- Im Fehlerfall versuchen, auf einen alternativen Dienst umzuschalten.

Je nach Auslegung des Benutzeragenten und Vorgaben durch den Benutzer kann der Benutzeragent entscheiden, ob, wann und wie ein Trader eingesetzt werden soll und kann diese Nutzung vor dem Dienstanwender verbergen.

2.2.2 Allgemeine Modellierung von Diensten

Diensterbringer besitzen Repräsentanten, die einen Dienst erbringen (Dienstagent, englisch Service Agent). Das Spektrum der möglichen Repräsentanten reicht wie beim Dienstnutzer von generischen Standardagenten (wie z.B. WWW-Server) über spezialisierte Diensterbringer bis hin zu einem Geflecht kooperativer Dienstkomponenten (vgl. Abschnitt 2.2.3 und 2.2.4).

Bei Diensterbringern definiert der Typ der *Dienstschnittstelle* die Art und Weise des Dienstzuganges. Mit dem *Schnittstellentyp* sind daher die genauen Festlegungen der zu verwendenden Dienste (z.B. für die Namensgebung und -resolution, für Authentifizierung und Autorisierung) und Kommunikationsprotokolle (z. B. TCP/IP, OSI, SNA) verbunden. Über die Dienstschnittstelle hinaus legt der *Diensttyp* die Funktionalität fest, die von einem Dienst erbracht wird. Mit anderen Worten, der Schnittstellentyp definiert die syntaktischen Aspekte eines Dienstes, wäh-

rend der Diensttyp die semantischen Aspekte definiert. Häufig existiert eine duale Beziehung, d.h. eine eindeutige Zuordnung, zwischen dem Schnittstellentyp und dem Diensttypen. Jedoch kann es für einen bestimmten Diensttyp verschiedene Schnittstellen und für eine bestimmte Schnittstelle unterschiedliche Diensttypen geben. Am Beispiel unterschiedlicher Drucksysteme läßt sich leicht erkennen, daß trotz gleicher Funktionalität unterschiedliche Dienstschnittstellen vorhanden sein können.

Dienste können zusätzlich mit Hilfe von *Dienstattributen* beschrieben werden, die allgemeine Eigenschaften wie z. B. die Gültigkeitsdauer oder die Dienstgüte und typabhängige Eigenschaften definieren. Beispiele für solche typabhängigen Eigenschaften bei einem Druckerserver sind dpi-Angaben oder die Unterscheidung in Schwarz-Weiß bzw. Farbe. Eine wichtige Rolle spielen Objekte, die von Diensten verwaltet werden, beispielsweise die von einem Dateiserver gehaltenen Dateien. Die Menge solcher *verwalteten Objekte* eines Dienstes ist in unserem Modell eine spezielle Eigenschaft eines Dienstes.

2.2.3 Das Modell eines verteilten Dienstes

Dienste lassen sich nach der Anzahl ihrer Zugangspunkte klassifizieren. Gibt es mehr als einen Zugangspunkt, so handelt es sich um einen verteilten Dienst. In diesem Fall wird die Dienstleistung durch eine Reihe von Dienstleistungskomponenten erbracht, die auf verschiedene Rechner verteilt sein können. Diese Komponenten interagieren miteinander, um einen Dienst zu erbringen. Abbildung 3 veranschaulicht die Situation. Dienstzugangspunkt bei verteilten Diensten ist entweder eine Menge ausgezeichneter Komponenten oder jede beliebige Komponente des entsprechenden Dienstes.

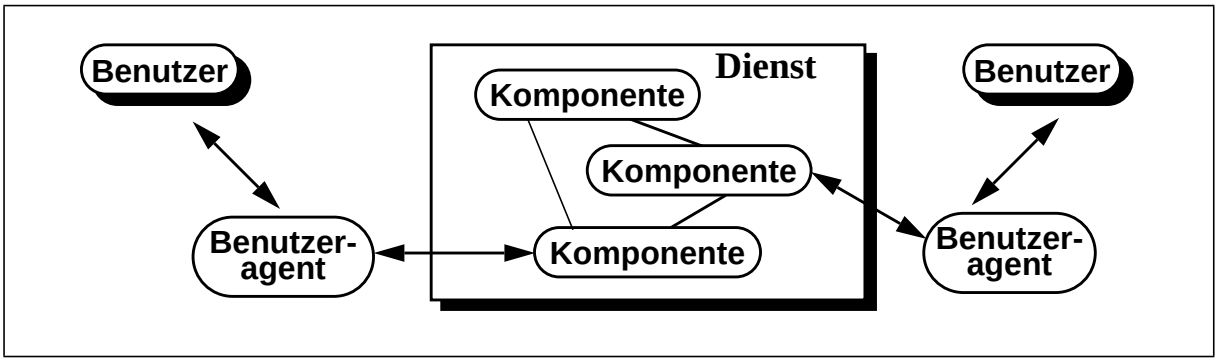


Abbildung 3. Das Modell eines verteilten Dienstes

Das Modell des verteilten Dienstes wird sehr häufig bei der Realisierung von Telekommunikationsdiensten (z.B. E-Mail oder Videokommunikation) eingesetzt. Weitere Beispiele bilden die im vorliegenden Bericht beschriebenen Trading- und Managementdienste.

2.2.4 Zusammengesetzte Dienste

Existiert in einem System bereits eine Menge von Diensten, so können daraus in beliebiger Weise komplexere Dienste kombiniert werden. Ein zusammengesetzter Dienst besteht demnach aus verschiedenen Diensten, die zu einem neuen Dienst integriert wurden. Von verteilten Diensten unterscheiden sich zusammengesetzte Dienste dadurch, daß sie aus elementaren Diensten zusammengesetzt wurden, die jeweils für sich bereits einen Dienst erbringen und auch weiterhin separat genutzt werden können. Insbesondere können elementare Dienste auch

für unterschiedlich zusammengesetzte Dienste herangezogen werden. Der Vorgang des Zusammensetzens erfolgt entweder von Seiten des Benutzers durch dessen Benutzeragenten oder durch einen neuen Diensterbringer.

Hauptgrund für die Realisierung zusammengesetzter Dienste ist die schnellere Erstellung des neuen Dienstes sowie die einfache Integration von neuen und alten Diensten. Beispielsweise werden Workflow-System häufig auf der Basis eines existierenden E-Mail-Dienstes realisiert. Durch die Nutzung des E-Mail-Dienstes erspart sich das Workflow-System, einen eigenen Transportdienst für die zu bearbeitenden Dokumente zu entwickeln. Auch kann es an der im E-Mail-Dienst verwirklichten Benutzerverwaltung partizipieren. Heutzutage geht der Trend bei Workflow-Systemen zu einer offenen Architektur, die die Integration unterschiedlichster Dienste erlaubt.

Zusammengesetzte Dienste benötigen für ein geeignetes Zusammenspiel verschiedener Dienste exakte Vorgaben über die verwendeten Schnittstellen. Dazu sind offene, aber exakt genormte (typisierte) Schnittstellen unbedingt erforderlich. Die Marktinfrastruktur unterstützt diesen konstruktiven Ansatz durch die Berücksichtigung der Schnittstelleninformation bei der Dienstvermittlung.

2.3 Domänen und Strategien

In den vorigen Abschnitten wurden die bei der Dienstnutzung beteiligten Komponententypen behandelt. Im folgenden soll nun auch der räumlichen Ausdehnung des elektronischen Marktes und seiner organisatorischen Struktur Rechnung getragen werden, wodurch sich eine Aufteilung in Gruppen mit gleichen oder ähnlichen Zielen ergibt. Dazu werden die in der Literatur verfügbaren Konzepte beschrieben.

2.3.1 Domänen als Strukturierungskonzepte für verteilte Systeme

Das für die Dienstnutzung angestrebte Zielszenario basiert auf einer weltweiten Vernetzung der verschiedenen Systemplattformen. Infolge der Größe sowie der durch Verteilung und Vielzahl paralleler Aktivitäten verursachten Komplexität geht die Überschaubarkeit und dementsprechend auch die Effizienz völlig verloren. Dem kann nur durch eine geeignete Strukturierung begegnet werden.

Bei den Strukturierungskonzepten ist zu unterscheiden zwischen einfachen Konzepten wie z. B. der Hierarchisierung von Servern oder Serverangeboten in einem Namensbaum und solchen Konzepten, die die Verknüpfung von Strukturierungsmengen mit Strategien erlauben. Letztere bilden die Basis für weitergehende Mechanismen und Konzepte, indem sie z. B. die Einbindung marktwirtschaftlicher Strategien ermöglichen und damit die Grundlage für die Errichtung eines elektronischen Marktes darstellen.

Eine Möglichkeit zur Strukturierung unter Einbindung von Strategien besteht in der Verwendung von Domänenkonzepten, die im Zusammenhang mit dem Management Verteilter Systeme eingeführt wurden (vgl. [SlMo89b], [SlMo89a], [Slom90] und [MoSl91a] gemäß Tabelle A.13). Eine *Domäne* faßt bestimmte Komponenten eines Systems (z. B. Dienstanbieter, Dienstnutzer, Systemplattform, Rechner und Netzwerke) zusammen, beinhaltet also eine Menge von zusammengehörenden *Domänenobjekten.* Die dabei verwendeten Kriterien können unterschiedlicher Natur sein. Einige Beispiele für Domänen enthält Tabelle 3.

Domänenart	Erläuterung
Verwaltungsdomäne	Definiert einen einheitlichen Verwaltungsbereich.
Gruppendomäne	Definiert eine Gruppe von Objekten, die einheitlich behandelt werden sollen (beispielsweise einer einheitlichen Strategie folgen).
Typdomäne	Enthält Objekte gleichen Typs (beispielsweise alle Drucker einer Abteilung).

Tabelle 3: Beispiele für Domänen

Wie Abbildung 4 zeigt, spielt im Zusammenhang mit der Dienstunterstützung vor allem der Typ der Verwaltungsdomäne eine Rolle (vgl. auch Abschnitt 2.3.2): Analog zur Unterteilung der realen Welt repräsentiert eine Domäne hier z. B. eine Abteilung, eine Firma oder allgemein einen administrativ abgeschlossenen Bereich mit einer bestimmten Strategie. In Abhängigkeit von den in der Domäne enthaltenen Komponenten (nur Dienstnutzer, nur Diensterbringer oder beide) verhält sich der zugeordnete Trader entweder analog zur Einkaufs- oder Verkaufsabteilung einer Firma. Jedoch kann es auch innerhalb von Domänen einen offenen Markt von Diensten, d.h. unterschiedliche Dienstanbieter und Nutzer geben. Verwaltungsdomänen werden im Regelfall von einer einheitlichen Systemverwaltung verwaltet und gesteuert.

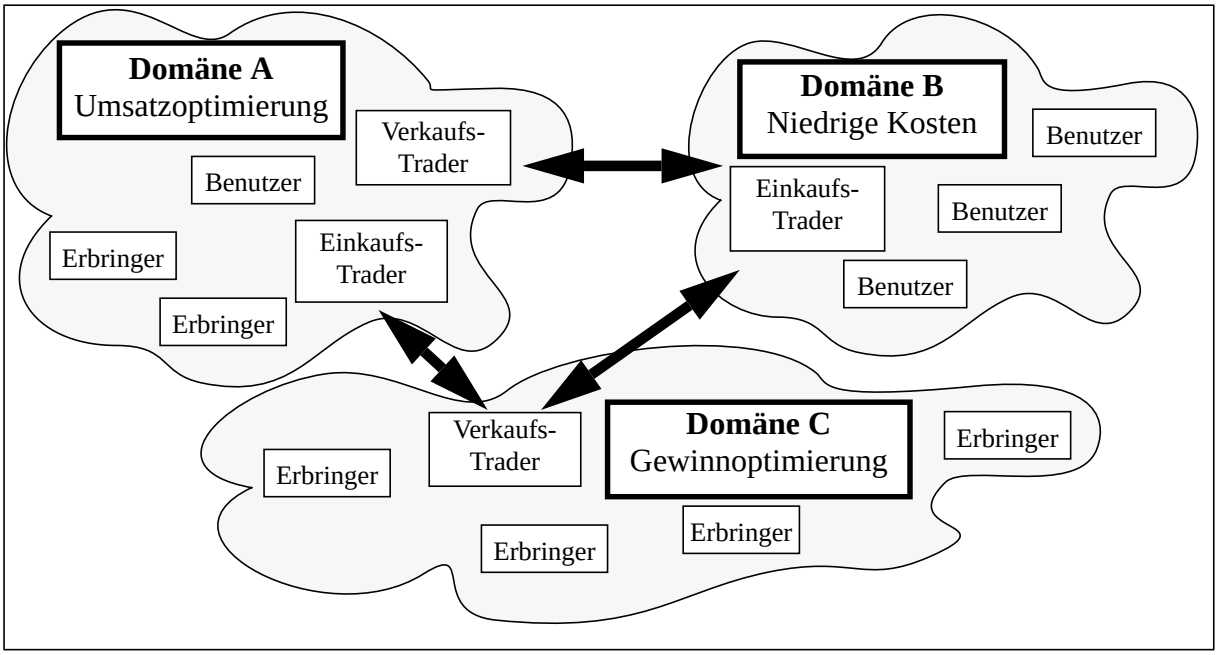


Abbildung 4. Domänen im Bereich der Dienstvermittlung

Domänen besitzen einen *Domänennamen*, mit dem sie eindeutig bezeichnet werden können. Die Objekte einer Domäne können innerhalb der Domäne einen eigenen Namen besitzen, den Domänenobjektnamen. Dieser muß innerhalb einer Domäne eindeutig sein. Durch die Konkatenation des eindeutigen Domänennamens mit dem Domänenobjektnamen können Domänenobjekte eindeutig bezeichnet werden. Auf einer Domäne sind verschiedene Operationen wie das Auflisten der Domänenmitglieder oder das dynamische Hinzufügen und Löschen von Domänenobjekten definiert.

Das Verhalten der Domänenobjekte wird durch *Domänenstrategien* bestimmt. Dementsprechend sind mit Domänen die folgenden Attribute verbunden:

- Definition der zu einer Domäne gehörenden Menge, entweder über eine Auflistung oder über die Eigenschaften derjenigen Komponenten, die in der Domäne enthalten sind.
- Definition von festen oder variablen, über Regeln definierten Domänenstrategien.
- Hinsichtlich der Anwendung von Domänenstrategien die Anzahl aktiver und passiver Komponenten.
- Beziehungen zwischen Domänen, z. B.mathematische Mengenbeziehungen oder auch die Sichtbarkeit und Kontrolle von Domänenobjekten in übergeordneten Domänen.

Zur weiteren Unterteilung können Domänen in Subdomänen aufgespalten werden. Dabei kann das Domänenkriterium für die Subdomänen sich von dem der Superdomäne unterscheiden. Die Domänennamen können hierarchisch strukturiert sein und dadurch die Struktur der Domänenhierarchie widerspiegeln. Eine wichtige Beziehung zwischen Domänen ist die Disjunktheit oder die Überlappung. Zwischen disjunkten Domänen können Kooperationsvereinbarungen getroffen werden, die eine exakte Regelung der gegenseitigen Dienstnutzung darstellen.

Im folgenden werden die Domänenkonzepte beim Trading genauer behandelt, auf die entsprechenden Konzepte im Managementbereich wird im Rahmen dieser Arbeit nicht eingegangen.

2.3.2 Domänenkonzepte im Trading

Im Bereich der Standardisierung ([ISO92], [ISO93] , [ISO94] und ISO95) wird der Begriff der Domänen direkt im Zusammenhang mit Trading verwendet, wobei in Abhängigkeit von den ODP viewpoints unterschiedliche Aspekte zur Domänenbildung betrachtet werden (vgl. Tabelle A.14). So führen der enterprise und der computational viewpoint zu einem Konzept disjunkter Domänen, in dem eine aktive Komponente, der Trader, eine Menge passiver Komponenten wie Dienstanbieter und -nutzer unter Verwendung bestimmter Strategien verwaltet. Zusammen mit administrativen Komponenten wird ein solches Konglomerat seit der Version [ISO93] als *trading community* bezeichnet. Ab Version [ISO94] können sich solche Gemeinschaften aus mehreren Domänen zusammensetzen, die sich jeweils auf unterschiedliche Aspekte wie z. B. Sicherheit oder Abrechnungsregelung beziehen. Außerdem wird der Begriff des *enterprise* definiert, der die Zusammenfassung mehrerer Gemeinschaften darstellt.

Da jeder einzelnen solchen Trading-Gemeinschaft vollständige Autonomie in allen traderspezifischen Aspekten (z.B. Definition von Diensttypen) zugesprochen wird, können beim Informationsaustausch zwischen unterschiedlichen Gemeinschaften Verständigungsprobleme auftreten. In diesem Fall sorgen die im Rahmen einer Zusammenarbeit vorgesehenen Verhandlungen für den Einsatz geeigneter Abbildungsmechanismen (vgl. Abschnitt 3.3 und 5.3).

Neben dem eben beschriebenen Domänenkonzept wurde ein weiteres eingeführt, das aus dem information viewpoint abgeleitet wurde und zu überlappenden Domänen führt. Dort umfaßt eine Domäne die Menge der durch Trading verfügbaren Dienste und enthält auch Dienste, die erst über eine Zusammenarbeit zwischen unterschiedlichen Tradern und ihren Domänen verfügbar werden. Derartige Domänen können intern weiter strukturiert werden über sogenannte Kontexte[1] (vgl. auch [BeRa91]). Kontexte können sowohl von Dienstnutzern als auch von Dienstanbietern und Tradern definiert werden und beinhalten wiederum jeweils eine

1. In ISO95 wird dieses Konzept durch die als *partitions* bezeichneten Abbildungen ersetzt, die auch zwischen Tradern existierende, sogenannte *links* einschließen.

bestimmte Strategie. Durch die unterschiedliche Gestaltung von Kontexten und die dadurch bei der Zusammenarbeit entstehende Heterogenitätsproblematik werden wiederum Abbildungsmechanismen benötigt.

2.3.3 Strategien zur Steuerung verteilter Systeme

Strategien (englisch Policies) definieren auf mehr oder minder abstraktem Niveau einheitliche Ziele für eine bestimmte, aus einem oder mehreren Subjekten bestehende Menge. Eine solche Menge kann z. B. durch eine Domäne definiert werden. *Strategieziele* beschreiben Objektzustände, die erreicht oder eingehalten werden müssen, machen Vorgaben über bestimmte Verhaltensweisen, o. ä.. Allgemein lassen sich Strategien gemäß [Wies94] in Verbote (*prohibitions*), erlaubte Möglichkeiten (*permissions*) und vorgeschriebene Aktionen (*obligations*) einteilen.

Aus einer Strategie ergeben sich *Handlungsweisen*, mit denen eine Strategie erfüllt wird. Diese müssen dann jeweils durch die Subjekte auf der Basis der zur Verfügung stehenden Systemmechanismen umgesetzt werden. In Abhängigkeit vom Abstraktionsniveau einer vorgegebenen Strategie ist die Ableitung der zugehörigen Handlungsweisen und verwendeten Systemmechanismen unterschiedlich komplex und in der Regel nicht automatisch ausführbar. Über eine Folge von Strategie-Verfeinerungsschritten (*Policy Refinement*) werden abstrakte Ziele auf konkrete Handlungsanweisungen abgebildet. Eine semi-automatische Umsetzung von Strategien ist dann möglich, wenn die zum Erlangen des Zieles benötigten Aktionen direkt definiert werden können. Häufig besteht die Strategiedefinition dann aus einer Auswahl aus einer Menge vorgegebener Strategien, eventuell verbunden mit der Angabe von *Strategieparametern*. Wie bereits erwähnt, kann durch Strategien in Kombination mit Domänen das Verhalten einer großen Menge von Domänenmitgliedern definiert werden.

Weiterführende Ansätze zur Realisierung von Strategien und zugehörigen Systemmechanismen sind in [Mars93], [MoSl91b] und [SMTK93] zu finden. Im folgenden werden Strategien für Trader genauer behandelt, auf die entsprechenden Konzepte im Managementbereich wird im Rahmen dieser Arbeit nicht eingegangen.

2.3.4 Trader-Strategien

Strategien von Tradern können zum einen die Dienstgüte des Traders betreffen, wozu nach [ISO92] die Parameter Verfügbarkeit, Realzeitanforderungen und eine möglichst gute Konsistenz zwischen dem Angebot eines Dienstleisters und seiner tatsächlichen Einsetzbarkeit zählen. Weitergehende Strategien, die sich auf alle beteiligten Komponententypen beziehen, sind seit der Version [ISO93] zu finden. Dort werden Strategien dazu verwendet, die Trading-Aktivitäten innerhalb einer Trading Community zu regeln. Dementsprechend werden Strategien definiert, die den Ablauf eines Dienstangebotes (Export), einer Dienstnutzung (Import) oder das Trader-Verhalten allgemein regeln. Z. B. kann durch eine Import-Strategie die Menge derjenigen Dienstanbieter eingeschränkt werden, die bei einer Dienstanfrage betrachtet werden sollen (Suchraum und -ordnung, vgl. Abschnitt 4.3.2). Trader-Strategien bestehen aus Regeln

- zur Behandlung von Sicherheitsaspekten,
- zur Annahme/Ablehnung und Gültigkeit von Dienstangeboten und Dienstanfragen,
- zur Spezifikation von Typen, ihrer Speicherung und der Suche nach ihnen,
- zur Behandlung von Konflikten bei einander widersprechenden Strategien und

- zur Abrechnung von Trader-Leistungen.

Seit der Version [ISO94] wird außerdem unterschieden zwischen *consult* und *unify policy*, die die Abfolge von Trader-Aktionen bzw. die Kombination von Strategien regeln.

Außerdem können Strategien zur Zusammenarbeit zwischen Tradern definiert werden, die Aktivitäten zwischen unterschiedlichen Domänen regeln und z. B. aus einer Liste erlaubter und verbotener Aktivitäten bestehen. Durch Strategien zur Zusammenarbeit ist außerdem festzulegen, in welchen Fällen eine Zusammenarbeit überhaupt initiiert werden soll. Im Zusammenhang mit dem Implementierungsentwurf in [ISO92] und in [NiGo94] wurde zunächst einmal vorgesehen, bei jeder nicht vermittelbaren Kundenanfrage eine Zusammenarbeit ohne jegliche Einschränkungen und Verbote zu initiieren. Die allgemeinere Strategie, Kunden ein möglichst breites Spektrum von Diensten und Dienstanbietern einen möglichst großen Kundenkreis verfügbar zu machen, wurde bisher noch nicht ausführlich behandelt.

Insgesamt läßt sich feststellen, daß Strategien allgemein und insbesondere Strategien zur Zusammenarbeit noch weiterer Überlegungen bedürfen, um zu sinnvollen Konzepten und Realisierungen zu kommen. Denkbar ist u.a. die Aufnahme von Leistungs- und Kostenkriterien in die entsprechende Strategie (vgl. Kapitel 5, [Burg90] sowie [TWH90], [WoTs90] und [WoTs91]).

2.4 Zusammenfassung

Im vorangegangenen Kapitel wurde das im MELODY-Projekt verwendete Systemmodell definiert. Dazu wurden die beteiligten Komponenten (Diensterbringer, Dienstnutzer und Systemverwalter) eingeführt und auf der Basis eines allgemeinen Kooperationsmodells hinsichtlich ihres Kooperationsbedarfs modelliert. Neben der Modellierung der einzelnen Komponenten wurde auch eine Strukturierung des Systems mit Hilfe von Domänen eingeführt. Domänen wurden dabei mit für die jeweiligen Domänen spezifischen Strategien verknüpft, um so unterschiedliche Verhaltensweisen für Mengen von Objekten zu definieren. Auch hier wurde ein allgemeines Modell definiert und dann auf das spezielle Szenario (Kooperation zwischen verschiedenen Trader-Domänen) angewendet. Auf der Basis des eingeführten Modells können nun im folgenden Kapitel einzelne Aspekte der Interaktion zwischen den Komponenten des Modells untersucht werden.

3. Interaktionen im Modell

Im folgenden werden Interaktionen zwischen den im letzten Kapitel beschriebenen Komponenten näher betrachtet. Die Schwerpunkte liegen dabei auf der Interaktion zwischen Dienstnutzern und Tradern (Abschnitt 3.1), auf der Interaktion zwischen Management- und Trading-Komponenten (Abschnitt 3.2) und auf den bereits verfügbaren Verfahren zur Interaktion zwischen Tradern (Abschnitt 3.3). Wegen der Bedeutung der Zusammenarbeit zwischen Tradern wird auf diesen Aspekt in Kapitel 5 noch einmal ausführlicher eingegangen, wenn die in Kapitel 4 beschriebenen Varianten des Trader-Modells zur Verfügung stehen.

3.1 Interaktion Dienstnutzer-Trader für unterschiedliche Nutzungstypen

Um die Interaktion zwischen Dienstnutzern und Tradern behandeln zu können, werden im folgenden die unterschiedlichen Formen der Dienstnutzung klassifiziert und die daraus resultierenden, unterschiedlichen Szenarien für einen Trader-Einsatz beschrieben. Dienstnutzungen lassen sich zunächst danach einteilen, ob der Diensterbringer für den Dienstnutzer *sichtbar* ist und direkt angesprochen wird (Abschnitt 3.1.1) oder ob er durch den Trader verborgen und damit *transparent* genutzt wird (Abschnitt 3.1.5).

Ein weiteres Klassifikationskriterium betrifft die *Verkettung* von Dienstnutzungen. Dienste können entweder *isoliert* (Abschnitt 3.1.6) oder durch eine Serie von Dienstaufrufen (Requests) genutzt werden, bei der zwischen den einzelnen Dienstaufrufen Information beim Server gespeichert wird[1] (Abschnitt 3.1.7). Derartige Informationen beeinflussen die Ergebnisse des jeweils folgenden Dienstaufrufs. Eine solche Folge von Dienstaufrufen, die über bestimmte Informationen verkettet sind, wird auch als *Sitzung* bezeichnet. Sitzungen können weiter strukturiert sein und ausgezeichnete Punkte enthalten, an denen Teilsitzungen beginnen.

3.1.1 Sichtbare Dienstnutzung

Im Fall einer sichtbaren Dienstnutzung fragt der Dienstnutzer beim Trader an, um einen Diensterbringer vermittelt zu bekommen, und wendet sich anschließend selbst an den Diensterbringer, um die Dienstleistung bei ihm anzufordern. Ein einfaches Beispiel dafür ist das sogenannte Klient-Server-Modell in verteilten Systemen. Falls der Klient den Server oder dessen Adresse nicht a priori kennt, kann er diese Angaben vom Trader erfragen. Voraussetzung dafür ist, daß der Server sein Dienstangebot zu einem früheren Zeitpunkt an den Trader gemeldet hat.

Sichtbare Dienstnutzungen lassen sich danach unterscheiden, inwieweit der Dienstnutzer den Vermittlungsgegenstand beschreiben und eingrenzen kann. Ist der gewünschte Diensterbringer

1. Isolierte Dienstnutzungen werden häufig auch als kontextfrei und dementsprechend untereinander verkettete Dienstnutzungen als kontext-sensitiv bezeichnet. Der Begriff des „Kontextes“ wird im Rahmen dieser Arbeit jedoch noch mit anderer Bedeutung verwendet (vgl. Abschnitt 2.3.2 und 4.2.4).

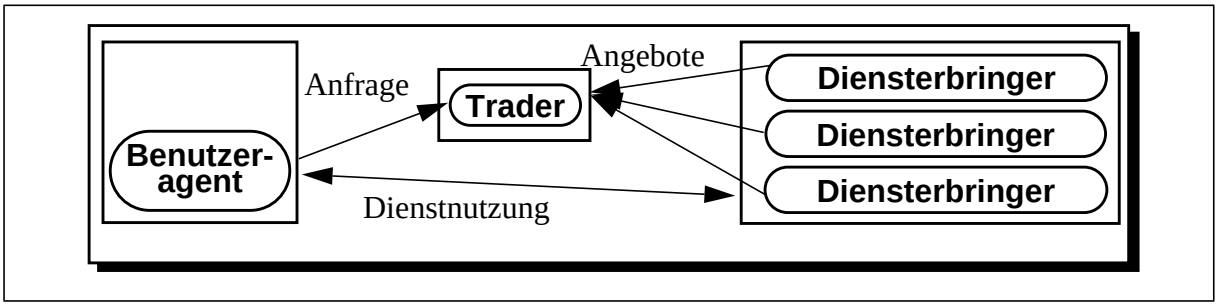


Abbildung 5. Sichtbare Dienstnutzung

bereits bekannt, so kann er schon soweit beschrieben werden, daß nur noch sein Zugangspunkt vom Trader zurückzuliefern ist (Abschnitt 3.1.2). Andernfalls wird durch Angabe eines Diensttyps erst ein Diensterbringer gesucht, der die entsprechende Dienstleistung erbringt (Abschnitt 3.1.3). Analoges gilt für die Anfrage nach einem Datenobjekt (Abschnitt 3.1.4), welches irgendwo im System vorhanden ist und von einem bestimmten Dienst verwaltet wird. In diesem Fall ist wiederum der Zugangspunkt des verwaltenden Diensterbringers von Interesse.

3.1.2 Server-orientierte Dienstnutzung

Beim server-orientierten Dienst ist die vom Benutzer benötigte Dienstleistung eng mit der Identität des Diensterbringers verknüpft, d. h. es wird lediglich der Zugangspunkt gesucht. Zum Beispiel kann die Identität eines Diensterbringers, der bestimmte Werkzeugmaschinen steuert, bekannt sein, sein Zugang kann jedoch wechseln und ist daher jedesmal neu abzufragen. Die für eine derartige Ortstransparenz benötigte Vermittlungsfunktionalität kann bereits durch einen Namens- oder Lokationsdienst erbracht werden.

Name	Erläuterung	Begründung
Mail-Delivery-Server	Liefert eine E-Mail an den Benutzer aus	Server repräsentiert den Benutzer.
Maschinen-Steuerung	Server steuert eine Maschine	Server repräsentiert die Maschine
Management-Agent	Agent ermöglicht den Zugang zur Managementinformation eines Netzknotens	Agent präsentiert den Knoten

Tabelle 4: Beispiele für server-orientierte Dienste

Ein solches Szenario wird verwendet, wenn der Dienstnutzer den gesuchten Diensterbringer fast vollständig beschreiben kann und explizit auf die Auswahl des Dienstes Einfluß nehmen möchte. Häufig repräsentiert der Diensterbringer einen wichtigen Aspekt des Systems - beispielsweise einen Anlaufpunkt für die Kommunikationsverbindung zu einem anderen Menschen oder den Agenten einer physischen Ressource (beispielsweise Maschinen in einer Fabrik). Beispiele für server-orientierte Dienstleistungen sind in Tabelle 4 zu finden.

3.1.3 Service-orientierte Dienstnutzung

Beim service-orientierten Dienst fordert der Dienstnutzer unter Angabe eines Diensttyps und eventuellen Auswahlkriterien eine Dienstleistung (Service) an, ohne explizit einen Diensterbringer zu nennen. Der Trader übernimmt dann einerseits die Vermittlung eines geeigneten Dienstes. Zur Berücksichtigung der Auswahlkriterien muß dabei möglicherweise zwischen mehreren gleichartigen Diensterbringern entschieden werden. Hierbei kann außer den Eigenschaften der Diensterbringer auch der Zustand ihrer Umgebung eine Rolle spielen (z.B. Belastung des Prozessors oder der Kommunikationsverbindung zwischen Dienstnutzer und potentiellem Erbringer). Andererseits fungiert der Trader jedoch auch als Lokator, d.h. als eine Instanz, die den geeigneten Dienstzugangspunkt ermittelt. Eine derartige Vermittlung wird unter anderem auch bei verteilten Diensten benötigt. Tabelle 5 zeigt Beispiele für service-orientierte Dienste.

Dienst	Erläuterung
E-Mail	Der E-Mail Dienst ist auf jedem Rechner vorhanden. Bei der Diensterbringung spielen verschiedene Dienstprozesse zusammen. Der lokale Benutzeragent ist so konfiguriert, daß er mit beliebigen Dienstprozessen zusammenarbeiten kann.
Zeitgeberdienste	Der Dienstnutzer erwartet eine akkurate Zeit, ohne daß ihn die Identität des Zeitgebers interessiert.
Namenssystem	Replizierte Nameserver sorgen für einheitlichen Zugang zum Namenssystem.

Tabelle 5: Beispiele für service-orientierte Dienste

3.1.4 Objekt-zentrierte Dienstnutzung

Objekt-zentrierte Dienste greifen auf Objekte zu, die irgendwo im Netz von einem Diensterbringer verwaltet werden. Die Operationen erfolgen nicht auf einem abstrakten Service oder auf einem konkreten Diensterbringer, sondern auf dem Datenobjekt, welches durch einen Diensterbringer verwaltet wird. Eine solche Vorgehensweise ermöglicht einen ortstransparenten (respektiv server-transparenten) Zugriff auf die entsprechenden Datenobjekte. Die Kommunikation erfolgt natürlich mit dem Diensterbringer, der das Datenobjekt hält. Dieser ändert sich nur für den Fall, daß das Datenobjekt migriert. Eine solche Dienstleistung wird dann i.a. nur verkettet genutzt, wobei das Datenobjekt selbst einen großen Teil der Sitzungsinformation beinhaltet (vgl Abschnitt 3.1.7).

Ein Spezialfall ergibt sich bei replizierten Datenobjekten, die bei mehreren unterschiedlichen Diensterbringern liegen und auf die der Zugriff bei jedem Diensterbringer erfolgen kann. In diesem Fall kann in der laufenden Sitzung zwischen verschiedenen Diensterbringern umgeschaltet werden (sofern die entsprechende Implementierung dies erlaubt), wobei die Datenobjekte auf dem neuen Diensterbringer ebenfalls verfügbar sein müssen. Gründe für solche Umschaltungen sind auftretende Fehler oder sich abzeichnende Überlastungen bei einem Diensterbringer.

Um dabei Ortstransparenz zu erreichen, kann in einfachen Fällen ein Lokalisationsservice (z.B. eine Namensverwaltung) eingesetzt werden. Falls aber auf dynamische Attribute zuge-

griffen werden muß, um zur Laufzeit zu ermitteln, bei welchem Diensterbringer sich das gesuchte Objekt befindet, wird Trader-Funktionalität benötigt. Im Spezialfall der replizierten Diensterbringer kann ein Trader zusätzlich Lastbalancierung oder Systemoptimierungen als Kriterien für die Auswahl des konkreten Diensterbringers berücksichtigen.

Art des Servers	Erläuterung	Datenobjekte
Dokumentenverwaltung	Verwaltet Dokumente und erlaubt eine effiziente Suche	Dokumente und ihre Eigenschaften
Dateiverwaltung	Verwaltet Dateien und ermöglicht entfernten Zugriff	Dateien und Struktur des Dateisystems (Directories)
Datenbank-Server	Datenbestände	Daten (z.B. in relationalen Tabellen)

Tabelle 6: Objekt-zentrierte Dienste

Wie im Fall der service-orientierten Dienstnutzung muß der Trader bei derartigen Optimierungen möglicherweise zwischen mehreren gleichartigen Objekten entscheiden. Hierbei kann er über die Eigenschaften der Objekte hinausgehend den Zustand der Objektumgebung berücksichtigen. Die Trader-Funktionalität bezieht sich in dem Fall auf die Auswahl eines geeigneten Objekts anstelle eines geeigneten Services. Ansonsten unterscheidet sich der zu erbringende Trading-Dienst nur in der Art der verwalteten Typen (Objekttypen vs. Diensttypen), der Eigenschaften und des Zugriffs auf die Objekte, bzw. Dienste.

Beispiele für objekt-zentrierte Dienste sind in Tabelle 6 zu finden.

3.1.5 Transparente Dienstnutzung

Im Gegensatz zur sichtbaren Dienstnutzung sucht der Dienstnutzer im transparenten Fall nicht einen Diensterbringer, sondern fordert beim Trader eine bestimmte Dienstleistung an. Der Trader erbringt diese Dienstleistung stellvertretend durch die Auswahl eines geeigneten Diensterbringers, wickelt mit diesem die einzelnen Arbeitsschritte ab und liefert anschließend das Ergebnis zurück an den Dienstnutzer. Dem Dienstnutzer ist in diesem Fall die Identität des Diensterbringers nicht bekannt. Der Trader hat daher die Freiheit, den Diensterbringer beliebig zu wechseln, sofern dies transparent für den Dienstnutzer durchführbar ist. Dadurch kann neben einer höheren Fehlertoleranz auch eine Leistungssteigerung erzielt werden. Allerdings hat der Dienstnutzer praktisch keinerlei Möglichkeit zur Einflußnahme. Dieses Modell ist

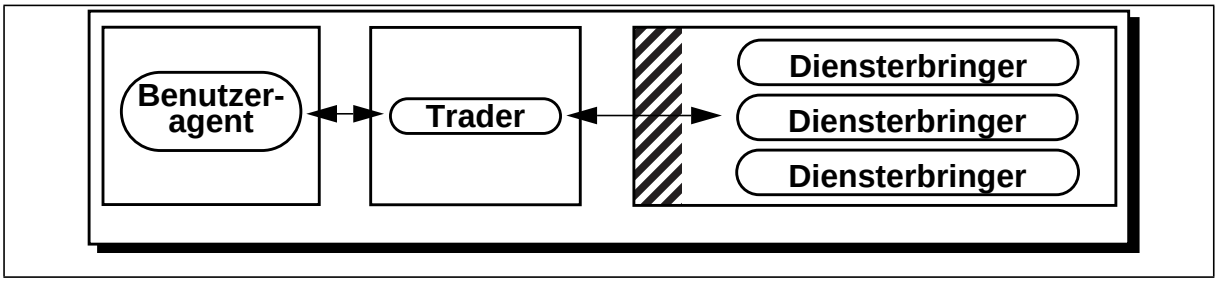


Abbildung 6. Transparente Dienstnutzung

daher auch nur für Fälle geeignet, in denen sich die Interaktion zwischen Dienstnutzer und -erbringer auf ein Minimum beschränkt.

3.1.6 Isolierte Dienstnutzung

Bei isolierten Dienstnutzungen wird beim Diensterbringer keine nutzungsabhängige Information verwaltet (vgl. die Beispiele in Tabelle 7). Dies bedeutet insbesondere, daß das Ergebnis der Dienstnutzung unabhängig vom augenblicklichen Zustand des Diensterbringers ist. Das hat den Vorteil, daß aufeinanderfolgende Operationen durch unterschiedliche Diensterbringer ausgeführt werden können. Ein einzelner Dienstaufruf kann im Fehlerfall einfach wiederholt werden. Für diese Klasse von Operationen ist der Einsatz von Tradern sehr einfach möglich, da im Prinzip jede einzelne Operation auf einem anderen Diensterbringer ausgeführt werden kann. Es gibt jedoch verschiedene Möglichkeiten, wann ein Trader eingesetzt wird.

Dienst	Erläuterung
Zeitgeber	Zur Uhrensynchronisation kann die lokale Zeit mit einem Zeitgeber verglichen werden. Da die Zeit keine Zustandsvariable eines Server darstellt, wird keine diesbezügliche Information auf dem Zeitgeber verwaltet.
Berechnungsdienst ohne permanenten Zwischenspeicher	Ein Berechnungsdienst wie z. B. ein Dienst zur Programmkompilation oder zur Bearbeitung von Simulationsprogrammen ist im allgemeinen nur von seiner Eingabe abhängig und seine Nutzungen daher isoliert.

Tabelle 7: Beispiele für isoliert genutzte Dienste

Strategie 1: Permanente Neuwahl

Mit Hilfe des Traders wird vor jedem Aufruf durch den Benutzeragenten ein neuer geeigneter Diensterbringer bestimmt.

Vorteil:

- Für jeden Auftrag wird ein neuer bester Diensterbringer gesucht. Dies kann vor allen Dingen bei langdauernden und rechenintensiven Prozessen von Vorteil sein, da sich die Entscheidungsgrundlage für die Auswahl eines Diensterbringer inzwischen geändert haben mag.
- Der Trader-Einsatz wird - bei geeigneter Auswahlstrategie - zur Lastbalancierung eingesetzt und führt zu einer gleichmäßigen Auslastung der vorhandenen Diensterbringer.
- Die Systemverwaltung kann mit Hilfe einer systemseitig vorgegebenen Auswahlstrategie das Verhalten des Systems beeinflussen. Wird jeder Auftrag mit Hilfe eines Traders bearbeitet und wird dabei die Strategie berücksichtigt, so reagiert das System schnell auf die neuen Anforderungen der Systemverwaltung.

Nachteile:

- Durch fortwährenden Einsatz des Traders erhöht sich die Laufzeit jedes Dienstaufrufs.

- Zusätzlicher Aufwand und Kosten durch die im Trader benötigten Kommunikationsvorgänge (Trader-Aufrufe, Zugriff auf dynamische Attribute).

Strategie 2: Bedingte Neuwahl

Der Trader wird jeweils nach dem Zutreffen einer bestimmten Bedingung neu eingeschaltet. Zu den möglichen Bedingungen gehören:

- Der Ablauf definierter Zeitscheiben,
- eine definierte Anzahl durchgeführter Aufträge,
- explizite Aufforderungen vom Diensterbringer,
- explizite Managementanweisungen an den Benutzeragenten,
- Überschreitung bestimmter Schwellwerte beim Diensterbringer und Rückmeldungen des Diensterbringers,
- fehlerhafte Aufrufe oder ausgefallene Diensterbringer.

Beim Eintreffen der entsprechenden Bedingung wird der Trader verwendet, um einen neuen Diensterbringer zu ermitteln. Die dabei getroffene Entscheidung gilt dann, bis die Bedingung das nächste Mal zutrifft. Man kann auch davon sprechen, daß zwischen einem Dienstnutzer und einem Diensterbringer eine Sitzung aufgebaut wird, die jedoch ohne sitzungsbezogene Information auskommt. Daher kann eine solche Sitzung beim Erreichen der Schedulingbedingung ohne zusätzlichen Aufwand zu einem anderen Diensterbringer migriert werden.

Vorteile:

- Verzögerungen durch den Einsatz des Traders werden anwendungsabhängig reduziert.
- Die Systemverwaltung hat durch die expliziten Anweisungen an den Benutzeragenten oder durch die indirekten Anweisungen vom Diensterbringer bessere Möglichkeiten, auf den Dienstnutzer einzuwirken.

Nachteile:

- Je nach Definition des Ereignisses (z.B. bei größeren Zeitperioden) kann es lange dauern, bis eine getroffene Entscheidung wieder überprüft wird.
- Der Benutzeragent muß die Überprüfung von Ereignissen durchführen.
- Die Systemverwaltung kann u. U. nicht wissen, wann ein Benutzeragent den Trader neu einsetzt. Eine Strategieänderung kann daher erst sehr viel später wirksam werden, wenn nicht zusätzliche Maßnahmen ergriffen werden.

Mit der Definition der Ereignisse werden Ziele verfolgt und Strategien realisiert. Beispielsweise kann ein Dienstnutzer durch einen häufigen Aufruf des Traders eine schnelle Anpassung an sich ändernde Systembedingungen erreichen. Welche Ziele jedoch verfolgt werden, hängt sehr stark von der Art des gesuchten Dienstes, der eigenen Dienstnutzung und den eigenen Erwartungen ab.

Strategie 3: Benutzergesteuerte Auswahl

Der Dienstnutzer kann die Fähigkeiten des Traders nutzen, um eine Menge von möglichen Diensten zu finden. Auch die Fähigkeit, eine Bewertung durch den Trader vornehmen zu lassen, kann von Vorteil sein (siehe [Walk94]). Nachdem die Auswahl vorgenommen wurde, wird dem Benutzeragenten der ausgewählte Diensterbringer übergeben und von diesem verwendet.

3.1.7 Verkettete Dienstnutzungen

Bei Diensten mit verketteter Nutzung hängt das Ergebnis einer bestimmten Dienstnutzung auch von sitzungsbezogener Information beim Diensterbringer ab. In der Regel ergibt sich daraus eine feste Bindung des Dienstnutzers an den Diensterbringer (in der Form der Sitzung) und der Aufbau von mit der Sitzung verbundener Information. Diese Information beinhaltet Zustandsinformation über den Dienstnutzer und über die bisherige Dienstnutzung. Ein Trader vermittelt daher i.a. einzelne Sitzungen.

Analog zur isolierten Dienstnutzung kann eine Dienstnutzerstrategie angeben, wann zwischen unterschiedlichen Sitzungen ein Trader neu verwendet wird. Während einer Sitzung ist es nur unter besonderen Bedingungen (migrationsfähige Sitzung und geeignete Diensterbringer) möglich, durch explizite Aktionen zu einem anderen Diensterbringer zu migrieren. Dies beinhaltet vor allen Dingen die Übertragung der sitzungsbezogenen Information von einem Diensterbringer auf den anderen. Bietet ein Dienst diese Möglichkeit, so kann durch eine Verwendung des Traders das Ziel für eine Sitzungsmigration bestimmt werden.

Ein weiterer Spezialfall ergibt sich, wenn eine Sitzung nur zu bestimmten Zeitpunkten migriert werden kann. Dies ist z.B. dann der Fall, wenn zu einem solchen Zeitpunkt wenig sitzungsbezogene Information vorliegt und daher sehr einfach migriert werden kann.

Eine Trader-Nutzung kann möglich sein:

- Am Anfang einer Sitzung,
- an ausgezeichneten Punkten während der Sitzung,
- bei migrationsfähigen Sitzungen zu jedem beliebigen Zeitpunkt.

Analog wie bei der isolierten Dienstnutzung können Dienstnutzerstrategien (permanente Neuwahl, bedingte Neuwahl, benutzergesteuerte Auswahl) den Trader-Einsatz genauer festlegen, z.B. indem immer am Anfang einer Sitzung, immer an ausgezeichneten Punkten einer Sitzung oder nach jeder Migrationsentscheidung beim Trader angefragt wird.

3.2 Interaktion zwischen Trading und Management

Die Integration eines Trading- und Managementsystems kann auf verschiedene Weisen erfolgen. Im folgenden werden unterschiedliche Ansätze diskutiert.

Vollständige Integration
Bei der vollständigen Integration wird das Trading als eine spezielle Managementfunktion aufgefaßt, woraufhin Interaktionen mit dem Trader mit Hilfe von Managementprotokollen erfolgen. Dieser Ansatz hat den Vorteil, daß sowohl im Trader als auch im Managementsystem ein einheitliches Informationsmodell verwendet wird. Es sind daher keine aufwendigen Abbildungsschritte notwendig. Entwicklungsplattformen für Managementfunktionen (Management Frameworks) nach dem OSI- ([HeAb93]) oder CORBA-Standard ([OMG91]) erlauben eine vollständige Abbildung der Trader-Schnittstelle auf die Schnittstelle eines Managementobjektes. Es muß sich jedoch zeigen, ob eine solche Lösung dem Trader-Problem angemessen ist. Weiterhin erzwingt diese Lösung die vollständige Ablösung bestehender Trading-Systeme und den Ersatz durch entsprechende Trader-Managementobjekte.

Hierarchische Integration
Der Trader bildet eine Anwendung, die auf Managementinformation zugreift. Diese Lösung hält das Managementsystem unabhängig vom Trader. Es besitzt eine einfache und klare Struk-

tur. Der Trader muß das Informationsmodell des Managementsystems auf das eigene Informationsmodell abbilden. Die Schnittstelle des Managementsystems muß derart gestaltet werden, daß dem Trader die benötigte Managementinformation effizient zur Verfügung gestellt wird. Aus der Sicht des Traders ist dies eine ausreichende Lösung. Jedoch werden hierbei die Möglichkeiten des Traders zur Berücksichtigung der Wünsche des Systemverwalters bei der Vermittlung nicht ausgenutzt. Dazu müßte das Management den Vermittlungsprozeß beeinflussen und ihn selbst nutzen können, was bei einer strikt hierarchischen Integration nicht möglich ist.

Gleichberechtigte Integration
Bei der gleichberechtigten Integration stellen beide Systeme Schnittstellen zur Verfügung, über die das jeweils andere zugreifen kann. So kann beispielsweise der Trader Fehlerinformation des Managementsystems anfordern, um möglichst keine nicht erreichbaren Dienste zu vermitteln. Im umgekehrten Fall kann das Managementsystem den Vermittlungsprozeß durch das Einbringen der eigenen Wünsche (z.B. in der Form von Regeln)) beeinflussen. Eine weitere Möglichkeit ist es, die Trader-Fähigkeiten zu nutzen, um für Managementfunktionen im System eine geeignete Ausführungsumgebung zu ermitteln. Bei der wechselseitigen Nutzung muß jedoch darauf geachtet werden, daß keine Verklemmungen entstehen.

Bewertung
Die vollständige Integration vereinigt beide Systeme auf der Basis gemeinsamer Kommunikationsprotokolle und eines einheitlichen Informationsmodells. Auf Grund der bereits bestehenden getrennten Entwicklungen wird dies in der Praxis selten der Fall sein. Die hierarchische Integration ist ein einfaches Modell, welches für den Trader durchaus ausreichend sein mag. Es berücksichtigt jedoch die Anforderungen des Management nur unzureichend. Eine gleichberechtigte Integration birgt die Problematik, daß die Interaktionen zwischen den Systemen sehr sorgfältig entworfen werden müssen, um Verklemmungen zu vermeiden. Es ergeben sich aber viele Möglichkeiten, die Ansätze beider Gebiete gemeinsam zu nutzen.

3.3 Zusammenarbeit zwischen Tradern

In einem verteilten System überschaubarer Größe reicht möglicherweise eine einzige Trader-Instanz zur Dienstvermittlung aus. Aber schon in diesem Fall empfiehlt sich die Verwendung mehrerer Instanzen zur Vermeidung von Engpässen und zur Erhöhung der Fehlertoleranz. Unvermeidbar sind mehrere Trader-Instanzen in verteilten Systemen mit einer großen Anzahl von Komponenten oder bei Anwendungsbearbeitungen, die sich über organisatorische und legislative Grenzen erstrecken können. Dabei ist der Fall nicht auszuschließen, daß sich Trader untereinander abstimmen müssen. Außerdem kann die Dienstvermittlung sowohl quantitativ als auch qualitativ wesentlich verbessert werden, wenn mehrere Trader-Instanzen sich mit ihren jeweils nur eingeschränkten Fähigkeiten gegenseitig ergänzen (vgl. auch die auf bestimmte Anwendungen spezialisierten Trader gemäß [MaBl92]). Aus diesen Gründen sind neben den in den letzten Abschnitten beschriebenen Interaktionen auch die zwischen Tradern zu betrachten.

Die bereits in der Literatur behandelten Interaktionen zwischen Tradern lassen sich nach folgenden Aspekten klassifizieren:

- Verbesserung quantitativer Aspekte durch Föderationen oder das sogenannte Interworking zwischen Tradern (vgl. Tabelle A.11).

- Verbesserung qualitativer Aspekte wie z.B. Verfahren, die durch Zuordnung von Benutzeranforderungen zu Servern für eine Leistungsoptimierung sorgen und bei denen zwischen den zuordnenden Instanzen nicht nur Zustandsinformation sondern Zwischenergebnisse der optimierenden Lösung ausgetauscht werden (vgl. den Überblick in [Burg90]). Diese lassen sich ohne weiteres in Trader integrieren (vgl. [TWH90], [WoTs90] und [WoTs91]).

In beiden Fällen ist eine Klassifikation der Interaktion zwischen den Tradern nach folgenden Kriterien möglich: Verteilung der Initiative (vgl. Tabelle A.12), Wissen über andere Trader, Heterogenität der beteiligten Trader, mögliche Operationen und ihre Auswirkungen, Objekte, auf die sich die Operationen beziehen können, Umfang einer Zugriffserlaubnis, zusätzliche Parameter und Möglichkeit der Abstimmung.

Für das Wissen über andere Trader sind die folgenden Modelle denkbar:

- Jeder Trader kennt alle anderen Trader im System (vgl. Implementierungsvorschlag in [ISO92]) entweder über eine eigene Datenbank oder mit Hilfe eines Namensdienstes.
- Jeder Trader kennt eine Teilmenge aller Trader (z. B. seine Nachbarn in räumlicher Beziehung). Dieses Modell entspricht den *links* und den dadurch entstehenden *Trading-Graphen* in [ISO94] und [ISO95] (vgl. auch die Konzepte zur Favorisierung von Partnern in [LeBe95]). Dabei ist zu gewährleisten, daß die Vereinigung aller dieser Teilmengen mit der Menge der kooperationswilligen Trader übereinstimmt, um keinen Trader zwangsweise zu isolieren. In diesem Modell sind auch Navigier- und Lernvorgänge möglich (vgl. die *search paths* in [ISO94], [ISO95] und [VBB95]).
- Trader greifen zur Vermittlung von geeigneten Kooperationspartnern auf einen speziell dafür ausgerichteten Trader-Trading-Dienst zu. Da auch die einzelnen Trader dieses Dienstes kooperieren können, entsteht eine rekursive Verkettung von Trading-Diensten. Diese kann nur durch Verwendung von Trader-Wissen der beiden zuerst genannten Typen beendet werden.

Um eine Zusammenarbeit zwischen heterogenen Tradern zu ermöglichen, sind bestimmte Vorkehrungen zu treffen. Gemäß [BeRa91] können sich Trader sowohl in ihrer Funktionalität als auch in ihrer Qualität unterscheiden (z.B. Existenz von Mechanismen für Zugriffs- und Konkurrenzkontrolle oder Recovery, unterschiedliche Directory-Hierarchien, Behandlung optionaler Attribute oder der Semantik von Diensten und Attributen, insbesondere unterschiedliche Typsysteme). Um trotz der Unterschiede zusammenarbeiten zu können, sind die verwendeten Verfahren und eventuelle Umsetzungen auszuhandeln. Solche Verhandlungen bestehen entweder nur aus einer Anfrage und Annahme/Ablehnung derselben oder sie lassen Gegenvorschläge zu und können damit aus mehreren Stufen bestehen. Eine Implementierung einer solchen, auch als Föderation bezeichneten Kooperation ist in [dPLJMa95] beschrieben.

Tabelle A.17 zeigt die bisher zwischen Tradern verwendeten Operationen. Dabei spielen vor allem EXPORT und IMPORT eine wichtige Rolle, mit denen eine Zugriffserlaubnis auf bestimmte Objekte zwischen Instanzen übertragen wird (im Gegensatz zur Semantik dieser Operationen zwischen Server - Trader bzw. Klient - Trader). Dieser Zugriff kann sich auf genau ein Objekt oder auf eine ganze Menge von Objekten und die darauf zugelassenen Aktionen beziehen (vgl. Tabelle A.15) und wird für ein bestimmtes Zeitintervall oder auf unbestimmte Dauer gewährt.

Ein weiteres Klassifikationskriterium betrifft die Anzahl der an einer Verhandlung beteiligten Trader. In bisherigen Ansätzen stand der Fall von zwei Kooperationspartnern im Vordergrund, während der von mehreren noch nicht ausführlich behandelt wurde. Lediglich das als *chaining* bezeichnete Verfahren in [BeRa91], mit dem Import- und Export-Operationen auf mehrere

Instanzen ausgedehnt werden können, resultiert in Kooperationsteams mit mehr als zwei Partnern, wobei jedoch immer nur zwischen zwei aufeinanderfolgenden Kettenmitgliedern eine direkte Beziehung besteht. Demgegenüber ist in [ISO95] die Verwaltung größerer Trader-Teams vorgesehen, jedoch noch nicht detaillierter definiert.

3.4 Zusammenfassung

Dieses Kapitel beschäftigte sich einerseits mit den unterschiedlichen Aspekten der Integration des Tradings in ein verteiltes System. Ausgehend von einer Klassifikation der Typen von Dienstnutzungen (sichtbar, transparent, isoliert, verkettet, auf unterschiedlicher Abstraktionsebene) wurden unterschiedliche Einsatzmöglichkeiten diskutiert. Hierbei hat ein Trader jeweils andere Aufgaben zu erfüllen. Im zweiten Abschnitt wurden verschiedene Möglichkeiten der Interaktion zwischen Trading- und Managementkomponenten untersucht und im dritten wurden bisherige Ansätze zur Interaktion zwischen Tradern klassifiziert.

Auf dieser Grundlage können im nächsten Kapitel unterschiedliche Aspekte eines Traders untersucht werden und damit ein allgemeines Trader-Modell definiert werden.

4. Varianten des Trader-Modells

Das folgende Kapitel diskutiert gebräuchliche Trader-Modelle und klassifiziert unterschiedliche Trader-Varianten aus der Literatur anhand eines verallgemeinerten Modells (vgl. die Tabellen in Anhang A). Abgesehen von der gemeinsamen Grundaufgabe unterscheiden sich Trader vor allem durch die unterschiedliche Mächtigkeit bei der Lösung der eigentlichen Dienstvermittlung. Diese Überlegungen führen zu einer Reihe von Verallgemeinerungen und Verfeinerungen bisheriger Trader-Modelle. So wurde anstelle des herkömmlichen Dienstangebots an den Trader die allgemeinere Dienstakquisitionsphase eingeführt, in der ein Trader die Datenbank der von ihm vermittelbaren Dienste u.a. auch durch aktives Absuchen eines verteilten Systems aufbauen kann. Außerdem wird die Vergleichbarkeit von Diensten über das bereits bekannte Kompatibilitätskriterium hinaus ausführlich untersucht. Dieser Punkt spielt insofern eine besondere Rolle, als es bereits im traditionellen Markt teilweise sehr schwierig ist, unterschiedliche Dienstleistungen vergleichend zu beurteilen. Im Gegensatz zu anderen Ansätzen im Trading-Bereich werden im folgenden auch dynamische Attribute ausführlich behandelt.

4.1 Das neue Trader-Grundmodell

Ein Trader besitzt Konfigurationsinformation über die von ihm verwaltete *Trader-Domäne*. Die Trader-Domäne umfaßt alle Dienstanbieter, die diesem Trader zugeordnet sind. Mit Hilfe dieser Information vermittelt er Dienstnutzern einen geeigneten Dienstanbieter, wobei Anforderungen aller Beteiligten, d.h. des Nutzers, Anbieters und der Systemverwaltung berücksichtigt werden können. Der Systemverwalter kontrolliert den Trader und nutzt dessen Möglichkeiten für Zwecke der Systemverwaltung. Abbildung 7 illustriert die einzelnen Interaktionen.

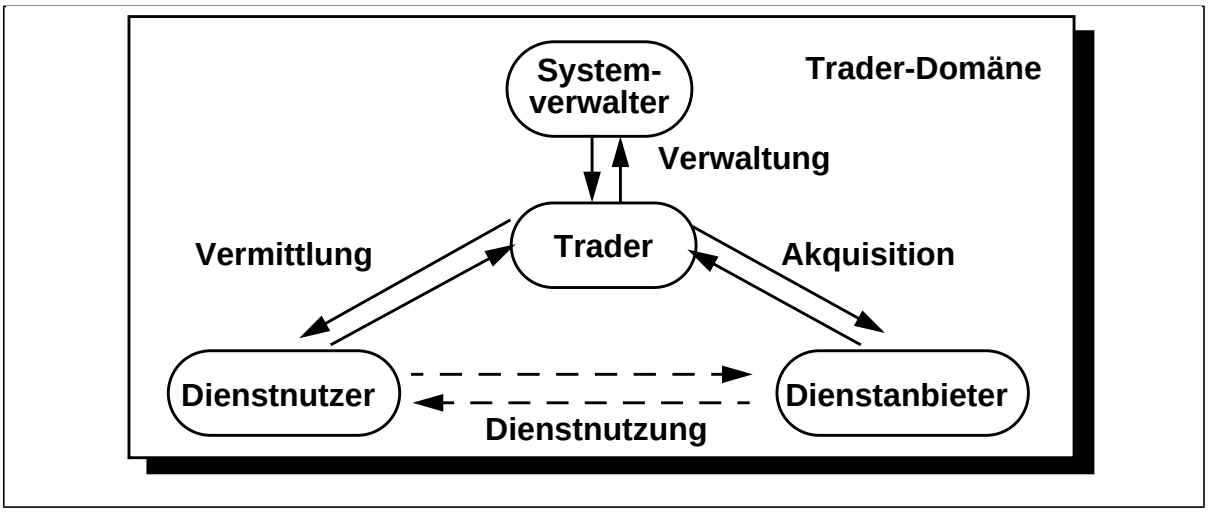


Abbildung 7. Das Trader-Grundmodell

Ein Trader akquiriert eine Reihe von *Dienstbeschreibungen* seiner Trader-Domäne und ordnet diese in seiner *Angebotsverwaltung* ein. Bei der Dienstvermittlung spezifiziert der Dienstnutzer in einer *Dienstanfrage* den gewünschten Dienst, wobei er Suchbereich, Einschränkungen und Kriterien für einen optimalen Dienst angeben kann. Aufgabe des Traders ist es, diese Dienstanfrage mit den Dienstbeschreibungen zu vergleichen und einen geeigneten Dienst zu finden. Das Ergebnis der Vermittlung ist dabei eine Angabe über den Diensterbringer, wobei es aus der Sicht des Traders keinen Unterschied macht, ob er dem Dienstnutzer lediglich die Identität eines geeigneten Diensterbringers mitteilt, einen Zugangspunkt ermittelt, ihm Auskunft über bestimmte Diensteigenschaften gibt oder unter einer Reihe von möglichen Zugangspunkten bereits eine Auswahl trifft. Der Vermittlungsprozeß umfaßt darüberhinaus eine Reihe von zusätzlichen Aufgaben wie die Überprüfung von kompatiblen oder ähnlichen Diensttypen, die Ermittlung von aktuellen Werten einer Diensteigenschaft und die Verhandlung von garantierten Dienstqualitäten.

Die folgenden Abschnitte untersuchen die Dienstakquisition mitsamt der Dienstbeschreibung und der Angebotsverwaltung (4.2), die Dienstvermittlung (4.3), den Einsatz dynamischer Attribute (4.4), die Ergebnisse des Trading-Dienstes (4.5) und potentielle Strategien eines Trading-Dienstes (4.6).

4.2 Dienstakquisition und Vergleichbarkeit von Diensten

Bei der Dienstakquisition wird die Datenbank des Traders durch die Akquirierung von Dienstbeschreibungen aufgebaut. Dabei werden Inhalt und Umfang einer Dienstbeschreibung wesentlich durch die realisierte Vermittlungsfunktion bestimmt. Im folgenden wird zunächst auf unterschiedliche Arten der Dienstakquisition (4.2.1) eingegangen. Anschließend werden die Spezifikation der Dienstfunktionalität mit Hilfe von Typsystemen (4.2.2), die Angabe von Diensteigenschaften mit Hilfe von Dienstattributen (4.2.3) sowie die Strukturierung der Angebote beschrieben (4.2.4). Jeder Abschnitt enthält eine Aufzählung unterschiedlicher Varianten und eine Bewertung der Alternativen.

4.2.1 Dienstakquisition

Der Trader hält in seiner Datenbank die Angebote der zur Verfügung stehenden Dienste. Um diese Datenbank aufzubauen, d.h. bei der Dienstakquisition, lassen sich in Abhängigkeit von der Dynamik und der Verteilung der Initiative folgende Varianten unterscheiden (vgl. Tabelle A.3):

- Statische Konfiguration auf der Basis von Konfigurationsdateien.
- Dynamische Akquisition durch eine Server-initiierte Export-Operation.
- Import von Dienstmengen (z.B. von anderen Tradern oder aus anderen Quellen).
- Aktives Suchen nach vorhandenen Diensten (z.B. durch Suche in einer Konfigurationsverwaltung oder durch ein allgemeines Ressourcen Discovery System gemäß [ScHa91] wie etwa einen Web-Wurm).

Die statische Variante wird vor allen Dingen bei sehr kleinen und überschaubaren Systemen angewendet, bei denen sich die Menge der angebotenen Dienste selten ändert. Die dynamische Akquisition der Dienstangebote und der Import von Dienstmengen stellen im Augenblick die Regel dar und unterstützen vor allen Dingen einen offenen Markt autonomer Dienstanbieter. Das aktive Suchen nach vorhandenen Diensten erfordert den Zugriff auf Informationsquellen,

die es dem Trader erlauben, aktiv nach Diensten zu suchen. Da das aktive Suchen in der Regel mit einem hohen Aufwand verbunden ist, wurde dieser Ansatz bisher nicht weiterverfolgt. In zukünftigen kooperativen Informationssystemen wird dieser Fall jedoch eine wesentliche Rolle spielen, da in der Regel auf zentrale Vermittlungskomponenten verzichtet wird. In diesem Fall muß ein Trader folglich selbst bei der Dienstakquisition aktiv werden, da er sich nicht auf den freiwilligen Export der Dienstanbieter verlassen kann. Eine Untersuchung dieses Falles führt jedoch über die Aufgabenstellung dieses Berichtes hinaus.

4.2.2 Beschreibung der Dienstfunktionalität durch Typsysteme

Die Dienstfunktionalität kann durch Typen eines bestimmten Typsystems beschrieben werden. Ein Typ spezifiziert die Art des angebotenen oder des gesuchten Dienstes. Dabei unterstützt das Typsystem eine Typüberprüfung zur Laufzeit und verhindert somit fehlerhafte Zuordnungen. Durch einen geeigneten Aufbau des Typsystems kann anstelle einer strikten Typüberprüfung auch die Überprüfung von Kombatibilitätsrelationen - beispielsweise im Sinne der objekt-orientierten Vererbungsrelation - treten, wodurch sich bei der Dienstvermittlung neue Möglichkeiten ergeben. Dies unterstützt beispielsweise eine inkrementelle Weiterentwicklung des verteilten Systems oder durch Definition geeigneter Abbildungen zwischen Typsystemen auch die Interoperabilität zwischen autonom und unabhängig entwickelten Systemen.

Wie im folgenden genauer erläutert, reicht die Spanne der verwendeten Typsysteme von einfachen, universell eindeutigen Bezeichnern über komplexe, strukturierte Typen, die unterschiedliche Aspekte eines Dienstes unabhängig charakterisieren, bis hin zu beschreibenden, dynamisch lernenden Typsystemen, die unscharfes Wissen über die Ähnlichkeit zwischen Diensten ausnützen, um den Benutzer auf gleichartige Dienste aufmerksam zu machen.

Eine fundamentale Diskrepanz zwischen verschiedenen Typsystemen besteht darin, inwieweit zwischen der Charakterisierung der Funktionalität eines Dienstes und der Betrachtung seines Verhaltens in einem Kommunikationsvorgang, d.h. seinen Anforderungen an die Syntax der Kommunikationsparameter, an die verwendeten Kommunikationsprotokolle, etc. unterschieden wird. Heutige Systemplattformen für verteilte Systeme wie der SUN RPC oder das DCE erfassen beide Aspekte in einem einzigen Schnittstellentyp. Aktuelle Ansätze im Trading-Umfeld unterscheiden zwischen dem Diensttyp und dem Schnittstellentyp.

Der Diensttyp definiert hierbei die Art der erbrachten Dienstleistung, der Schnittstellentyp die syntaktischen Details der Dienstschnittstelle, beispielsweise die angebotenen Operationen, ihre Signaturen, die Art des Dienstzugangs (z.B. das verwendete Kommunikationsprotokoll). Im Regelfall legt der Diensttyp bereits den Schnittstellentyp fest, jedoch werden immer häufiger generische Dienste realisiert, die eine allgemein nutzbare Schnittstelle anbieten, über die dann unterschiedliche Dienste realisiert werden (Beispiel: verschiedene Informationsserver im WWW).

Typsysteme werden mit Hilfe eines Typmanagements verwaltet. Dieses erlaubt die Definition von Typen und von Relationen zwischen diesen Typen, stellt eine (formelle oder informelle) Typbeschreibung zur Verfügung und kann die Typdefinition mit weiteren beschreibenden Attributen unterstützen.

Diensttypen und Relationen
Zwischen Diensttypen können Kompatibilitätsrelationen bestehen. Allgemein drückt eine Kompatibilitätsrelation aus, daß der kompatible Dienst anstelle des ursprünglichen Dienstes verwendet werden kann. Um dies konkret zu nutzen, wird entweder eine ebenfalls kompatible

Schnittstelle oder eine Methode zur Abbildung der beiden verschiedenen Dienste benötigt. Hierbei gibt es für Diensttypen die folgenden Varianten (vgl. Tabelle A.1):

- Flache Bezeichner
  Diensttypen werden durch einen strukturlosen, flachen Identifier bezeichnet, wobei Beziehungen zwischen Diensttypen nicht spezifiziert werden können. In diesem Fall werden meist auch keine separaten Schnittstellentypen betrachtet.
- Flache Typen mit Versionen
  Ein Typ kann mehrere Versionen besitzen. Unterschiedliche Versionen eines Diensttyps können zueinander kompatibel sein. Im Regelfall besteht eine Aufwärtskompatibilität zwischen aufeinanderfolgenden Versionen. Verschiedene Entwicklungswerkzeuge erlauben es einem Server, gleichzeitig mehrere Versionen eines Diensttyps anzubieten (beispielsweise der SUN RPC, [Inc87] oder DCE-RPC; bei SUN-RPC trägt der Implementierer die Verantwortung, daß für jede Version eines RPC-Interfaces die entsprechende Implementierung vorhanden ist). Dadurch kann die Aufwärtskompatibilität progammiertechnisch unterstützt werden.
- Explizite Subtyprelation
  Die Subtyprelation erlaubt es, einen Dienst anstelle eines anderen zu verwenden. Bei baumartigen Relationen kann die Baumstruktur zur Benennung eines Dienstes verwendet werden. Bei einem azyklischen Graphen, wie er bei Mehrfachvererbung typischerweise vorkommt, werden flache Bezeichner für den Typnamen und eine explizite Repräsentation der Relation im Typmanager verwendet.
- Konzeptgraphen
  Konzeptgraphen (PMGG95) modellieren Ähnlichkeiten zwischen Diensten unter Zuhilfenahme eines bipartiten Graphens. Der Graph besteht aus Konzepten und Relationen zwischen Konzepten. Ein Abstandsmaß bestimmt die Ähnlichkeit zwischen verschiedenen Clustern des Konzeptgraphen.
- KI-Sprachen KIF und KQML
  Gemäß [GSS94] können die in der Künstlichen Intelligenz entwickelten Sprachen KIF (knowledge interface language) und KQML (knowledge query and markup language) dazu eingesetzt werden, die Fähigkeiten und Bedürfnisse von Agenten zu definieren und auf diesem Wege eine Vermittlung zu ermöglichen.

Die Typverwaltung enthält noch weitere Angaben über einen Diensttyp. Beispielsweise können zusätzlich die für einen Diensttyp vorgeschriebenen, notwendigen Dienstattribute, eine Reihe von DEFAULT-Auswahlregeln, o.ä. verwaltet werden.

Schnittstellentypen
Analog zum Diensttyp wird auch der Schnittstellentyp mit Hilfe des Typmanagements verwaltet. Für Schnittstellentypen gibt es die folgenden Möglichkeiten

- Flache Namen bzw. flache Namen mit Versionen
  können analog zum Diensttyp verwendet werden.
- Explizite Subtyp-Relationen
  Schnittstellentypen können genauso wie Diensttypen in einer Subtyprelation stehen. Eine Subtyprelation definiert dabei, daß für jede Operation des Supertyps eine kompatible Operation im Subtyp enthalten ist. Infolgedessen können Dienste des Subtyps anstelle des Supertyps genutzt werden. Subtyprelationen werden in der Regel mit Hilfe eines azyklischen gerichteten Graphen explizit spezifiziert.

- Schnittstellenbeschreibung mit Signaturen
  Die Schnittstelle wird durch die Signatur der angebotenen Operationen definiert (siehe z.B. [BHJH87] oder [MML95]). Die Signatur kann zum Aufbau eines Typgraphen und dementsprechend für die spätere Dienstvermittlung genutzt werden, um kompatible Operationen zu bestimmen. Beispiel: Operation A ist kompatibel zu B, falls der Definitionsbereich von A eine Obermenge des Definitionsbereichs von B darstellt und gleichzeitig der Wertebereich von A eine Teilmenge des Wertebereichs von B ist. Diese Untersuchung der Signaturen kann genutzt werden, um eine Subtyprelation automatisch zu erstellen. Diese kann explizit im Typmanagement gespeichert sein oder dynamisch jeweils zur Laufzeit berechnet werden.
- Schnittstellentypen mit Signaturen und Typ Coercion
  Typ Coercion nutzt die Kompatibilität zwischen Datentypen (z.B. zwischen Integer und Realzahl), um eine Kompatibilität zwischen unterschiedlichen Operationen zu definieren (siehe [MaBl92]).
- Heterogene Dienstnutzung
  Dienste mit unterschiedlichen Schnittstellen aber gleicher Funktionalität können mit Hilfe von geeigneten Funktionen aufeinander abgebildet werden. Häufig wird dazu noch eine abstrakte (lokale) Schnittstelle eingeführt, welche dann durch ein Übersetzungsmodul auf die konkrete Dienstschnittstelle abgebildet wird. Dies kann sogar durch dynamisch Erweiterung des laufenden Systems erfolgen (siehe [ChRa94]). Andere Ansätze führen die Dienstschnittstellenbeschreibung auf eine Algebra über die beteiligten Sorten (Datentypen) und Operationen zurück und versuchen einen Homomorphismus zwischen diesen Algebren zu bestimmen ([HePo92]).
- Mapping-Dienste
  Anstelle von abgeleiteten bzw. konstruierten lokalen Mapping-Funktionen können vordefinierte Mapping-Dienste verwendet werden, mit denen verschiedene Schnittstellen aufeinander abgebildet werden. Diese Abbildungsdienste können mit Hilfe des Typ-Managers - beispielsweise als beschreibendes Attribut einer Subtyp-Relation - verwaltet werden. Beispiele für derartige Mapping-Dienste sind Anwendungs-Gateways oder Proxy-Agenten im Netzwerkmanagement.

Neben den hier erläuterten Möglichkeiten läßt sich noch eine Reihe von weiteren Ansätzen aufzählen, zu denen es jedoch nur wenige praktische Erfahrungen gibt. Beispielsweise gibt es Ansätze zur formalen Spezifikation von Dienstschnittstellen, die dann dynamisch aufeinander abgebildet werden können (siehe [vB90]). [MML94b] bietet die Möglichkeit, Alias-Namen für Typen zu verwenden, um damit dem menschlichen Benutzer die Verwendung von unterschiedlichen Namen für denselben Typ zu ermöglichen.

Eine Typverwaltung kann auch einen Typraum aus unterschiedlichen Arten von Typen verwalten. Beispielsweise können der Diensttyp und der Schnittstellentyp im gleichen Typraum verwaltet werden (siehe [Beut92] oder [MML94b]). Verteilte objektorientierte Systeme benötigen einen gemeinsamen Typraum für unterschiedliche Objekte, um eine verteilte Programmentwicklung zu unterstützen ([BHJH87]). [IBR93] beschreibt ein allgemeines Type Management System für ODP Trader.

Bewertung:
Durch unterschiedliche Repräsentationen der Funktionalität von Diensten (z.B. durch einen standardisierten Diensttyp bei ODP, durch Konzeptgraphen oder Beschreibungen in speziellen KI-Sprachen) wird eine Vermittlung für unterschiedliche Benutzertypen (traditionelle Klientensoftware, menschliche Benutzer oder intelligente Agenten) ermöglicht. Wie bereits

erwähnt, kann der Vermittlungsspielraum des Traders durch Kompatibilitätsrelationen zwischen Typen erheblich erweitert werden. Unterschiedliche Schemata unterstützen dies in unterschiedlichem Maße. Während flache Namen lediglich der eindeutigen Kennzeichnung eines Diensttyps dienen, können durch Hinzunahme von Versionen Beziehungen zwischen unterschiedlichen Diensttypen ausgedrückt werden können. Unterschieden wird darüberhinaus, inwieweit Relationen zwischen Typen explizit vom Benutzer angegeben werden müssen oder automatisch herausgefunden werden können.

Ein Beispiel für Kompatibilitätsrelationen ist die Subtyp-Relation, die mit der Vererbungshierarchie im objektorientierten Programmieren vergleichbar ist. Hier wie da wird mit dieser Relation eine Kompatibilität ausgedrückt, bei der ein Objekt des einen Typs den Platz des Objektes des anderen Typs einnehmen kann. Dies erlaubt z.B. eine flexible Weiterentwicklung eines Dienstes, wobei Dienstnutzer mit einem alten Benutzeragenten die neuen Dienstvarianten weiterhin nutzen können. Ein automatisches Erkennen von Suptyprelationen ist teilweise schon möglich.

Durch die Unterscheidung in Dienst- und Schnittstellentyp wird zwischen der syntaktischen und der semantischen Kompatibilität von Diensten so weit wie möglich getrennt. Bei kompatiblen Schnittstellentypen kann der Dienst in der gleichen syntaktischen Weise genutzt werden. Bei kompatiblen Diensttypen wird zwar der gleiche funktionelle Dienst erbracht, jedoch nicht notwendigerweise mit der gleichen Schnittstelle. Zwischen kompatiblen Diensttypen mit unterschiedlichen Schnittstellentypen werden Abbildungsfunktionen benötigt. Falls diese nicht automatisch abgeleitet werden können, so sind sie explizit zu definieren. Dabei entstehen die im folgenden behandelten Probleme.

Bei Diensttypen mit Signaturen können Operationen mit gleichen oder ähnlichen (Typ Coercion) Signaturen automatisch angepaßt werden. In der Regel ist dies auf einer syntaktischen Ebene problemlos möglich, scheitert jedoch meist bei der Betrachtung der Semantik der Server-Operationen.

Die heterogene Dienstnutzung nutzt die Möglichkeiten der Dienstbeschreibung mit Signatur (entspricht den algebraischen Sorten), um die Kompatibilität zwischen den Diensten auf die Überprüfung eines Homomorphismus zwischen den Algebren zu reduzieren. Wird ein Homomorphismus ermittelt, so ist dadurch automatisch eine Abbildungsfunktion gefunden. Auch hier stellt sich die Frage, ob eine syntaktische Gleichheit auch eine semantische Äquivalenz bedeutet. Der Definition von Mapping-Funktionen liegt die Idee zu Grunde, die Abbildung zwischen unterschiedlichen Diensten explizit zu definieren und nicht automatisch zu generieren. Bei explizit definierten Mapping-Funktionen kann davon ausgegangen werden, daß die Abbildung sinnvoll ist und ein vernünftiges Resultat liefert, wovon man bei automatisch generierten Abbildungen nicht immer ausgehen kann.

Allgemein wird nach Möglichkeiten gesucht, die starr festgelegten Schnittstellen zwischen Dienstnutzern und Dienstanbietern in kontrollierter Weise zu durchbrechen. Mit Hilfe der dadurch gewonnenen Flexibilität erhofft man sich eine leichtere Weiterentwicklung und Anpassung bestehender Systeme an neue Gegebenheiten. Jedoch kann man angesichts der vielen vorgeschlagenen Speziallösungen feststellen, daß die vorhandenen Möglichkeiten hier bisher nur unzureichend untersucht worden sind.

4.2.3 Eigenschaften von Diensten

Neben dem Diensttyp gibt es noch weitere Eigenschaften, die einen Dienst beschreiben und daher in der Dienstvermittlung berücksichtigt werden. Die Dienstbeschreibung enthält daher

neben dem Dienst- und Schnittstellentyp Dienstattribute für Verwaltungsinformationen, Qualitäts- und Kostenaussagen des Dienstes. Diese lassen sich aufteilen in zwingend erforderliche wie beispielsweise den Zugangspunkt zu einem Dienst (d.h. Name oder Adresse) und in optionale wie z.B. seine Warteschlangenlänge.

Neben den Attributen, die den Dienst direkt beschreiben, gibt es Eigenschaften, die aus dem Umfeld des Diensterbringers stammen, jedoch bei der Auswahl eines Diensterbringers eine wesentliche Rolle spielen. Beispiele für diese Umgebungsparameter sind: Eigenschaften des Gastrechners, auf dem ein Dienstleistungsprozeß abläuft, Aussagen über die Verbindung zwischen einem Klienten und dem Server oder die bei anderen Diensten (z.B. dem Sicherheitsdienst) registrierten Eigenschaften.

Festlegung der Attributmenge:
In Abhängigkeit von der Flexibilität der Dienstvermittlung kann die Attributmenge einer Dienstbeschreibung unterschiedliche Ausprägungen besitzen:

- Freie Attributwahl
  Die bei der Beschreibung verwendeten Attribute sind zwar standardisiert, können aber vom Diensterbringer frei vergeben werden. Dies erlaubt natürlich eine flexible Beschreibung eines Dienstes, erschwert jedoch die Formulierung geeigneter Auswahlkriterien. Selbst wenn gleiche Diensteigenschaften gleich benannt werden, können unvergleichbare Dienstbeschreibungen entstehen, wenn z.B. nicht alle Attribute präsent sind.
- Feste Attributmenge
  Alle Dienste werden immer durch die gleiche Menge an Attributen beschrieben. Dies erlaubt die Formulierung allgemeiner Auswahlkriterien, ist aber im Regelfall zu unflexibel für einen allgemeinen Trading-Dienst, da nicht auf die individuellen Besonderheiten des einzelnen Dienstes eingegangen werden kann.
- Diensttyp-spezifische Attribute
  Für jeden Diensttyp existiert eine definierte Menge an Attributen, die immer angegeben werden muß. Damit können diensttyp-spezifische Eigenschaften beschrieben und dementsprechend auch spezifische Auswahlkriterien formuliert werden. Durch die feste Vorgabe der Attribute kann es außerdem keine Probleme bei der Formulierung diensttyp-spezifischer Anfragen geben.
- Notwendige und zugelassene Attribute
  Attribute werden in notwendige und optionale Attribute unterteilt. Notwendige Attribute müssen präsent sein, optionale Attribute können vorhanden sein. Dies erlaubt mehr Flexibilität bei der Beschreibung eines Dienstes, wirft jedoch Probleme bei der Behandlung von Auswahlkriterien auf, da es hier bei fehlenden optionalen Attributen zu undefinierten Fällen kommen kann. Häufig wird die Menge der notwendigen und zugelassenen Attributen mit der Vererbungsrelation zwischen Diensttypen weitergegeben, so daß formulierte Auswahlkriterien auch auf abgeleitete Dienste angewendet werden können.
- Diensttyp-spezifische Attribute und freie Zusatzattribute
  Mit diesem Fall wird versucht, die Vorteile einer definierten Menge von Dienstattributen mit den Möglichkeiten frei wählbarer Zusatzattribute zu kombinieren. Jeder Diensttyp muß die für ihn definierten notwendigen und zugelassenen Attribute unterstützen. Der Dienstanbieter kann darüberhinaus noch eigene Attribute zur Dienstbeschreibung definieren. Das Konzept der freien Zusatzattribute kann analog den “Conditional Packages” des OSI Managements erweitert werden.

- Dienstattribute und Attribute des Dienstangebots
  Neben den Dienstattributen, die Eigenschaften des Dienstes beschreiben und nach einem der oben definierten Schemata definiert werden, gibt es noch Attribute, die das Dienstangebot selbst beschreiben. Beispielsweise kann eine Dienstbeschreibung mit einem Gültigkeitsdatum oder Zugriffskontrollisten versehen sein, die bestimmte Aspekte der Nutzung der Dienstbeschreibung regeln.

- Planungsattribute
  Für die eigenen Vermittlungszwecke kann der Trader zusätzliche Attribute einführen. Beispielsweise kann er sich in einem zusätzlichen Planungsattribut die Anzahl der Vermittlungen eines Dienstes merken und diese bei späteren Vermittlungen berücksichtigen.

Eine abschließende Bewertung der verschiedenen Varianten fällt schwer, da hier ein geeigneter Kompromiß zwischen der Standardisierung von Dienstattributen und den zugelassenen Freiheitsgraden für einen Dienstanbieter gewählt werden muß. Mit einem festen Satz an Dienstattributen (entweder für alle oder für alle Dienste des gleichen Typs) lassen sich Auswahlkriterien einfach formulieren und leicht testen. Die freie Wahl der Attributmenge läßt den Dienstanbietern große Freiheiten, erschwert jedoch die Suche nach geeigneten Diensten, da der Vergleich zwischen Dienstbeschreibungen nur schwer möglich ist.

Typen von Dienstattributen
Die Dienstattribute bestehen aus einem Attributtyp und einem Attributwert. Manchmal wird der Attributtyp noch in einen Namen und einen Datentyp aufgeteilt, jedoch erschwert dies im Regelfall den Umgang mit Attributen (da z.B. immer eine Datentypkompatibilität geprüft werden müßte), ohne wesentlich neue Möglichkeiten zu eröffnen. Attribute können auch nach weiteren Kriterien klassifiziert werden (vgl. Tabelle A.2):

- Einfache und strukturierte Attribute
  Analog zu einfachen und strukturierten Datentypen kann der Wert eines Attributs aus einem einfachen Datentyp oder aus einem zusammengesetzten Datentyp bestehen. Zusammengesetzte Datentypen erschweren die Formulierung von Auswahlkriterien und werden häufig vermieden.

- Mengenwertige Attribute
  Mengenwertige Attribute sind ein Spezialfall von strukturierten Attributen, bei denen der Wert eines Attributs eine Menge von (Basis-) Datentypen sein kann. Durch die Mengenoperationen stehen einfache Möglichkeiten zur Formulierung von Auswahlkriterien bereit, die noch durch Funktionen auf Mengen (beispielsweise Minimum, Durchschnitt, o.a.) ergänzt werden können.

- Statische und dynamische Attribute
  Bei statischen Attributen bleibt der Wert eines Attributs über einen längeren Zeitraum unverändert, bei dynamischen Attributen ändert er sich mit der Zeit. Dynamische Attribute erfordern in der Regel eine eigene Art der Behandlung. Einige Ansätze bilden dynamische Attribute auf OSI-Managementobjekte (z.B. [PTT91]) oder auf verteilte Objektsysteme ([Wira94]) ab.

Neben den direkt vom Diensterbringer unterstützten Attributen gibt es eine Reihe von Attributen, die aus anderen Quellen gewonnen werden und für die Dienstvermittlung von Interesse sind:

- Implizite Attribute
  Implizite Attribute sind Eigenschaften eines Dienstes, die durch Beobachtung gewonnen werden können. Zu diesen gehören beispielsweise mittlere Job-Ausführungszeiten oder die

durchschnittliche Auslastung eines Dienstes. Implizite Attribute werden in der Regel mit Hilfe einer Managementfunktion gewonnen, wobei der Dienst dem Managementsystem häufig gewisse Grundwerte (z.B. über die augenblickliche Auslastung) zur Verfügung stellen muß. Managementfunktionen berechnen daraufhin den gewünschten Wert und stellen diesen dem Trader zur Verfügung.

- Umgebungsattribute
  Umgebungsattribute beschreiben die Umgebung des Dienstes. Es sind Eigenschaften, die für den Trading-Prozeß relevant sind. So kann beispielsweise der Rechner, auf dem ein Dienst ausgeführt wird oder die Netzwerk-Verbindung durch Umgebungsattribute beschrieben werden. Typischerweise werden Umgebungsattribute ebenfalls durch das Managementsystem zur Verfügung gestellt.
- Klienten-spezifische Attribute
  Diese Attribute beschreiben Eigenschaften eines Servers in Abhängigkeit von speziellen Klienten. Dies kann von Aussagen über die bisherige Dienstnutzung eines Klienten bis hin zum mittleren Durchsatz der Kommunikationsverbindung zwischen Klient und Server reichen.
- Historien der Dienstnutzung
  Sowohl Dienstnutzer als auch Dienstanbieter können über bisherige Dienstnutzungen Buch führen. Diese Historien können in die Dienstvermittlung einfließen.

Die Möglichkeiten einer Dienstvermittlung werden entscheidend durch Beschreibungsmöglichkeiten eines Dienstangebots bestimmt. Demzufolge ist es wünschenswert, nicht nur statische sondern auch dynamische Attribute bei der Dienstvermittlung zu berücksichtigen. Dabei erfordern zusätzliche Möglichkeiten auch zusätzlichen Aufwand, um die gewünschten Informationen bereitzustellen. Ein großes Problem hierbei ist die Vielfalt der unterschiedlichen Dienste und die damit verbundene Vielzahl an Dienstattributen. Es stellt sich die Frage, ob es im Fall von hochkomplexem Vermittlungsbedarf sinnvoller ist, einen generischen Dienst geeignet anzureichern oder besser spezialisierte Vermittlungsfunktionen bereitzustellen. Eine genaue Untersuchung dieser Fragestellung steht jedoch noch aus.

4.2.4 Die Datenbank des Traders: die Angebotsverwaltung

Die Angebotsverwaltung speichert die Dienstangebote eines Traders. Eine weitergehende Strukturierung der Angebotsmenge erlaubt es, neben den reinen Suchkriterien auf der Basis der Diensttypen und Dienstattribute auch strukturelle Aspekte (meist administrativer Art, wie die Zuordnung des Diensterbringers zu einer Abteilung) bei der Dienstvermittlung zu berücksichtigen. Hierzu gibt es verschiedene Möglichkeiten (vgl. Tabelle A.4):

- Flache Angebotsstruktur
  Angebote werden in einer flachen Angebotsliste verwaltet, die keinerlei Struktur aufweist.
- Einstufige Dienststruktur (Zellen)
  Dienste werden in einer einstufigen Hierarchie (Zellen) gehalten. Dies ermöglicht eine Unterteilung des Dienstangebots nach genau einem Kriterium, meist nach der administrativen Zugehörigkeit ([Walk94]).
- Hierarchische Dienststrukturierung (Kontexte) - ohne Ordnungskriterien
  Dienstangebote werden in hierarchischen Kontextstrukturen eingetragen - dem Kontextraum in Baumform. Die Struktur des Kontext wird durch die Verwalter und Nutzer des Traders erzeugt. Ordnungskriterien für die Struktur des Baumes sind daher nur implizit

vorhanden und können vom System nicht überprüft oder irgendwie berücksichtigt werden. Meist werden Konventionen verwendet, um zwischen den verschiedenen Nutzern ein gemeinsames Verständnis für die Baumstruktur zu erreichen. Das System unterstützt keine Überwachung dieser Konventionen. Dienste können in einem Kontext beliebig gemischt werden. In der Praxis läßt sich erwarten, daß sich (analog zu hierarchischen Dateisystemen) die Flexibilität eines nicht-restriktiven Baumaufbaus bewährt.

- Hierarchische Dienststrukturierung (Kontexte) - mit Ordnungskriterien
  Dienste werden in Kontexte eingeordnet. Kontexte stehen in einer hierarchischen Beziehung, durch die eine Baumstruktur entsteht. Diese Baumstruktur kann Ordnungskriterien widerspiegeln, beispielsweise die Hierarchie zwischen Lokationen (Land - Stadt - Straße). Andere Möglichkeiten nutzen die Struktur des Baumes, um administrative Zuständigkeiten oder funktionelle Unterschiede zu gliedern. Wichtigstes Merkmal dieser Variante ist die feste Vorgabe der Ordnungskriterien. Die festen Vorgaben können kontrolliert werden. So kann das Ordnungskriterium Teil einer Attributmenge sein, die den Kontext beschreibt.
- Mitgliedbedingung (Membership-Rules)
  Kontexte werden durch eine Mitgliedsbedingung definiert. Dienste werden automatisch in die Kontexte exportiert, deren Mitgliedsbestimmungen auf sie zutreffen. Diese Vorgehensweise hat Vorteile bei der Zusammenarbeit von Tradern, da hier automatisch die Dienste der Partner-Trader in die richtigen Kontexte eingeordnet werden und somit ohne weitere Maßnahmen zur Verfügung stehen ([BeRa93]). Im Unterschied zur hierarchischen Dienststrukturierung mit Ordnungskriterium stehen die Kontexte in keiner explizit festgelegten Beziehung.
- Attribut-basierte Namen und Verwaltung
  Dienstangebote werden mit einer Menge von Attributen beschrieben, die das Dienstangebot eindeutig beschreiben. Der gewünschte Dienst kann dann mit Hilfe eines attributierten Anfrageschemas ([Pete88]) gefunden werden. Ansätze dieser Art helfen, das Problem der Namensheterogenität unterschiedlicher Systeme zu lösen (siehe [NiGo93]).
- Verwendung des X.500 Directories
  Die Verwendung des X.500 Directories stellt einen häufig anzutreffenden Spezialfall der Angebotsverwaltung im Trader da. In diesem Fall werden die Dienstangebote im X.500-Directory verwaltet. Diese Lösung hat den Vorteil, daß das für diese Fälle vorgesehene internationale Directory-System auch weiterhin genutzt wird, während die zusätzliche Aufgaben der Trader-Komponente in der Auswahl geeigneter Dienste - u.U. unter Berücksichtigung dynamischer Attribute oder vorgegebener Auswahlstrategien - besteht.
- Verwendung einer allgemeinen Strukturierungsabbildung
  Gemäß ISO95 werden Kontexte über eine als *partition* bezeichnete Abbildung definiert.

Bewertung

Die Strukturierung der Angebotsverwaltung erlaubt die Einschränkung der Dienstvermittlung auf Teilmengen der Angebotsverwaltung. Mit diesen Teilmengen können administrative oder organisatorische Randbedingungen verknüpft sein. Auch können hiermit Mechanismen zur Autorisierung von Trader-Operationen (beispielsweise die Verwaltung der Zugriffs- oder Änderungsrechte) oder zur Replikation - indem Kontexte beispielsweise als Einheit der Replikation verwendet werden - verbunden werden.

4.3 Dienstvermittlung

Die Dienstvermittlung vergleicht eine *Dienstanfrage* (4.3.1) mit den Dienstangeboten in der Angebotsverwaltung. Neben der Typkompatibilität werden unterschiedliche Suchbereiche (4.3.2) sowie verschiedene Selektions- (4.3.3) und Auswahlstrategien (4.3.4) berücksichtigt. Bei der Suche nach einer besten Auswahl können verschiedene Arten der Optimierung (4.3.5) gewählt werden.

4.3.1 Dienstanfrage

Die *Dienstanfrage (Service Request)* wird vom Dienstnutzer an den Trader übergeben. Sie enthält Angaben über den gesuchten Dienst sowie über die Vorgehensweise, die der Trader bei der Dienstvermittlung wählen soll. Im Einzelnen enthält die Dienstanfrage:

- Den *Typ* des gesuchten Dienstes,
- den *Suchbereich*, in dem nach einem Dienst gesucht werden soll,
- die Angabe einer *Auswahlstrategie*,
- ein optionales *Selektionskriterium*, welches notwendige Anforderungen an den Server beinhaltet,
- das *Optimierungskriterium*, das im Falle der BEST-CHOICE Strategie beschreibt, wie bei mehreren alternativen Servern eine Auswahl durchzuführen ist,
- *Kontrollparameter*, die die Vorgehensweise des Traders beeinflussen (z.B. Zeit- oder Umfangsbegrenzungen).

Aufgrund des spezifizierten Diensttyps und den in den Auswahlkriterien angegebenen Bedingungen werden Dienstangebote ausgewählt und bewertet. Je nach Vorgabe des Benutzers werden eine Menge von mehreren Servern oder ein einzelner Server vorgeschlagen. Typischerweise wird eine Dienstanfrage an den Trader gestellt, woraufhin dieser aktiv wird. Prinzipiell ist es jedoch auch denkbar, Dienstanfragen ebenfalls im Trader zu verwalten, um sich dann über die Änderung der verfügbaren Angebotsmenge informieren zu lassen. Der Trader muß dann nach einer vordefinierten Auswertungsstrategie die Dienstanfragen wiederholt prüfen. Dieser Fall entspricht dem in [GSS94] beschriebenen Konzept, in dem Agenten ihre Fähigkeiten und Bedürfnisse einer dem Trader vergleichbaren Komponente spezifizieren und von dieser geeignete Partner vermittelt bekommen.

4.3.2 Suchbereiche und -ordnungen

Die Suche nach geeigneten Diensten kann auf bestimmte Bereiche der eigenen oder fremder Angebotsverwaltungen beschränkt werden. Dies vermindert den durch das Trading entstehenden Aufwand. Je nach Strukturierung des Dienstangebots in eine flache Zellenstruktur oder in einen hierarchischen Baum lassen sich unterschiedliche Formen der Bereichsdefinition denken:

- Aufzählung der abzusuchenden Bereiche,
- Angabe eines Wurzelkontexts und eines Kriteriums, wieweit der anschließende Baum abgesucht werden soll (z.B.: n-te Ebene, vollständiger Teilbaum),

- Berücksichtigung von Ordnungskriterien, beispielsweise alle Kontexte, die Druckdienste enthalten,
- Auswahl der Breiten- oder Tiefensuche in einem Baum,
- Angabe eines Selektionskriteriums, welches die abzusuchenden Kontexte spezifiziert (z.B. auf der Basis kontext-beschreibender Attribute).

Darüberhinaus ist es möglich, über eine Verkettung von Suchbereichen mit Hilfe von bestimmten Übergangsbedingungen eine Suchordnung zu definieren. Dadurch kann der ursprüngliche Suchbereich erweitert werden und es kann zu einer mehrstufigen Suche kommen. Eine mehrstufige Vorgehensweise bietet die Möglichkeit, bestimmte Bereiche des Netzes zu bevorzugen. Dabei kann die Bedingung, wann zu einem folgenden Suchbereich übergegangen wird, flexibel definiert sein. Beispiele für solche Bedingungen sind:

- Vorheriger Suchbereich hat kein Ergebnis geliefert.
- Gefundenes Ergebnis entspricht nicht einer Mindestanforderung, wodurch eine aufwendigere Suche gerechtfertigt ist. Bisherige Ergebnisse können entweder ignoriert oder in den neuen Suchbereich übernommen werden.
- Vorgegebene Betriebsmittel wie Zeit oder erlaubte Netzanfragen wurden noch nicht vollständig ausgeschöpft.

Diese Vorgehensweise kann gekoppelt werden mit der Spezifikation neuer Selektionskriterien (vergleiche folgenden Abschnitt). Dabei kann für jeden Schritt der Suchreihenfolge ein neues Suchkriterium angegeben werden.

Das folgende Beispiel (Abbildung 8) illustriert eine mögliche Suchreihenfolge. Falls im ersten Schritt kein Ergebnis erzielt wurde, wird zum zweiten Suchschritt übergegangen. Erzielt auch dieser kein befriedigendes Ergebnis (“kein Lastfaktor unter 10”), so wird zum dritten Suchschritt übergegangen. Der vierte Suchschritt wird ausgeführt, falls genügend Trader-Ressourcen zur Verfügung stehen, während der fünfte Suchschritt durchgeführt wird, falls die Anfrage zwischen 18:00 und 19:00 Uhr stattfindet.

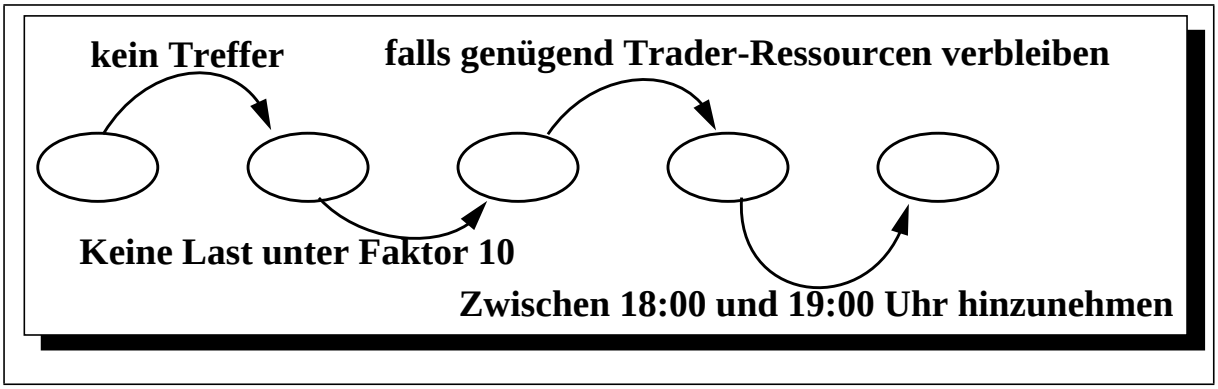


Abbildung 8. Suchreihenfolge

Suchbereiche und -ordnungen können sowohl von Dienstnutzern als auch von Systemverwaltern vorgeben werden. Daher behandeln [PoMe94], [VBB95] sowie [ISO95] die Kopplung von Nutzer- und Trader-Strategien mit einer zweistufigen Vorgehensweise: erst die Festlegung des Suchbereichs einer Anfrage, dann die Einschränkung durch eine systemseitig festgelegte Strategie (vgl. die Ähnlichkeit zu den in Abschnitt 8.1.5 eingeführten, impliziten Auswahlkri-

terien). [Eber93] definiert Views, mit deren Hilfe die Sicht des einzelnen Benutzers auf das Gesamtangebot eingeschränkt werden kann.

Aus unserer Sicht gibt es noch eine Reihe weiterer Möglichkeiten, die bei der Definition der Suchbereiche berücksichtigt werden können, genauere Untersuchungen hierzu stehen jedoch noch aus:

- Einführung von Prioritäten
- Verzweigungen verbunden mit Parallelarbeit auf anderen Tradern
- Optimierung der Evaluation (beispielsweise bevor eine Auswertung des Suchbereichs A fertig ist, bereits Evaluierung des Bereiches B)
- Unterschiedliche Suchkriterien für verschiedene Schritte der Anfrage erlauben eine flexible Suche nach Diensten:

  Beispiel:

  [Bereich A (Leistungszahl > 10 und Kosten < 5);]
  if (no Match) then [Bereich A (Leistungszahl > 5 und Kosten < 8)]

Bewertung:
Durch die Angabe von Suchbereichen werden mehrere Ziele verfolgt. Zum einen soll die Anzahl der zu überprüfenden Dienste auf eine Teilmenge der Gesamtheit reduziert werden. Zum zweiten wird die Struktur der Angebotsverwaltung genutzt, um Strukturen der realen Welt, beispielsweise Abteilungen einer Firma, widerzuspiegeln. In diesem Fall dient die Einschränkung des Suchbereichs der Berücksichtigung der entsprechenden Strukturen bei der Anfrage. Durch die Verkettung verschiedener Suchbereiche mit Übergangsbedingungen wird unnötig häufige Kommunikation zwischen dem Dienstnutzer und dem Trader eingespart.

4.3.3 Selektionskriterium

Mit Hilfe der Dienst- und Schnittstellentypen wird die Menge der vorhanden Dienste auf Grund der funktionalen Aspekte des Dienstes eingeschränkt. Das Selektionskriterium erlaubt es, die Menge der zutreffenden Dienste auf Grund ihrer sonstigen Eigenschaften einzuschränken. Es gibt folgende Varianten:

- Boolscher Ausdruck
  Ein Selektionskriterium ist ein boolscher Ausdruck über der Menge der Attribute eines Dienstes. Der boolesche Ausdruck gibt ein absolutes Kriterium an, welches ein Dienst erfüllen muß, um bei der Dienstauswahl berücksichtigt zu werden.
- Dienstbewertung und Abstandsnorm
  Anstelle eines absoluten Selektionskriteriums wird eine Abstandsnorm angegeben, die ein Dienstangebot bewertet. Dienste, die in einem bestimmten Radius vom Ursprung liegen, werden verwendet. Diese Vorgehensweise hat den Vorteil, daß sehr einfach das absolute Auswahlkriterium zu einem unscharfen Kriterium aufgeweicht werden kann, indem Dienste, die zwar außerhalb aber immer noch nahe am Auswahlradius liegen, ggf doch berücksichtigt werden können.

4.3.4 Auswahlstrategie

Die Auswahlstrategie legt fest, wie unter der Menge der durch das Selektionskriterium eingeschränkten Dienste ein bestimmter Dienst ausgewählt wird. Auswahlstrategien stellen eine einfache Möglichkeit dar, die Trader-Entscheidung an bestimmte verfolgte Ziele (Strategien) anzupassen. Beispielsweise wird ein Systemverwalter durch den Zwang zur gleichmäßigen Auslastung der Systemressourcen eher eine Strategie bevorzugen, die die Last gleichmäßig zwischen verschiedenen Dienstanbietern verteilt. Der individuelle Benutzer hingegen möchte einen möglichst optimalen und qualitativ hohen Dienst (z.B. mit schnellen Antwortzeiten) erhalten. Insbesondere in einem kooperativen Umfeld mit im Wettbewerb stehenden Dienstanbietern ergeben sich hier neue Problemstellungen. In der Literatur werden die folgenden Möglichkeiten diskutiert (vgl. Tabelle A.5):

- FIRST CHOICE
  Bei der Suche im Suchbereich wird der erste Server ausgewählt, der dem Auswahlkriterium genügt. Der Suchprozeß hängt daher von der internen Ordnung in der Datenbasis des Traders ab. Dies führt häufig zu einer fortwährenden Auswahl des gleichen Servers.
- CONSERVATIVE CHOICE
  Hierbei wird einfach der gleiche Server verwendet, der bei einer vorherigen Entscheidung verwendet wurde. Dies ist vor allen Dingen bei statischen Zuweisungen zwischen Klienten und Servern sinnvoll. Probleme ergeben sich hierbei daraus, daß der Trader über jede Anfrage Buch führen und die Entscheidung für den zuletzt ausgewählten Server überprüfen muß.
- RANDOM
  Unter den möglichen Servern wird eine zufällige Auswahl getroffen. Diese Strategie kann um den Einsatz von Wahrscheinlichkeiten erweitert werden, mit deren Hilfe verschiedene Server bewertet werden können. In diesem Fall kann durch geeignete Bevorzugung leistungsstarker Server trotzdem eine gerechte Verteilung der Last erreicht werden.
- CYCLIC CHOICE
  Unter den vorhandenen Servern werden Zuordnungen in einer zyklischen Reihenfolge vorgenommen. Hier muß entschieden werden, wie bei Anfragen mit unterschiedlichen Selektionskriterien zu verfahren ist.
- BEST CHOICE
  Unter den Servern, die dem Selektionskriterium genügen, wird ein bester ausgewählt. Dies erfolgt durch ein anzugebendes Optimierungskriterium.

Häufig wird in der Literatur auch die sogenannte PROBING-Strategie ([WoTs91]) genannt, die jedoch keine eigentliche Auswahl-Strategie ist. Vielmehr wird zur Reduktion der Vermittlungszeiten und der Netzlast nur eine Untermenge der Server bei der endgültigen Auswahl berücksichtigt. Es handelt sich daher weniger um eine Auswahlstrategie im eigentlichen Sinne (obwohl die Art der Realisierung des PROBINGs sicherlich Einfluß auf die Wahl eines Servers hat) als um eine Strategie zur Beeinflussung der Kosten des Trading-Dienstes.

Je nach Einsatzgebiet des jeweiligen Traders ist die Auswahlstrategie fest in den Trader eingebaut oder kann gezielt durch das Systemmanagement oder den Trader-Nutzer festgelegt werden. Für einen Systemdienst Trading ist sicherlich die freie Auswahl einer geeigneten Auswahlstrategie notwendig.

4.3.5 Optimierungskriterium

Bei der Auswahlstrategie BEST-CHOICE erfolgt eine Bewertung des Dienstangebots nach einem Optimierungskriterium. Es gibt unterschiedliche Vorgehensweisen, wie ein solches Optimierungskriterium aussehen kann (vgl. Tabelle A.7):

- Minimum (Maximum) eines einzelnen Attributs
- Minimum (Maximum) einer geschlossenen Gewichtsfunktion über einer Reihe von Werten
- Punktewertung
  Die unterschiedlichen Dienste bekommen mit Hilfe von Bewertungsregeln Punkte zugeteilt. Die erzielten Punkte sind ein Maß für die Qualität des Dienstes oder für den Abstand des Dienstes von einem idealen Dienst. Punktebewertungen stellen eine ideale Möglichkeit dar, die Wünsche des Benutzers und die Wünsche des Systemverwalters unter einen Hut zu bringen. Beispielsweise können die Anforderungen des Nutzers und des Verwalters getrennt bewertet und dann prozentual gewichtet werden. Probleme bereitet bei diesem Verfahren die flexible Definition der Punktebewertungen.
- Spezialfunktionen (z.B. Simulationen)
  Der Dienst wird durch Spezialfunktionen bewertet ([Walk94]). In der Regel haben Dienste des gleichen Typs die gleiche Bewertungsfunktion. Bei der Realisierung kann die Spezialfunktion in den Trader eingebaut sein. Es ist jedoch genauso möglich, die Bewertungsfunktion durch einen externen Entscheider ausführen zu lassen. Dies ermöglicht eine flexible Gestaltung des Trading-Dienstes.
- Intelligente Agenten
  Der Trader arbeitet mit intelligenten Agenten zusammen, die auf die Auswahl bestimmter Dienstklassen optimiert sind. Diese Agenten können dabei sowohl Wissen über die bisherige Dienstnutzung als auch über eine Wissensbasis zur Bewertung der vorhandenen Dienste verfügen (vergl. [Haas94]). Durch die Kooperation mehrerer Agenten können die unterschiedlichen Interessen von Dienstnutzern und Systemverwaltung auf kanonische Weise modelliert werden. Falls keine Einigung erzielt werden kann, muß ein Schiedsrichter-Agent zwischen den verschiedenen Interessen der bewertenden Agenten koordinieren.

Neben einzelnen durch den Klienten spezifizierbaren Optimierungskriterien können auch komplexe Optimierungsentscheidungen in den Trader eingebaut werden (vgl. z.B. [Walk94]). Dadurch gehen zwar Freiheitsgrade beim Trading verloren, es können jedoch u.U. spezialisierte und auf den Anwendungsfall zugeschnittene Trader entstehen.

4.4 Berücksichtigung dynamischer Attribute

Dynamische Attribute beschreiben Eigenschaften eines Servers, die permanenten Veränderungen unterworfen sind. Um ein möglichst aktuelles Abbild der augenblicklichen Situation zu erlangen, müssen dynamische Attribute im Trader gesondert behandelt werden, indem ihre Werte zu geeigneten Zeitpunkten auf den neuesten Stand gebracht werden (gemäß ISO95 wird dazu ein noch nicht näher spezifizierter *service offer evaluator* eingesetzt). Zur Realisierung dynamischer Attribute gibt es unterschiedliche Implementierungsvarianten (vgl. die Aufstellung in [Kell93] oder [Küpp94]), die im folgenden erläutert werden.

4.4.1 Lokales Caching

Attribute werden in einem lokalen Cache zwischengespeichert und stehen somit für eine gewisse Zeit zur Verfügung. Bei den verschiedenen Varianten zur Aktualisierung unterscheidet man zwischen der Invalidierungs- und der Updatestrategie. Die Invalidierungsstrategie sagt aus, wann ein Wert im Cache ungültig wird. Die Update-Strategie bestimmt dann, zu welchem Zeitpunkt der Wert erneut übertragen wird ([ABC90]).

Invalidierungsstrategie

- Statisches Altern:
  Attributwerte werden nach einem statisch vorgegebenen Intervall ungültig.
- Aktualitätsprädikate ([Kova93a], [Helb92]):
  Aktualitätsprädikate (AP) formulieren Bedingungen für die Invalidierung des Caches, wie z.B. ein zeitliches Gültigkeitsintervall, Anzahl der Cache-Zugriffe, Anzahl der wichtigen Änderungen, etc.

  Aus einem AP ergeben sich entweder Invalidierungsbedingungen, die lokal im Cache überprüft werden können, oder solche, die nur an der Quelle überprüft werden können. Beispielsweise kann bei einem Änderungsprädikat, welches nur eine bestimmte Anzahl an Änderungen der Quelleninformation zuläßt, die Invalidierungsbedingung ausschließlich an der Quelle überprüft werden. In diesem Fall wird der alten Cache-Inhalt durch eine Update-Meldung invalidiert und durch den neuen Wert ersetzt.
- Invalidierung durch Invalidierungsnachricht:
  In diesem Fall ist der Wert im Cache immer gültig, bis eine Invalidierungsnachricht eintrifft. Bei kleinem Datenvolumen wird die Invalidierungsnachricht gleich den neuen Wert des Attributs enthalten. Diese Strategie wird in der Regel dann angewendet, wenn die Update-Strategie von der Originalquelle bestimmt wird.

Update-Strategie

- Zugriff nach der Invalidierung
  Der neue Wert wird direkt nach der Invalidierung geholt.
- Pre-Fetching
  Der neue Wert wird vor dem Ablauf der Invalidierung geholt, um bei der Vermittlung Wartezeiten für den Zugriff über das Netz zu reduzieren. Auslösende Ereignisse für ein Pre-Fetch können z.B. ein abgelaufenes Zeitintervall oder der Beginn einer Leerlaufphase des Traders sein.
- Update bei Zugriff
  Der Wert im Cache wird nach einer Invalidierung erst beim Zugriff auf den Cache erneuert.
- Steuerung durch die Quelle
  Die Quelle des Attributs triggert die Invalidierung und sendet dann sofort den Update.
- Schätzung ([Küpp94])
  Zwischen zwei Updates wird der Wert des Attributs durch Extrapolation (z.B. mit Hilfe des Gradienten) geschätzt. Bei einer guten Schätzfunktion liefert diese Methode gute Ergebnisse. In der Regel muß aber die Schätzfunktion individuell für den Einzelwert bestimmt werden.

Welche Strategie verwendet wird, sollte individuell für einzelne Attribute anpaßbar sein. Aktualitätsprädikate stellen eine Möglichkeit dar, die Anforderungen an die Aktualität für ein einzelnes Attribut individuell zu formulieren.

4.4.2 Weitere Konzepte für die Behandlung von dynamischen Attributen

Dedizierte Zustandsserver
Ein dedizierter Zustandsserver sammelt Informationen über die dynamischen Attribute der verteilten Umgebung und stellt diese den Tradern zur Verfügung. Der Hauptvorteil dieses Servers ist die Trennung zwischen der Informationssammlung und der Nutzung im Trader. Der Zugriff auf alle dynamischen Attribute erfolgt durch einen einzelnen Kommunikationsschritt. Dadurch können bei häufigen Trader-Anfragen unnötige Kommunikationsschritte vermieden werden. Es stellt sich jedoch die Frage, ob durch den mehrstufigen Zugriff die benötigte Information noch ausreichend aktuell ist.

Das Prinzip der gestaffelten Auswertung
Bei diesem Prinzip ([Kova92], [Wira94]) werden für jeden Dienst des gesuchten Typs zuerst die Teile des Suchausdrucks evaluiert, die mit statischer Information auskommen. Führt dies nicht zu einer vollständigen Auswertung, so wird anschließend auf die dynamischen Attribute zugegriffen. Diese Zugriffe können durch mehrere Maßnahmen (asynchroner Zugriff, parallele Bewertung) weiter optimiert werden. Dieses Prinzip reduziert die benötigten Zugriffe und erreicht durch eine Verschiebung des Zugriffs auf dynamische Informationen zum Ende des Auswertungsintervalls eine höhere Aktualität.

Probing
Das Probing ist ein Spezialfall der gestaffelten Auswertung. Hierbei wird nur bei einer Teilmenge der möglichen Dienstangebote auf die dynamische Information zugegriffen. Die Teilmenge selbst kann durch statische Information, beispielsweise durch die Vergabe von Prioritäten, bestimmt werden.

4.4.3 Bewertung

Die Vielzahl der vorgeschlagenen Lösung zeigt, daß das Problem des Zugriffs auf dynamische Attribute noch nicht abschließend gelöst wurde. Hier haben jedoch gerade die Größe der Trading-Domäne, die Anzahl der zu untersuchenden Dienste und die Anzahl der Anfragen einen entscheidenden Einfluß, so daß es möglicherweise keine optimale Lösung gibt. Der durch den Einsatz von dynamischen Attributen gewonnene Vorteil läßt häufig den erhöhten Aufwand gerechtfertigt erscheinen.

4.5 Ergebnis der Dienstvermittlung

Das Ergebnis der Dienstvermittlung hängt von den Erwartungen des Dienstnutzers und von der Einbindung des Traders in den Kommunikationsprozeß (3.1.1) ab. In Abhängigkeit davon liefert der Trader unterschiedliche Ergebnisse. Mögliche Ergebnisse sind:

- Identität eines Dienstanbieters
  Es wird nur die Identität des Diensterbringers, beziehungsweise eines seiner Repräsentanten geliefert. Der Dienstnutzer muß daraufhin einen Namensdienst oder - im Fall mobiler Nutzer und Anbieter - einen Lokalisationsdienst benutzen, um einen Zugangspunkt zum Dienst zu finden.

- Ein oder mehrere Zugangspunkte
  Der Trader kennt die Adresse eines (oder mehrerer) Zugangspunkte, die er als Teil des Ergebnisses zurückliefert.
- Dienstkontrakt
  Das Ergebnis besteht aus einem ausgehandelten Dienstkontrakt, der die Garantien des Dienstanbieters bei der Nutzung enthält. Dabei können die ausgehandelten Garantien explizit repräsentiert und somit einer Überprüfung durch den Nutzer unterzogen werden.

Im Falle der transparenten Dienstnutzung leitet der Trader einen Dienstaufruf direkt an den ermittelten Server weiter. Das Ergebnis dieser Tradernutzung ist daher das Ergebnis des Dienstaufrufs. Ergänzend kann die Identität des ausführenden Servers mit dem Ergebnis des Dienstaufrufs dem Dienstnutzer mitgeteilt werden. Weiterhin kann der Trader den gefundenen Dienst durch Konfigurationsmaßnahmen auf die anstehende Nutzung vorbereiten und damit einen individuell zusammengestellten Dienst vermitteln (Haas94).

4.6 Dienstqualität des Trading-Dienstes

Die Qualität eines Trading-Dienstes kann durch verschiedene Strategien beeinflußt werden. Diese sind entweder fest für einen Trader vorgegeben, können durch Parametrisieren eingestellt werden oder unmittelbar in der Trader-Operation vorgegeben werden. Die Strategien des Trading-Dienstes beeinflussen dessen Wirkungsweise und die Qualität des erbrachten Dienstes. Sie sind damit unabdingbare Elemente eines vollständigen Trading-Dienstes, um eine genaue Kontrolle des Dienstes sowohl durch Anwender (Dienstnutzer oder -erbringer) als auch durch den Systemverwalter zu ermöglichen.

Zu möglichen Strategieparametern für Trader-Operationen zählen u.a.:

- Beschränkungen für die Zeitdauer und die Anzahl gefundener Dienste einer Anfrage
- Maximale Anzahl der involvierten Partner-Trader
- Festlegung des Verhaltens des Traders, falls dynamische Attribute nicht verfügbar sind

Die Festlegung der Dienstqualität eines Traders kann einerseits statisch vorgegeben sein oder dynamisch durch die Definition der entsprechenden Strategien (mitsamt den benötigten Strategieparametern) erfolgen.

4.7 Zusammenfassung

Das vorliegende Kapitel präsentierte Varianten eines allgemeinen Trader-Dienstes. Es wurden verschiedenen Lösungsmöglichkeiten sowohl bei der Dienstakquisition als auch bei der Dienstvermittlung vorgestellt. Die Bandbreite der vorgestellten Lösungen zeigt, daß es zum einen noch viele offenen Probleme und Meinungsverschiedenenheiten auf dem Weg zu einem allgemeinen Trading-Dienst gibt. Für spezielle Anforderungen wird es sehr wahrscheinlich spezielle Ausprägungen eines Trading-Dienstes geben. Eine Herausforderung bei der Definition einer allgemeinen Trader-Funktion wird die Abdeckung eines möglichst breiten Spektrums an Möglichkeiten sein.

5. Kooperation zwischen Tradern

Wie bereits erwähnt, spielt die Kooperation zwischen Tradern, insbesondere zwischen solchen aus unterschiedlichen Domänen, für die Errichtung eines globalen elektronischen Marktes eine wesentliche Rolle. Nachdem im vorangegangenen Kapitel die Wirkungsweise eines einzelnen Traders behandelt wurde, können nun darauf aufbauend Kooperationskonzepte entwickelt werden, die über die in Abschnitt 3.3 klassifizierten Ansätze aus der Literatur hinausgehen. Dazu wird zunächst das in Abschnitt 2.1 eingeführte Kooperationsmodell erweitert, wodurch sich das ganze Spektrum einer Zusammenarbeit von Tradern zeigen läßt. Im Vergleich zu dem in Abschnitt 3.3 geschilderten Stand der Forschung umfaßt dieses Spektrum qualitative Aspekte von Trader-Kooperationen, eine Verallgemeinerung der bisher eingesetzten Verhandlungstypen hinsichtlich Komplexität und Anzahl beteiligter Trader-Partner sowie eine Erweiterung der zur Steuerung von Kooperationen verwendeten Strategien.

Zunächst wird in Abschnitt 5.1 das in Abschnitt 2.1 eingeführte allgemeine Modell kooperativer Subjekte auf den Fall von Trader-Instanzen angewendet (vgl. [Burg95]). Danach werden in Abschnitt 5.2 und 5.3 die Verfeinerung des auf dem Subjektmodell basierenden Kooperationsmodells und seine Anwendung auf die Kooperation von Tradern beschrieben.

5.1 Trader-Instanzen als kooperative Subjekte

Für die Anwendung des in Abschnitt 2.3 eingeführten allgemeinen Modells kooperativer Subjekte werden in den folgenden beiden Abschnitten sowohl das Potential als auch die Ziele von Tradern im Hinblick auf ihre Kooperationsbereitschaft behandelt.

5.1.1 Potential von Trader-Instanzen

Aus den Beschreibungen der Trader-Funktionalität in Kapitel 4 läßt sich das Potential von Trader-Instanzen wie folgt definieren:

Aktionen lassen sich in nutzer- (Abschnitt 4.3 - 4.5) und anbieterorientierte (Abschnitt 4.2.4) aufteilen. Sie dienen also einerseits der Behandlung von Kundenanfragen und andererseits der Registrierung von Dienstangeboten. In beiden Fällen ist eine Reihe von Optionen möglich: bei kundenorientierten Aktionen z. B. Angabe von Suchraum, -ordnung und Auswahlkriterien, synchrone oder asynchrone Behandlung, bei anbieterorientierten Aktionen z.B. Angabe eines Gültigkeitszeitraumes und von Kriterien, die die Menge der zugelassenen Kunden einschränken. Da unterschiedliche Optionen unterschiedliche Aktionen erfordern können, entsteht eine breite Palette möglicher Trader-Aktionen.

Es ist davon auszugehen, daß Trader-Instanzen jeweils nur über einen Ausschnitt aus der Gesamtheit der möglichen Aktionen verfügen. Z. B. wird nicht jeder Trader alle von Kunden oder Diensterbringern gewünschten Auswahlkriterien berücksichtigen können. Außerdem

können Unterschiede in qualitativer Hinsicht, wie z.B. der Vermittlungsgeschwindigkeit, der Fehlertoleranz oder den Kosten (s. u.) eines bestimmten Traders bestehen.

An **Ressourcen** werden bei diesen Aktionen z. B. Rechenleistung, Plattenplatz und Kommunikationseinrichtungen sowie Algorithmen (z.B. zur Optimierung) in Anspruch genommen. Da deren Kapazitäten begrenzt sind, ergeben sich Einschränkungen für das Potential der darauf zugreifenden Trader-Instanz. Insbesondere werden auch die Speicherfähigkeit und Vermittlungsgeschwindigkeit beeinflußt.

Kosten für die Trader-Leistung sind vom Systemadministrator zu spezifizieren. Dies kann aufgeschlüsselt nach den einzelnen Aktionen des Trader-Potentials und in Abhängigkeit von Kunden und Diensterbringern (z. B. deren Zugehörigkeit zu einer bestimmten Domäne) erfolgen. Denkbar wäre auch eine automatisierte Steuerung der Kosten nach marktwirtschaftlichen Prinzipien. Dieser Ansatz wird hier jedoch nicht weiter verfolgt.

Jeder Trader verfügt über **Wissen** über die Eigenschaften der bei ihm gespeicherten Dienstangebote, insbesondere deren Funktionalität und qualitative Eigenschaften, über die Wünsche von Kunden und zumindest teilweise über die Eigenschaften anderer Trader, d.h. über mögliche Kooperationspartner, wie in Abschnitt 3.3 beschrieben. Durch unterschiedliche Repräsentationen der Funktionalität von Diensten wird eine Vermittlung für unterschiedliche Benutzertypen ermöglicht (vgl. Abschnitt 4.2.2).

Insgesamt ergibt sich ein breites Spektrum von unterschiedlichen Trader-Ausprägungen. Bei geeigneter Kombination der unterschiedlichen Trader-Potentiale im Rahmen von Kooperationen können alle beteiligten Trader profitieren. Inwieweit ein Trader sein Potential, d.h. seine Aktionen, Ressourcen und sein Wissen, anderen Tradern zur Verfügung stellen bzw. Potential von anderen Tradern in Anspruch nehmen möchte, hängt von seinen Zielen ab.

5.1.2 Ziele von Trader-Instanzen

Auch die beim Trading verwendeten Ziele lassen sich ansatzweise aus den Ausführungen in Kapitel 2 und 4 herauslesen (vgl. auch Tabelle 1). Einige Beispiele, die für die Kooperationsbereitschaft von Tradern eine wesentliche Rolle spielen, werden im folgenden aufgelistet. Es zeigt sich, daß diese Ziele sich in drei Gruppen aufteilen lassen, die jeweils aus einem quantitativen und einem qualitativen Teil bestehen, und daß sie teilweise in Konflikt zueinander stehen. Im folgenden werden diese Ziele informell beschrieben. Ihre Quantifizierung erfolgt in Abschnitt 5.3.2.

- Kundenorientierte Ziele:
  a. Zuordnung jeder Kundenanfrage.
  b. Geeignete Kardinalitäten der Angebotsmengen, um Zuordnungen im Hinblick auf bestimmte Kriterien wie z. B. Minimierung von Antwortzeiten, Kosten bei der Benutzung oder die Einhaltung von zugesagten Abarbeitungsgarantien optimieren zu können.
- Ziele von Dienstanbietern:
  c. Eine möglichst große Menge potentieller Kunden, um die Wahrscheinlichkeit von Anfragen zu erhöhen.
  d. Geeignete Kardinalität der Menge potentieller Kunden, um Zuordnungen im Hinblick auf Kriterien von Dienstanbietern wie z. B. Maximierung des Durchsatzes bei einem Server oder einer Gruppe von Servern, Maximierung der für Dienstleistungen berechneten Kosten optimieren zu können.
- Trader-orientierte Ziele:

e. Hohe Qualität des Trader-Dienstes (d.h. bzgl. Speicherfähigkeit, Vermittlungsgeschwindigkeit und Fehlertoleranz).
f. Bei Bedarf Kooperation mit anderen Tradern.
g. Autonomie, auch bei einer Aufgabe von Rechten (vgl. BeRa91).

Entsprechend dieser Zielorientierungen bietet sich eine Aufteilung in benutzer- und servernahe Trader an, wie sie in Abschnitt 5.3.4 genauer behandelt wird. Aufgabe von Kooperationen zwischen diesen Tradern unterschiedlicher Ausprägung ist dann u.a. die Aushandlung von Kompromissen bei widersprüchlichen Zielen. Dabei spielen Gewichtungen eine wesentliche Rolle.

In bisherigen Arbeiten über Trader wurden keine Zielgewichtungen verwendet. Untersucht man die einzelnen Ziele hinsichtlich ihrer Gewichtungsmöglichkeit, so ergeben sich unterschiedliche Trader-Ausprägungen, je nachdem, ob funktionale (z. B. Ziel a.)), Optimierungs- oder Kostenaspekte im Vordergrund stehen. Für die Festlegung der Gewichtung gibt es drei Möglichkeiten:

- Durch den Systemadministrator.
- Durch eine Gruppe von Dienstanbietern bzw. einen Kundenkreis, je nachdem, ob es sich um einen anbieter- oder benutzernahen Trader handelt.
- Es besteht auch die Möglichkeit, übergeordnete Ziele vorzugeben und Gewichtungen anhand dieser Ziele im laufenden Betrieb zu adaptieren. Ein solcher Ansatz wird jedoch im Rahmen dieser Arbeit nicht weiter verfolgt.

Trader können sich sowohl in ihren Zielen als auch in deren Gewichtung unterscheiden.

5.2 Verfeinerung des Ablaufmodells für Kooperationen

Im folgenden werden allgemeingültige Konzepte für den Ablauf von Kooperationen aufgestellt, deren Anwendung auf Trader sich entweder unmittelbar ergibt oder die in Abschnitt 5.3 durch Beispiele näher erläutert wird. Wie bereits in Tabelle 2 in Abschnitt 2.1.1 gezeigt wurde, findet vor Beginn einer Kooperation eine Vorbereitungsphase statt, die durch die Entdeckung von Kooperationsbedarf ausgelöst wird. Die Bereitschaft eines Subjektes zur Kooperation entsteht dadurch, daß einerseits ein Kooperationsbedarf entdeckt wird und andererseits anhand von strategischen Vorgaben entschieden wird, ob dieser Bedarf im Verhältnis zu den anderen Zielen wichtig genug ist, um tatsächlich eine Kooperation einzugehen (interne Konfliktlösung, am Beispiel einer Trader-Kooperation in Abschnitt 5.3.2 beschrieben). Analog wird bei beliebigem Planungsbedarf im Verlauf einer Kooperation und insbesondere auch beim Wunsch nach Modifikation oder Abbau von Kooperationsbeziehungen vorgegangen, d.h. im Rahmen einer Planungsphase. Zu Beginn einer Kooperation oder einer Planungsphase kann außerdem eine Partnersuche erforderlich sein.

Die in Planungsphasen stattfindenden Verhandlungen werden in Abschnitt 5.2.1 genauer behandelt. Sie verlaufen nach dem Schema „Kooperations-/Planungsvorschlag an Partner - Antwort(en) zurück“.[1] Falls die Antwort negativ ausfällt oder allgemein ein Konflikt aufgetreten ist, kann die Verhandlung dadurch fortgesetzt werden, daß mehrfach Vorschläge und Gegenvorschläge ausgetauscht werden. Die Anzahl derartiger Austauschvorgänge und ganz allgemein die Kriterien zur Erstellung und Bewertung von Vorschlägen werden nach strategischen Gesichtspunkten bestimmt (vgl. [Kirs94]).

1. Weitere Verhandlungselemente wie z.B. Drohungen, Versprechungen oder Konzessionen (vgl. [RoZl94]) werden vorerst nicht eingesetzt.

Sind die Verhandlungen zu Beginn einer Kooperation erfolgreich beendet, so werden die zugehörigen Kooperationsbeziehungen aufgebaut. Anschließend können die ausgehandelten Aktionen und Interaktionen wie geplant im Rahmen einer Realisierungsphase ausgeführt werden. Tritt dabei ein Konflikt auf, so ist dieser durch erneute Verhandlungen aufzulösen oder die Kooperation muß abgebrochen werden. Eine erneute Planungsphase kann auch dann erforderlich werden, wenn einige der angestrebten Ziele erreicht sind oder Änderungen der Umweltbedingungen eingetreten sind. Demzufolge wechseln sich Planungs- und durch Koordination überwachte Realisierungsphasen gegenseitig ab, bis alle Ziele erreicht sind oder die Kooperation abgebrochen werden muß.

5.2.1 Verhandlungen

Um eine Kooperation zu initiieren, versendet dasjenige Subjekt, das einen Kooperationsbedarf entdeckt hat, Kooperationsvorschläge an andere Subjekte. In analoger Weise können Modifikationen und der Abbau einer Kooperationsbeziehung durch einen Planungsvorschlag eingeleitet werden.

Allgemein können Kooperations- und Planungsvorschläge folgende Elemente enthalten:

- Alle diejenigen Potentialteile, bezüglich derer das initiierende Subjekt einen Mangel oder Überfluß entdeckt hat und die es anderen gemäß seiner Kooperationsstrategie, möglicherweise mit bestimmten Restriktionen, bekanntgeben möchte. Die Angabe dieser Potentialteile erfolgt entweder direkt oder implizit über die Ziele, bei deren Erreichung ein Defizit entdeckt wurde (in diesem Fall sind fehlende Potentialteile nicht unbedingt namentlich bekannt). Zu den Randbedingungen für einen Austausch von Potentialteilen gehören z. B. Kosten oder zeitliche Restriktionen, die die Gültigkeit von Angeboten oder Nutzungswünschen festlegen.
- Die Zuteilung von Potentialteilen oder Zielen zu potentiellen oder existierenden Partnern.
- Die zu verwendende Koordinationsstruktur (s. Abschnitt 5.2.3).
- Das zur Ergebniszusammenführung verwendete Verfahren (vgl. [DuLe91]).

Je nach Umfang des Wissens über andere Subjekte können Vorschläge zur Initiierung von Kooperationen möglicherweise gezielt verschickt werden (vgl. Abschnitt 5.1.1). Dazu muß je fraglichem Potentialteil bekannt, welches Subjekt darin einen Überfluß bzw. Mangel aufweist. Ansonsten sind Vorschläge an alle per Namen und Adresse bekannten Subjekte zu versenden. Planungsvorschläge werden i.a. an die bereits an der Kooperation beteiligten Partner verschickt, außer für den Fall, daß zusätzliche Teilnehmer ins Team aufgenommen werden sollen, dann werden auch diese mit einbezogen.

Eingehende Vorschläge zur Initiierung oder Modifikation von Kooperationen werden einerseits dahingehend überprüft, ob das empfangende Subjekt (noch) als Partner geeignet ist, d.h. ob es aufgrund seines momentanen Potentials überhaupt in der Lage ist, den gemeldeten Mangel oder Überfluß auszugleichen. Falls dies zutrifft, ist anhand strategischer Überlegungen und Vergleiche mit seinen sonstigen Zielen festzustellen, ob oder inwieweit es einen derartigen Ausgleich befürwortet. Dabei sind auch die möglicherweise im Vorschlag enthaltenen Randbedingungen zu berücksichtigen (z. B., ob die Kosten oder die Gültigkeitsdauer akzeptiert werden können). In ähnlicher Weise werden auch die vorgeschlagene Koordinationsstruktur und das zur Ergebniszusammenführung verwendete Verfahren durch Vergleich mit der eigenen Kooperationsstrategie geprüft.

Das Ergebnis aller dieser Überlegungen schlägt sich in der Antwort an den Initiator der Kooperation oder Modifikation nieder, indem dessen Vorschlag entweder vollständig bejaht oder verneint wird oder indem ein Gegenvorschlag an ihn verschickt wird. Außer bei einer positiven Antwort kann es zu einer Fortsetzung der Verhandlung kommen, bei der je nach Kooperationsstrategie der beteiligten Subjekte mehrfach Vorschläge mit Modifikationen der ursprünglichen Fassung ausgetauscht werden. Dieser Fall entsteht durch einen Konflikt zwischen den beteiligten Subjekten und wird in Abschnitt 5.2.2 gesondert behandelt.

In Abhängigkeit von den Kooperationsstrategien der beteiligten Subjekte erlangen die ausgetauschten Vorschläge entweder nur dann Gültigkeit, wenn alle Partner zustimmen, oder es reicht schon eine gewisse Mindestzahl an Ja-Stimmen aus. Diejenigen Subjekte, die die getroffenen Vereinbarungen nicht befürworten, können unter Verwendung eines Veto-Verfahrens, wie es z. B. in [Wong93] beschrieben wurde, versuchen, ihren „Willen“ doch noch durchzusetzen.

5.2.2 Externe Konfliktlösung

Widersprechen die in einem empfangenen Kooperationsvorschlag enthaltenen Fakten den Zielen des Subjektes in irgendeiner Weise und ist es trotzdem an der Kooperation bzw. ihrer Modifikation interessiert, so kann es durch einen Gegenvorschlag versuchen, eine Einigung zu erzielen. Gegenvorschläge werden erstellt durch Ändern der ursprünglichen Vorschläge in einem mit Zielen und Kooperationsstrategie verträglichen Sinne. Dabei gibt die Strategie vor, in welche Richtung und mit welcher Schrittweite modifiziert wird. Außerdem wird durch die Strategie festgelegt, wie häufig maximal ein Gegenvorschlag erstellt wird, bevor das Subjekt die Konfliktlösung als nicht erfolgreich verbucht (vgl. den aus dem Alltag bekannten „Geduldsfaden“).

Um dies zu konkretisieren, sei ein Beispiel angeführt. Unterscheiden sich Angebot und Forderung in Bezug auf die Kosten zur Nutzung eines bestimmten Potentialteiles, so wird das eine Subjekt durch Erhöhen seines Angebotes und das andere durch Absenken seiner Forderung versuchen, zu einer Einigung zu kommen, es wird also gefeilscht. Die Schrittweite dieser Modifikationen und die Beendigung des Feilschens werden durch die Strategie bestimmt und sind dem Partner a priori nicht bekannt. Dadurch wird die Kompromißfindung indeterministisch.

Ein weiteres Beispiel, das speziell bei Trader-Kooperationen auftritt, wird in Abschnitt 5.3.4 behandelt.

5.2.3 Verhandlungsformen und Koordinationsstrukturen

Verhandlungen können unterschiedlich komplex sein (vgl. Abschnitt 5.2). Allgemein lassen sie sich nach folgenden Kriterien klassifizieren:

- Anzahl der in Vorschlägen enthaltenen Elemente allgemein und speziell der darin enthaltenen Potentialteile oder Ziele
- Möglichkeit von Gegenvorschlägen und Veto-Recht sowie Anzahl der Verhandlungsschritte.

Bild 9 zeigt die beiden Grenzformen, den einfachsten und den am meisten komplexen Fall einer Verhandlung. Dazwischen ist jede beliebige Verhandlungsform denkbar.

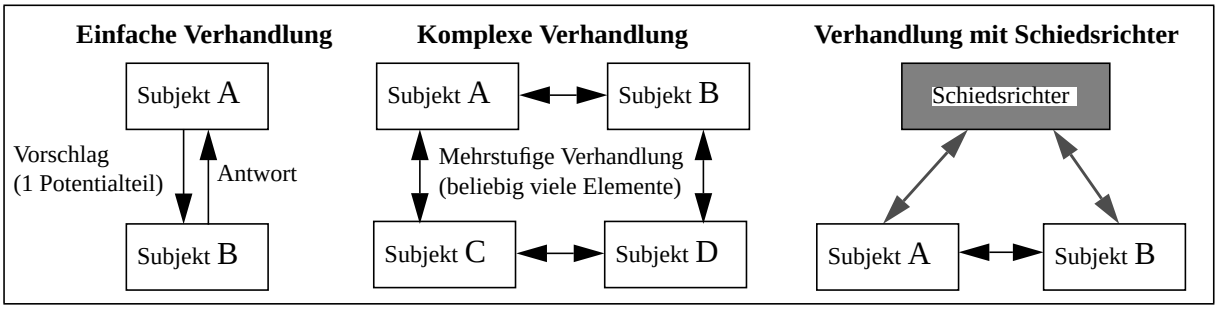


Abbildung 9. Verhandlungskomplexität

Eine weitere Variante für Verhandlungen ergibt sich im Zusammenhang mit der Koordinationsstruktur. Diese kann z. B. multilateral oder hierarchisch sein und muß im multilateralen Fall die Beschlußform festlegen, d.h. ob einstimmige Beschlüsse erforderlich sind oder ob ein Mehrheitsbeschluß ausreicht. Je nach der Anzahl Partner im Kooperationsteam können Mehrheitsbeschlußformen möglicherweise zu einem Patt führen. Eine Auflösung kann dann durch Erweiterung der Verhandlung um den Anruf eines Schiedsrichters erfolgen (vgl. Bild 9).

5.2.4 Kooperationsstrategien

Ob ein Kooperations- oder Planungsbedarf tatsächlich zu einer Kooperation führt oder ob er hinter anderen Zielen zurückstehen muß, ist anhand der vorgegebenen Kooperationsstrategie zu entscheiden. Diese Strategie dient dazu, Richtlinien für alle im Zusammenhang mit Kooperationen auftretenden Probleme vorzugeben und enthält dementsprechend:

- Gewichtung von Kooperations- oder Planungsbedarf und anderen Zielen, um eine interne Konfliktlösung zu ermöglichen.
- Richtlinien zur Erstellung von Koordinations- und Gegenvorschlägen.
- Behandlung von Koordinations- und Gegenvorschlägen, d.h. Kriterien für Annahme/Ablehnung von Vorschlägen/Gegenvorschlägen, maximal zugelassene Anzahl Verhandlungsvorgänge bis zur endgültigen Entscheidung.
- Beschlußform, z.B. die erforderliche Anzahl Ja-Stimmen für Teamakzeptanz.
- Richtlinien für Veto-Recht und Verwendung eines Schiedsrichters.

Welche Parameter dabei eine Rolle spielen, hängt von der konkreten Anwendung ab. Als Beispiel wird im folgenden die Kooperation zwischen Tradern behandelt.

5.3 Kooperation zwischen Tradern

Um Kooperationen zwischen Tradern zu beschreiben, ist das in Abschnitt 5.2 erläuterte Modell auf Trader anzuwenden. Dazu sind die auslösenden Ereignisse von Kooperationsphasen, phasenspezifische Aktionen und Interaktionen sowie deren Ergebnisse bei Tradern zu identifizieren.

5.3.1 Kooperations- und Planungsbedarf bei Tradern

Gemäß Abschnitt 3.3 wurde Kooperationsbedarf bei Tradern in bisherigen Arbeiten dadurch entdeckt, daß eine Kundenanfrage nicht behandelt werden kann, da der betroffene Diensttyp

nicht bekannt ist. Wie die in Abschnitt 5.1.2 beschriebenen Trader-Ziele zeigen, wird dadurch jedoch nur ein Spezialfall ungenügender Quantität des Trader-Dienstes abgedeckt.

Allgemein entsteht Kooperationsbedarf immer dann, wenn ein Trader im Verhältnis zu seinen Zielen entweder über zu wenig oder über zu viel Potential verfügt. Dabei äußert sich ein Potentialmangel in mangelnder Quantität oder Qualität des erbrachten Trader-Dienstes oder in mangelnder Qualität des vermittelten Dienstes. Im einzelnen ergeben sich dadurch folgende Fälle für Diskrepanzen zwischen Potential und Zielen:

- Wissensmangel bzgl. Diensterbringern oder Tradern.
- Aktionen-/Ressourcenmangel, z.B. mangelnde Geschwindigkeit oder Fehlen des Optimierungsalgorithmus zur Behandlung eines bestimmten Auswahlkriteriums (vgl. Abschnitt 5.1.1).
- Falsche Proportionen zwischen Diensterbringer- und Kundenmenge (z.B. Kundenmenge zu klein, um eine akzeptable Anzahl Zuordnungen vornehmen zu können oder sogar gemäß serverorientierter Kriterien zu optimieren).

Es gibt also eine Reihe von Gründen, eine Kooperation zwischen Tradern zu favorisieren.

Diese Gründe können sich im Verlauf einer tatsächlich eingegangenen Kooperation ändern oder sie können ganz entfallen. In beiden Fällen entsteht ein Planungsbedarf, der auf eine Modifikation oder Auflösung von Kooperationsbeziehungen abzielt. Sowohl der initiale Kooperations- als auch der möglicherweise später auftauchende Planungsbedarf sind unter Verwendung der im nächsten Abschnitt beschriebenen Kooperationsstrategie zu untersuchen und gegebenfalls geeignet weiterzubehandeln.

5.3.2 Interne Konfliktlösung bei Tradern

Im folgenden wird ein Beispiel behandelt, in dem ein interner Konflikt zwischen der Bereitschaft zum Aufbau von Kooperationsbeziehungen und Zielen, die dieser Bereitschaft entgegenstehen, auftreten kann. Da die Initiierung und Verwaltung von Kooperationsbeziehungen mit einem erheblichen Aufwand verbunden ist, kann die Qualität des Trader-Dienstes durch das Eingehen von Kooperationen beeinträchtigt werden (z. B. durch eine Verlängerung der Antwortzeiten bei der Vermittlung). Damit können sich das Ziel, einen möglichst guten Trader-Dienst zu erbringen, und dasjenige, bei Bedarf eine Kooperation einzugehen, widersprechen. Je nach der durch die Strategie vorgegebenen Gewichtung wird entschieden, welches Ziel zugunsten des anderen aufzugeben ist.

Grundsätzlich sind in diesem Zusammenhang zwei Strategien denkbar (vgl. [Burg95]):

- Kooperationsbereitschaft ist höher gewichtet als die Dienstqualität des Traders, d.h. es wird so viel wie möglich kooperiert, um alle an Dienstnutzern und Diensterbringern orientierten Ziele entsprechend ihrer Gewichtungen zu erreichen (vgl. das Beispiel eines Wissenschaftlers, der die Kosten einer Fernleihe auf sich nimmt, um einem besonders wichtigen Literaturverweis nachgehen zu können).
- Die Dienstqualität des Traders ist höher gewichtet als seine Kooperationsbereitschaft, d.h. es wird zwar so viel wie möglich kooperiert, um allen Dienstnutzern und Diensterbringern gerecht zu werden, aber nur solange, wie die Dienstqualität nicht darunter leidet. In diesem Fall wird eine Anpassung der Kooperationsbereitschaft benötigt.

Um derartige Strategien realisieren zu können, sind die in Abschnitt 5.1.2 eingeführten Ziele durch geeignete Wahl von Parametern zu quantifizieren. Für die zweite der genannten Strategien wird dabei insbesondere auch die Adaptierbarkeit von Parametern benötigt.

Da das in Abschnitt 5.1.2 eingeführte, am Dienstnutzer orientierte Ziel a., jede Anfrage zu beantworten, nicht adaptiert werden kann, ist es durch das folgende schwächere, dafür aber variable Ziel zu ersetzen:

- Wenn eine bestimmte Anzahl $N^{max}_{unbekannt}$ von Dienstnutzeranfragen bezüglich eines bestimmten unbekannten Diensttyps S erreicht ist, ist eine Kooperation mit anderen Tradern einzuleiten, um denjenigen Diensterbringer ausfindig zu machen, der den entsprechenden Dienst erbringt (vgl. das Beispiel eines Supermarktes, in dem ein Kunde nach einem Produkt fragt, das momentan nicht geführt wird; erst wenn sich derartige Anfragen häufen, wird die Marktleitung eine Aufnahme des Produktes erwägen).

In diesem Ziel kann der Parameter $N^{max}_{unbekannt}$ variiert werden. Für den Fall $N^{max}_{unbekannt} = 1$, stimmt die schwächere Form mit der ursprünglichen überein.

Die beiden, in Abschnitt 5.1.2 genannten Optimierungsziele b. und d. stehen in engem Zusammenhang mit einer Lastbalancierung über Domänengrenzen hinweg. Da dies möglicherweise zu organisatorischen und Heterogenitätsproblemen führt, ist es sinnvoll, ein einfaches Lastverteilungsverfahren mit Schwellwerten zu verwenden (vgl. die Klassifikation von Zuordnungsverfahren in [Burg90]). Dazu werden die folgenden Parameter benötigt:

- Oberer Schwellwert $UT_{server}(S)$ für die Last eines Dienstleisters, der den Dienst von Typ S erbringt.
- Unterer Schwellwert $LT_{trader}(S)$ für Anfragen von Dienstnutzern bezüglich des Diensttyps S und unterer Schwellwert $LT_{server}(S)$ für die Last eines Dienstleisters, der den entsprechenden Dienst erbringt.
- Zeitliche Dauer T, die angibt, wie lange Ausdrücke mit den oben definierten Parametern gelten müssen, bevor sie berücksichtigt werden können.

Diese Parameter werden folgendermaßen zur Realisierung der Kooperationsstrategien eingesetzt:

- Die Anfragen bezüglich eines bestimmten unbekannten Diensttyps S werden gezählt und mit $N^{max}_{unbekannt}$ verglichen. Falls beide Werte übereinstimmen, wird eine Kooperation eingeleitet, um einen Diensterbringer für S ausfindig zu machen.
- Die Last aller Diensterbringer, die einen bestimmten Dienst vom Typ S leisten können, wird mit $UT_{server}(S)$ verglichen. Wenn die Last den Schwellwert während des gesamten Zeitraumes T übersteigt, wird eine Kooperation eingeleitet, um weitere Diensterbringer für S ausfindig zu machen.
- Die Nutzungen eines bestimmten Dienstes vom Typ S werden gezählt und mit $LT_{trader}(S)$ verglichen. Wenn der untere Schwellwert während des gesamten Zeitraumes T nicht überschritten wurde, wird die Last der zugehörigen Diensterbringer mit $LT_{server}(S)$ verglichen. Wenn auch dieser untere Schwellwert während des gesamten Zeitraumes T nicht überschritten wurde, wird eine Kooperation eingeleitet, um anderen Tradern den Diensttyp S anzubieten.

Bei einer anfänglichen Konfiguration wird $N^{max}_{unbekannt}$ auf eins gesetzt und die Schwellwerte erhalten umgebungsabhängige, vordefinierte Werte. Für die zweite der oben genannten Strategien kann die Kooperationsbereitschaft durch Modifikation dieser Werte beeinflußt werden.

Dazu wird die Dienstqualität des Traders beobachtet und die Kooperationsbereitschaft entsprechend angepaßt: bei steigender Dienstqualität wird die Kooperationsbereitschaft erhöht, bei sinkender wird sie verringert. Der Umfang dieser Modifikation kann entweder durch eine vordefinierte Konstante festgelegt werden oder vom Gradienten der Dienstqualität abhängen. Bild 10 zeigt diesen Mechanismus am Beispiel der mittleren Antwortzeit des Traders.

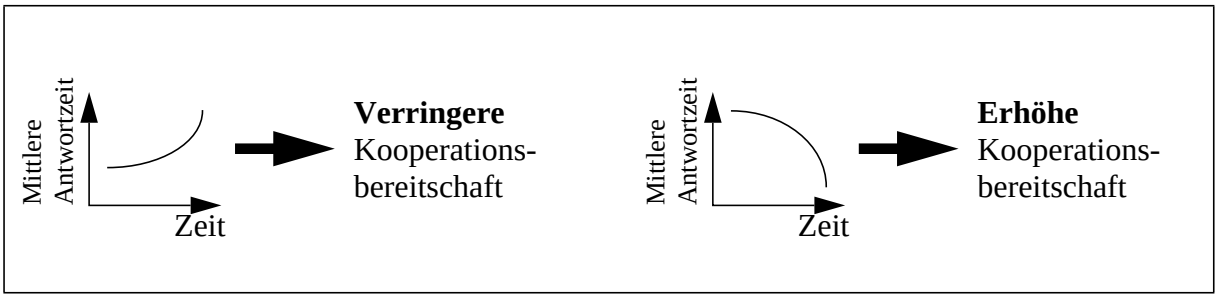


Abbildung 10. Anpassung der Kooperationsbereitschaft

5.3.3 Zwischen Tradern ausgetauschte Kooperationsvorschläge

Erste Ansätze zu Kooperationsvorschlägen sind in [BeRa91] und [ISO93] zu finden. Die dort vorgeschlagenen Formen für Kooperationsvorschläge beschränken sich allerdings auf den Umfang von Wissen über Dienstleister und die darauf zugelassenen Aktionen. Gemäß Abschnitt 5.1 sind darüber hinaus die folgenden Potentialteile in einem Kooperationsvorschlag möglich:

- Aktionen, die ein Trader für andere ausführen möchte bzw. durch einen anderen ausführen lassen muß (z. B. wegen fehlender Ressourcenkapazität oder mangels des zur Berücksichtigung von Auswahlkriterien benötigten Algorithmus). Das dabei über Dienstleister benötigte Wissen befindet sich beim beauftragenden Trader und ist dem oder den zur Ausführung vorgesehenen in geeigneter und mit der Strategie verträglicher Weise mitzugeben (vgl. den Fall einer Unternehmensberatung, wobei die wirtschaftliche Lage des Auftraggebers zu schildern ist).
- Durch die Strategie vorgegebener Prozentsatz an Kunden oder Diensterbringern, den ein Trader an andere abgeben bzw. von anderen übernehmen will.

5.3.4 Beispiele für unterschiedliche Verhandlungsformen zwischen Tradern

Je nach den Eigenschaften der beteiligten Trader lassen sich unterschiedliche Ausprägungen von Kooperationen unterscheiden. Ein Beispiel wurde bereits in Abschnitt 5.1.2 angesprochen. Dort wurden anhand der Zielorientierungen benutzer- und servernahe Trader identifiziert. Eine Kooperation zwischen derartigen Trader-Ausprägungen erfordert i.a. eine aufwendigere Kooperationsform als eine Kooperation zwischen relativ gleichartigen Tradern. Außerdem wirken sich auch der Umfang des Kooperationsbedarfes sowie die Anzahl der beteiligten Partner auf die Kooperationsform aus.

Allgemein lassen sich Kooperationen zwischen Tradern nach folgenden Kriterien klassifizieren:

- Einsatz von Verhandlungen.
- Komplexität der Verhandlung (vgl. Abschnitt 5.2.3).

In [Burg95] wurde ein Verfahren vorgeschlagen, in dem ähnlich wie bei der internen Konfliktlösung (Abschnitt 5.3.2) in Abhängigkeit von der aktuellen Dienstqualität des Traders eine Auswahl unter unterschiedlich komplexen Verhandlungsformen getroffen wurde.

Eine Möglichkeit der Kooperation zwischen Tradern ist die *implizite Trader-Kooperation.* Dazu müssen die folgenden Bedingungen erfüllt sein:

- Das Wissen über andere Subjekte und ihre Zustandsänderungen bezüglich des kooperationsrelevanten Teiles von Potential und Zielen im Lauf der Zeit muß entweder vollständig sein oder Wissensmangel kann durch Kompensationsmethoden ausgeglichen werden.
- Der Zugriff auf fremdes Potential kann ohne eine Kommunikation mit den zugehörigen Eigentümern erfolgen.

Ein Beispiel für eine implizite Kooperation zwischen Tradern in einem lokalem Netz wurde in [Burg90] beschrieben. Dort wurde das unvollständige Wissen über den aktuellen Zustand des Potentials mit Hilfe stochastischer Programmierung kompensiert. Allgemein ist jedoch festzuhalten, daß diese Art der Kooperation sich nur innerhalb von geeignet definierten Domänen anwenden läßt, wobei Randbedingungen extern auszuhandeln sind und auch Zugriffsrechte anderweitig abgeprüft werden müssen.

Für den Fall, daß jeweils nur eine geringe Anzahl von Anfragen/Angeboten/potentiellen Trader-Kooperationspartnern betroffen ist, Typen und Kontexte homogen sind und keine Sicherheits- und Abrechnungsprobleme auftreten können, läßt sich die *einfache unilaterale Trader-Kooperation* einsetzen. Dazu kann derjenige Trader, der einen Potentialmangel oder -überfluß beheben möchte, die normale Schnittstelle eines anderen Traders verwenden.

In komplexeren Fällen, in denen die Anzahl betroffener Anfragen/Angebote/potentieller Trader-Kooperationspartner nicht bekannt ist oder Probleme durch Heterogenität (Typ, Kontext) oder im Zusammenhang mit Sicherheit und Abrechnung auftreten können, ist dagegen eine *allgemeine multilaterale Trader-Kooperation* mit einer der in Abschnitt 5.2.3 beschriebenen Verhandlungsformen zu verwenden.

5.3.5 Externe Konfliktlösung zwischen benutzer- und servernahen Tradern

In Abschnitt 5.1.2 wurde bereits darauf hingewiesen, daß sich aufgrund unterschiedlicher Zielorientierungen von Tradern eine Aufteilung in benutzer- und servernahe Trader anbietet. Dieser Fall bildet einen Spezialfall für eine allgemeine multilaterale Trader-Kooperation, da benutzernahe Trader immer einen Potentialmangel aufweisen und servernahe ihr Potential anbieten.

Bei Kooperationen zwischen Tradern dieser beiden unterschiedlichen Typen kann es außer zu dem bereits in Abschnitt 5.2.2 erwähnten Kostenkonflikt z. B. auch zum folgenden Zielkonflikt kommen: ein benutzernaher Trader wird die Vermittlung eines besonders leistungsstarken Diensterbringers favorisieren während ein von Diensterbringern beauftragter Trader auch leistungsschwache Diensterbringer vermitteln wird, um deren Auslastung zu erhöhen und leistungsstarke zu entlasten. Zur Konfliktlösung ist dementsprechend auf der Skala der „Leistungsstärke“ von Diensterbringern zu modifizieren. Je nach Kooperationsstrategie der beteiligten Partner besteht der Kompromiß also in der Festlegung auf einen Diensterbringer einer ganz bestimmten Leistungsstärke.

5.4 Zusammenfassung

Es wurde ein gegenüber bisherigen Ansätzen zu Trader-Kooperationen erweiterter Rahmen abgesteckt. Insbesondere wurden die Elemente von Kooperationsstrategien eingeführt. Außerdem erfolgte eine ausführliche Behandlung unterschiedlicher Kooperationsformen, vor allem der dabei benötigten Lösung von Konflikten. Darüberhinaus wurde die Erweiterung von Teams auf mehr als zwei Mitglieder betrachtet.

6. Entwurf des MELODY-Traders

Das MELODY-System unterstützt einen offenen Markt an Diensten durch die Realisierung einer einheitlichen Dienstverwaltung. Diese Marktinfrastruktur besteht aus zwei eigenständigen Teildiensten mit genau definierter Funktion. Der Trader-Dienst sorgt für die Dienstvermittlung, der Managementdienst für die Kontrolle der Dienstkomponenten und für die Überwachung von verteilten Diensten. Beide Grunddienste sind unabhängig voneinander, arbeiten jedoch eng zusammen. Im folgenden Kapitel werden die Entwurfsentscheidungen für den Trader-Dienst erläutert. Kapitel 7 behandelt den Managementdienst und Kapitel 8 beschreibt die Implementierung dieses Trader-Dienstes und seiner Integration mit dem Managementdienst.

Bevor die Entwurfsentscheidungen für den MELODY-Trader dargelegt werden können, sind zunächst die Anforderungen an den Trader und die inhaltlichen Schwerpunkte der Entwicklungsarbeit zu definieren. Allgemeine Anforderungen werden in Abschnitt 6.1.1 aufgelistet, wobei der Einfluß von Umgebungsparametern auf Dienstnutzungen besonders hervorgehoben wird. Anschließend wird auf die beiden Schwerpunkte genauer eingegangen: Behandlung des Konsistenzproblems bei Zustandsinformationen durch benutzerdefinierte Unschärfebedingungen (6.1.2) und Kooperation zwischen Tradern (6.1.3). In den darauf folgenden Abschnitten wird der Entwurf des MELODY-Traders erläutert. Dazu werden der Trader-Dienst definiert (6.2.1) und die Anfragesprache beschrieben (6.2.2), anschließend wird auf die Behandlung von dynamischen Attributen (6.2.3) und auf die Verwaltung von Auswahlregeln eingegangen (6.2.4). Am Ende wird beschrieben, wie Anforderungen des Diensterbringers an seine Dienstnutzer behandelt werden (6.2.5) und wie Garantien für Dienstqualitäten vom Trader ausgehandelt werden (6.2.6).

6.1 Anforderungen an den MELODY-Trader

6.1.1 Allgemeine Anforderungen

Der MELODY-Trader bildet einen Teil einer offenen Marktinfrastruktur und muß sich daher an existierenden Standards orientieren. In diesem Fall ist es der ISO Draft Standard für den ODP Trader ([ISO92], [ISO94], [ISO95]). Dieser war in den letzten Jahren ständigen Änderungen unterworfen. Aus diesem Grund beruht der ursprüngliche Entwurf des Traders auf der Version von 1992 ([ISO92]), die sich teilweise erheblich vom aktuellen Entwurf unterscheidet. Durch die Weiterentwicklung des eigenen Traders sind die Unterschiede jedoch nicht allzu groß, so daß ein Übergang auf den endgültigen Standard leicht möglich sein sollte.

Der realisierte Dienst nach dem Klient-Service Modell muß auf einer allgemein verfügbaren Systemplattform beruhen. Diese soll ausreichende Möglichkeiten für eine effiziente Implementierung der Trader-Funktionen in einem heterogenen Umfeld gewährleisten. Dabei soll die Trader-Schnittstelle jedoch unabhängig von der gewählten Systemplattform sein. Weiterhin müssen ausreichende Sicherheitsfunktionen (sowohl zur Authentisierung als auch zur Autori-

sierung) realisierbar sein. Sowohl der ursprünglich gewählte SUN-RPC als auch die spätere Plattform DCE bieten dafür gute Möglichkeiten. Ein objekt-orientierter Entwurfs- und Implementierungsansatz dient zur Trennung der Trader-Funktionalität von der gewählten Systemplattform und zur einfachen Wiederverwendbarkeit der Trader-Komponenten. Für den menschlichen Benutzer bietet eine graphische Benutzungsoberfläche eine einfache Nutzung des Trader-Dienstes.

Für eine ausreichende Funktionalität der Dienstvermittlung müssen komplexe Auswahlkriterien ausgewertet und unterschiedliche Optimierungskriterien berücksichtigen werden. Einerseits können Auswahlkriterien von jedem einzelnen Nutzer individuell vorgegeben werden, andererseits müssen Auswahlkriterien auch einheitlich für eine große Anzahl von Nutzern definiert werden können. Aus diesem Grund werden Auswahlregeln eingeführt, die im Trader verwaltet werden und anstelle eines individuellen Auswahlkriteriums angegeben werden können. Im Gegenzug müssen bei der Vermittlung auch Anforderungen des Diensterbringers an den Dienstnutzer berücksichtigt werden. Die hierfür notwendigen Charakterisierungsattribute sind vom System bereitzustellen.

Einen weiteren Schwerpunkt der Untersuchungen bilden diejenigen Eigenschaften eines Diensterbringers, auf denen die Dienstauswahl beruht. Dabei sollen neben Eigenschaften des Dienstanbieters selbst auch Eigenschaften seiner Ablaufumgebung und der Netzanbindung zum Klienten berücksichtigt werden. Dies erfordert einerseits das Zusammenspiel des Trading-Dienstes mit einem Managementdienst, der dem Trader dynamische Attribute über das System bereitstellt, und andererseits eine besondere Behandlung im Trader, um die mit dem entfernten Zugriff verbundenen Verzögerungen so weit wie möglich zu reduzieren.

Es versteht sich von selbst, daß Trader-Entscheidungen auf möglichst aktueller Information beruhen müssen. Die hieraus resultierenden Anforderungen sind in Abschnitt 6.1.2 ausführlicher dargestellt. In einem dynamischen Umfeld mit sich ändernden Randbedingungen ist es für einen Dienstnutzer wichtig, eine bestimmte Dienstqualität garantiert zu bekommen. Hierzu sind gesonderte Untersuchungen zur Aushandlung der geforderten Dienstqualitäten und zur Reservierung der benötigten Ressourcen notwendig. Insbesondere die gleichzeitige Verhandlung mit unterschiedlichen Dienstanbietern stellt noch ein weitgehend unerforschtes Gebiet dar. Die hierzu angestellten Überlegungen sind nicht abgeschlossen, so daß in diesem Bericht nur erste Ansätze geschildert werden.

6.1.2 Benutzerdefinierte Unschärfebedingungen (Aktualität von Zuständen)

Die Entscheidungen des Traders beruhen auf dem aktuellen Zustand eines verteilten Systems. Aufgrund der fehlenden globalen Sicht ist jedoch eine absolute Konsistenz zwischen den Werten der berücksichtigten Entscheidungsparameter und den aktuellen Werten im System nicht möglich. Eine solche Konsistenz und damit die vollkommene Aktualität ließe sich höchstens durch Sperrkonzepte erreichen, deren Einsatz jedoch in der Mehrzahl der Fälle unrealistisch ist. Beispielsweise ist es nicht wünschenswert, die Warteschlangen aller geeigneten Dienstanbieter solange zu sperren, bis der Trader eine Entscheidung gefällt hat. Weiterhin unterstützen viele der benötigten Entscheidungsparameter kein Sperrkonzept. Beispielsweise beeinflußt die freie Bandbreite eines Netzwerkes die Qualität eines Kommunikationsdienstes. Bei den heutzutage vorherrschenden Netzwerkarchitekturen ergibt sich die freie Bandbreite jedoch aus dem Verhalten der kommunizierenden Anwendungen. Dies läßt sich aber nicht durch den Trader-Prozeß festlegen.

Aus diesem Grund wird anstelle eines absoluten Konsistenzbegriffs für die Aktualität von Informationen ein abgeschwächter Begriff eingeführt. Ein schwacher Konsistenzbegriff fordert, daß jede Änderung der Information nach einer gewissen (in absoluten Angaben jedoch undefinierten) Zeitspanne beim Trader sichtbar wird. Wie lange dies dauert, ist jedoch nicht Teil des Konsistenzbegriffs. Daher kann eine schwache Konsistenz in der Größenordnung der Nachrichtenlaufzeiten der Netzwerke liegen, aber auch ein deutlich größeres Zeitintervall umfassen. Zur Realisierung eines abgeschwächten Konsistenzbegriffs gibt es prinzipiell zwei alternative Strategien:

- **Direkter Zugriff**
  Jeweils zum Vermittlungszeitpunkt wird auf die benötigte Information zugegriffen. Der beobachtete Zustand ist dabei etwa um eine Nachrichtenlaufzeit vom tatsächlichen entfernt.
- **Änderungsmeldungen**
  Jede Änderung eines Attributes wird dem Trader umgehend mitgeteilt. Auch hier befindet sich der beobachtete Zustand etwa eine Nachrichtenlaufzeit vom aktuellen Zustand entfernt.

Beide Lösungen haben Vor- und Nachteile: Der direkte Zugriff erzeugt ein u.U. burst-artiges Nachrichtenaufkommen und verzögert die Dienstvermittlung um die Zugriffszeiten auf die dynamischen Attribute. Änderungsmeldungen belasten das System u.U. mit einer Reihe von überflüssigen Meldungen, die vom Trader nicht benötigt werden. Welche Strategie besser geeignet ist, hängt außerdem vom Verhalten der Trader-Nutzer und der Änderungshäufigkeit eines Attributs ab.

Zu berücksichtigen ist außerdem, daß unterschiedliche Attribute unterschiedliche Bedeutung für den Vermittlungsprozeß haben. In vielen Fällen kann mit teilweise veralteter Information gearbeitet werden, ohne daß sich die Entscheidungsqualität der Dienstvermittlung grundlegend verschlechtert. Das hierzu benötigte Anwendungswissen muß jedoch gezielt und fallspezifisch angegeben werden.

Für das MELODY System wird daher eine automatische Adaption des Zugriffsverhaltens an die gültigen Zugriffs- und Änderungscharakteristiken gefordert. Dabei sollen vom Benutzer spezifizierbare Unschärfebedingungen berücksichtigt werden. Zu solchen Unschärfebedingungen zählen Anforderungen an das Alter der lokalen Kopie, an die Anzahl der unberücksichtigten Änderungen sowie Abhängigkeiten zwischen verschiedenen Attributen. Das Zugriffssystem entscheidet, welche Strategie zur Aktualisierung der Werte verwendet wird. Diese Entscheidungen müssen transparent für den Trader und den Dienstanbieter getroffen werden. Die hierfür notwendigen Systemmechanismen sind unabhängig von der eigentlichen Dienstnutzung zu gestalten.

6.1.3 Kooperation zwischen Trader-Instanzen

Trader-Instanzen verwalten Dienste einer bestimmten Trader-Domäne. Informationen über andere Domänen und ihre Trader können zunächst dem DCE-Name-Directory entnommen werden. Darüberhinaus können Trader Angebote über ihren Dienst an andere Trader verschikken, so daß Trader-Dienste gemeinsam mit den Anwendungsdiensten im Trader verwaltet werden. Diese beiden Verfahren sollen zur Partnersuche zwischen Tradern eingesetzt werden. Die Kooperation an sich soll nach den in Abschnitt 5.3 eingeführten Konzepten verwirklicht werden. Je nach vorgegebener Domänenstrategie werden Kooperationen statisch oder situations- und fallbezogen eingegangen. Dazu müssen die Trader-Strategien für jede Domäne einzeln definierbar sein. Die zur Umsetzung der Strategien benötigten Mechanismen sind so zu realisieren, daß ein direkter Übergang zwischen verschiedenen Kooperationsformen möglich ist.

Die Trader-Dienstkomponenten sind so zu entwerfen, daß sie die für die Strategieentscheidungen notwendigen Verwaltungsinformationen (beispielsweise über die aktuelle Dienstgüte) bereitstellen.

6.2 Entwurf des MELODY-Traders

6.2.1 Der MELODY-Trader-Dienst

Der MELODY-Trader-Dienst erlaubt die Vermittlung unterschiedlicher Dienste an anfragende Klienten. Dabei werden sowohl Klientenanforderungen als auch Anforderungen der Systemverwaltung bei der Vermittlung berücksichtigt. Der Trading-Dienst ist in unterschiedliche Trading-Domänen strukturiert, die verschiedene administrative Bereiche repräsentieren (vergleiche Abschnitt 2.3.2). Verschiedene Trading-Domänen kooperieren, um Quantität und Qualität des Trading-Dienstes zu verbessern (siehe Kapitel 5).

Die Dienstleistungen, die vom Trading-Dienst angeboten werden, unterteilen sich in die funktionalen Bereiche

- Angebotsverwaltung (Service Management),
- Anfrage- und Auswahlschnittstelle (Import Interface),
- Trader-Verwaltung (Trader Management) einschließlich Typ- und Kontextmanagement,
- Regelverwaltung (Rule Management) und
- Kooperationsmanagement (Co-operation Management).

Angebotsverwaltung und Importschnittstelle stellen die für den Dienstanbieter (Export) und den Dienstnutzer (Import) notwendigen Operationen zur Verfügung. Die Trader-Verwaltung erbringt Dienste zur Verwaltung des Trading-Dienstes. Sie untergliedert sich weiter in das *Typmanagement*, das *Kontextmanagement* und das *Trader-Dienstmanagement*. Typ- und Kontextmanagement beeinflussen die gleichnamigen Trader-Strukturen. Das Trader-Dienstmanagement erlaubt die Überwachung und die Steuerung des Trader-Dienstes selber. Die Regelverwaltung ermöglicht es, den Prozeß der Dienstauswahl durch die Definition von Regeln zu steuern. Das Kooperationsmanagement verwaltet die Kooperationsbeziehungen zwischen Tradern. Tabelle 8 enthält eine Auflistung der benötigten Operationen.

Funktionsbereich	Operationen
Angebotsverwaltung	insertService, modifyService, deleteService
Auswahlschnittstelle	listService, searchService, selectService
Typverwaltung	insertType, modifyType, listType

Tabelle 8: Operationen der Trader-Funktionsbereiche

Funktionsbereich	Operationen
Regelverwaltung	Die Regelverwaltung ist in die Typverwaltung integriert.
Kontextmanagement	insertContext, deleteContext, listContext
Trader-Dienstmanagement	getHost (Lokation des Trader-Objektes) isAvailable (Verfügbarkeit des Traders) prepareConnection (Vorbereitung der Trader-Verbindung)
Kooperationsverwaltung	initCoop (Initiieren einer Verhandlung) answerCoop (Antwort während einer Verhandlung) receiveCoop (Empfang während Verhandlung) executeCoopAction (Aufforderung zu ausgehandelter Aktion)

Tabelle 8: Operationen der Trader-Funktionsbereiche

Ein Dienstanbieter wird die Methoden *insertService*, *modifyService* und *deleteService* zum Exportieren, Ändern und Zurückziehen eines Dienstes verwenden (Angebotsverwaltung). Ein Dienstnutzer wird die Operationen *listService*, *searchService* und *selectService* nutzen, um sich den Kontextraum des Traders aufzulisten, sich unter Berücksichtigung der eigenen Auswahlkriterien eine Reihe von Diensterbringern auflisten zu lassen oder um sich vom Trader einen Dienst auswählen zu lassen (Auswahlschnittstelle). Ein Systemverwalter kann den Typraum und den Kontextraum mit den entsprechenden Methoden manipulieren und an die eigenen Zwecke anpassen. Zur Überprüfung der Trader-Verfügbarkeit sowie zur Ermittlung der Lokation des Trader-Objektes werden entsprechende Trader-Managementfunktionen angeboten. Die Kooperationsverwaltung dient zur Zusammenarbeit zwischen verschiedenen Tradern. Mit *initCoop* kann eine Kooperationsverhandlung initiiert werden, die im folgenden durch verschiedene Schritte von *answerCoop* und *receiveCoop* ausgehandelt wird. Im Anschluß daran können die in der Verhandlung vereinbarten Aktionen ausgeführt werden.

Die Regelverwaltung verwaltet Auswahlregeln, die bei der Dienstvermittlung berücksichtigt werden. Die Verwaltung dieser Regeln ist in die Typverwaltung integriert, da Auswahlregeln typ-spezifisch interpretiert werden (siehe Abschnitt 6.2.4). Trader-Strategien im ODP Sinn sind momentan durch feste Vorgaben im Trader definiert. Mit jeder Trader-Operation kann jedoch ein Kontrollblock übergeben werden, der einzelne Strategieparameter für die jeweilige Operation ändern kann. Beispielsweise kann die maximale Zeitdauer einer Trader-Operation verändert werden. Ein separates Strategiemanagement ist in Erweiterung der bisherigen Konzepte geplant.

Der Trading-Dienst soll mit einer allgemein verfügbaren Implementierungsplattform realisiert werden. Im ersten Schritt war dies der SUN-RPC, später wurde dann auf den DCE-RPC übergegangen. Mit Hilfe des Konzepts der *Objektadapter* bietet ein Objekt die Operationen des Trader-Dienstes als Methodenaufrufe an, bildet aber intern die Methoden auf die Prozeduren des Kommunikationsvorgangs ab. Dadurch werden die Details der Kommunikation vor den Trader-Objekten verborgen. Diese Separierung ermöglichte beispielsweise einen einfachen Übergang von der früher verwendeten Kommunikationsplattform SUN-RPC zum DCE.

6.2.2 Die Dienstanfragesprache (Service Request Language, SRL)

Die Dienstanfragesprache dient zur Beschreibung von IMPORT-Anfragen. In MELODY besteht die SRL aus einem Textausdruck, der zur Laufzeit interpretiert wird. Dieser Ausdruck enthält zwei Terme. Der *Einschränkungsterm* (Constraint-Term) gibt die Randbedingungen an, die ein Dienst unbedingt erfüllen muß. Er besteht aus einem boolschen Ausdruck über den Attributen eines Dienstes. Der *Optimierungsterm* (Optimization Term) legt das Kriterium fest, nach dem ein Dienst ausgewählt wird. Der Optimierungs-Ausdruck besteht aus einer geschlossenen Formel über den Attributen eines Dienstes. Die genaue Grammatik der Sprache findet sich in [Hamh92].

Das folgende Beispiel für die Auswahlsprache enthält einen Einschränkungsterm und einen Optimierungsterm, die durch das Zeichen ~ getrennt werden:

[(QueueLength<10) && (ResponseTime<10) ~ min (5 * QueueLength + 2 *ResponseTime)]

Der Einschränkungsterm gibt vor, daß nach einem Dienst gesucht wird, dessen aktuelle Warteschlange (QueueLength) kleiner als 10 Einträge und dessen Antwortzeit (ResponseTime) kleiner als 10 Sekunden ist. Der Optimierungsterm gibt an, daß die beiden Variablen im Verhältnis 5:2 gewichtet werden und dabei ein Minimum gesucht wird.

Die Regeln zur Dienstauswahl werden ebenfalls in der SRL spezifiziert. Regeln können entweder innerhalb eines SRL-Ausdrucks referenziert werden oder durch die Angabe ihres Namens direkt verwendet werden (siehe 6.2.4).

6.2.3 Dynamische Attribute im MELODY-Trader

Der *MELODY*- Trader besitzt unterschiedliche Klassen von Attributen. Generell wird zwischen statischen und dynamischen Attribute unterschieden. Statische Attribute werden in zwei Klassen unterteilt. Die einfachste enthält Attribute, deren Wert sich nicht ändert. Die zweite Klasse enthält Attribute, die bei Bedarf mit Hilfe einer Modify-Operation angepaßt werden können.

Zum Zugriff auf dynamische Attribute können verschiedene Zugriffssysteme eingebunden werden. Realisiert ist bislang zum einen der Zugriff mit Hilfe des *MELODY*-Managementsystems (vergl. Kapitel 7), was eine Abbildung dynamischer Attribute auf Werte dieses Managementsystems erfordert. Als Zugriffsfunktionen sind dabei die Operationen (und Optimierungen) des MELODY-Managementsystems vorgesehen. Zum anderen kann die durch das OMG-Modell definierte CORBA-Schnittstelle zum Zugriff verwendet werden, wobei dynamische Attribute auf Objekteigenschaften (Object Properties) des OMG Objektmodells abgebildet werden. Da dieser Objektdienst noch nicht vollständig definiert ist, wird eine vorläufige Version dieser Schnittstellendefinition im MELODY-Trader verwendet (vgl. [Kule95]).

Bei der Auswertung der dynamischen Attribute wurde darauf geachtet, daß keine unnötigen Wartezeiten durch die Abfrage von entfernten Attributen entstehen. Das MELODY Managementsystem bietet dazu die Möglichkeit, asynchrone Anfragen zu stellen oder mit Hilfe von Threads die Evaluierung der Auswahlkriterien für jeden Server getrennt (und damit quasi-parallel) durchzuführen. Auch eine Kombination beider Verfahren ist möglich. Bei der Nutzung des Managementsystems steht ein lokaler Cache zur Verfügung, dessen Invalidierungs- und Update-Strategien durch Aktualitätsprädikate bestimmt werden. Dadurch wurden erhebliche Effizienzsteigerungen bei der Bewertung von Servern erreicht ([KoWi94]).

6.2.4 Verwaltung von Auswahlregeln

In Abhängigkeit von den individuellen Zielen des Trader-Nutzers wird jeweils nur eine eingeschränkte Menge von Diensteigenschaften zur Dienstvermittlung herangezogen. So werden bei der Suche nach einem schnellen Server eher solche Kriterien wie Auslastung, Warteschlangenlänge, Qualität des Gastrechners, u.a. wie im folgenden Beispiel für einen Suchausdruck eine Rolle spielen:

```
FASTEST_SERVER =    ~MIN ( 2,5 * PAGE_RATE/PROCESS_NUMBER
                                + 3 * QUEUE_LENGTH)
```

Dagegen sind z.B. für die Bestimmung eines kostengünstigen Servers eher die Dienst- und Netzkosten maßgeblich. Der jeweilige Suchausdruck ist sowohl sehr spezifisch für das gewünschte Optimierungsziel, als auch sehr abhängig vom betrachteten Diensttyp. Aus mehreren Gründen empfiehlt es sich, den Suchausdruck zu einer *Auswahlregel* zusammenzufassen, mit einem Regelnamen zu versehen und zentral, z.B. im Trader, zu speichern:

1. Alle Trader-Nutzer verwenden für das gleiche Suchziel den gleichen Suchausdruck. Durch geeignete Suchkriterien ist so eine soziale Optimierung möglich. Beispiel: Gleichmäßige Verteilung von Anfragen auf unterschiedliche Server durch das Optimierungskriterium [~min(QueueLength).].
2. Die zentrale Speicherung erlaubt die Pflege des Suchausdrucks durch einen Systemadministrator. Bei Systemänderungen und dadurch bedingten Änderungen des Suchausdrucks müssen die Trader-Nutzer daher nicht unbedingt informiert werden, da automatisch der neue Suchausdruck verwendet wird. Dadurch kann sich der Benutzer auf die Angabe eines abstrakten Optimierungsziels beschränken, welches durch die spezielle Definition der Regel konkretisiert wird. Die Definition kann dann jeweils durch den Systemverwalter angepaßt werden. Wie das obige Beispiel zeigte, können Auswahlkriterien u.U. beliebig komplex sein. Auch dies spricht für die Verwaltung von Auswahlregeln im Trader.
3. Indem der Trader die Auswahl und Bewertung geeigneter Server im Auftrag des Klienten durchführt, beeinflußt er direkt die Abläufe im verteilten System. Durch die Verwendung von Auswahlregeln im Trader kann die Dienstvermittlung dennoch vom Systemmanagement beeinflußt werden. Dies kann für die Aufgaben der Dienstverwaltung (z.B. gleichmäßige Auslastung von Servern) verwendet werden.

Um diese Forderungen erfüllen zu können, wurden Auswahlregeln mit dem jeweiligen Diensttyp assoziiert und mit einem Namen versehen. Mit Hilfe des Namens kann der Dienstnutzer beim Trader-Aufruf eine bestimmte Regel als Auswahlkriterium festlegen. Auswahlregeln entsprechen exakt Selektionskriterien, nur daß sie noch Variablen enthalten können, die vor der Evaluierung des Ausdrucks im Kontext des Dienstnutzers gebunden werden (Beispiel: Rechner, auf dem der Dienstnutzer angesiedelt ist).

Explizite Auswahlregeln

Explizite Auswahlregeln ermöglichen es, vorformulierte Auswahlkriterien (z.B. einen möglichst günstigen Dienst oder einen möglichst schnellen Dienst) in den Trader zu speichern und über einen Namen zu referenzieren. Wenn ein Klient Namen und Bedeutung solcher vordefinierten Auswahlstrategien kennt, kann er die Art der Dienstauswahl schon allein dadurch festlegen, daß er den entsprechenden Namen bei der Import-Operation angibt. Diese

Vorgehensweise hat den Vorteil, daß das Auswahlkriterium nur einmal an neue Situationen angepaßt werden muß und daraufhin von allen Klienten einheitlich benutzt werden kann.

Explizite Auswahlregeln sind spezifisch für einen Diensttyp. Daher werden sie im MELODY-Trader im Typraum verwaltet. Explizite Auswahlregeln werden über einen Namen referenziert. Diese Namen sind im Zusammenhang mit dem zugehörigen Diensttyp eindeutig. Es können aber bei verschiedenen Diensttypen Regeln mit gleichen Regelnamen existieren. Dadurch läßt sich z.B. eine allgemeine Regel FASTEST_SERVER definieren, die dann in Abhängigkeit vom verwendeten Diensttyp interpretiert wird. Die Suche nach einer Regel im Typgraph beginnt am Knoten des in der Anfrage angegebenen Typs. Sie erstreckt sich dann in der Form einer Breitensuche (infolge der Mehrfachvererbung) in Richtung der Wurzel. Diese Suchstrategie erlaubt es, die spezifischste (d.h. die am nächsten zum Ausgangsknoten gelegene) Regel für einen gegebenen Regelnamen zu finden.

Implizite Auswahlregeln

Implizite Auswahlregeln sind ebenfalls im Trader gespeichert, jedoch sind sie für den Nutzer eines Traders nicht sichtbar. Sie werden implizit bei der Dienstauswahl berücksichtigt. Mit Hilfe von impliziten Auswahlregeln kann ein Systemverwalter beispielsweise eine Liste von Dienstinstanzen angeben, die er gezielt von der Vermittlung ausschließen möchte. Implizite Auswahlregeln dienen dazu, ein definiertes Verhalten bei der Dienstauswahl zu erreichen, sind also Teil einer Trader-Strategie (vgl. 6.2.1).

DEFAULT-Regeln

DEFAULT-Regeln sind spezielle implizite Regeln, die dann angewendet werden, wenn vom Anwender kein eigener Auswahlausdruck angegeben worden ist.

6.2.5 Berücksichtigung von Server-Anforderungen an den Klienten

Im Trading-Prozeß müssen nicht nur die Anforderungen des Klienten an den Server berücksichtigt werden, es müssen auch die Anforderungen des Servers an den Klienten überprüft werden. Beispielsweise kann für bestimmte Klienten aus Sicherheitsgründen der Zugriff auf einzelne Server gesperrt sein. Hier macht es Sinn, den entsprechenden Dienst einfach nicht zu vermitteln, da die anschließende Dienstnutzung aus Sicherheitsgründen unterbunden werden würde und der Klient möglicherweise auch gar nicht von der Existenz des Dienstes in Kenntnis gesetzt werden darf.

In diesem Zusammenhang wird jeder Nutzer durch eine Menge an Attributen beschrieben, die beispielsweise seine Identität, die Adresse seines Rechners, die Zugehörigkeit zu bestimmten Systemgruppen, o.ä. beschreiben. Diese Informationen werden teilweise mit der Trader-Anfrage mitgegeben, teilweise mit Hilfe des Kommunikationssystems ermittelt. Jedes Dienstangebot kann eine Auswahlregel enthalten, die für einen gegebenen Dienstnutzer zutreffen muß, damit er den Dienst überhaupt vermittelt bekommen darf.

6.2.6 Garantie von Dienstqualitäten

Bei der Auswahl eines Dienstes spielen seine Eigenschaften eine wichtige Rolle. Für bestimmte Eigenschaften ist es dabei unerläßlich, daß sie nicht nur zum Zeitpunkt der Dienstvermittlung, sondern auch zum Zeitpunkt der Dienstnutzung gültig sind (Garantie der Dienstqualität). Liegen diese Zeitpunkte weit auseinander oder kann sich die betreffende Eigenschaft

rasch ändern, so müssen gesonderte Maßnahmen zur Garantie der Dienstqualität getroffen werden.

Dazu kann der Trader mit dem Diensterbringer in eine Verhandlung eintreten, um sich bestimmte Dienstqualitäten garantieren zu lassen. Zur Garantie der ausgehandelten Qualitäten wird der Diensterbringer Ressourcen-Reservierungen vornehmen. Der Trader übergibt nach erfolgreicher Vermittlung dem Dienstnutzer neben den üblichen Angaben über den Diensterbringer auch eine Identifikation des ausgehandelten Dienstkontrakts, die der Dienstnutzer später dem Diensterbringer präsentiert, um die reservierten Ressourcen nutzen zu können. Aus Sicherheits- und Leistungsgründen dürfen derartige Reservierungen nur für eine bestimmte Zeit gültig sein. Die Zeitdauer für die Reservierung wird durch einen Strategieparameter des Dienstes gesteuert.

6.3 Zusammenfassung

Das vorliegende Kapitel beschreibt den Entwurf des MELODY-Traders, ohne auf Implementierungsdetails einzugehen. Auf der Basis der verschiedenen Aufgabenbereiche eines Traders wurden verschiedene Trader-Operationen identifiziert und ihre Aufgaben beschrieben. Die Beschreibungssprache für Trader-Anfragen (MELODY-SRL) wurde erläutert und verschiedene Klassen von Diensteigenschaften wurden eingeführt. Zur Vereinfachung des Auswahlprozesses gibt es Auswahlregeln, die es erlauben, einen Anfrageausdruck im Trader zu speichern. Als eine weitere Besonderheit des MELODY-Traders kann der Trader Anforderungen des Dienstanbieters an den Dienstnutzer unterstützen. Am Ende wurde die Garantie von Dienstqualitäten behandelt.

7. Das Melody-Managementsystem und sein Einsatz im Trading

Das vorliegende Kapitel beschreibt in knapper Form das Melody-Managementsystem und seinen Einsatz im Trading. Das Managementsystem ist vollständig objektorientiert aufgebaut. Das Informationsmodell (7.1.1) beschreibt die Struktur der verwendeten Objekte, ihre Benennung und ihre Verwaltung in Managementdomänen. Das Systemmodell (7.1.2) beschreibt die Bausteine des Managementsystems, welches sich aus einem allgemeinen Kernsystem (7.1.3) mit für alle Objekte verfügbaren Objektdiensten (7.1.4) und verschiedenen gemeinsam einsetzbaren Managementfunktionen (7.1.5) zusammensetzt. Die eigentlichen Managementanwendungen sind hierarchisch in verschiedenen funktionellen Ebenen gegliedert (7.1.6).

7.1 Übersicht

Der MELODY-Managementdienst realisiert einen konfigurierbaren Systemdienst für das Management verteilter Anwendungen in großen, weitverteilten Umgebungen. Der Managementdienst unterstützt die Strukturierung des Netzes in verschiedene Domänen. Instanzen des Managementdienstes in verschiedenen Domänen kooperieren miteinander, um ein domänenüberschreitendes Management zu ermöglichen.

Infolge der Größe und Verteilung des zu verwaltenden Systems ist das Managementsystem selbst verteilt realisiert. Dies erlaubt es, Leistungskriterien und Aspekte der Fehlertoleranz zu berücksichtigen sowie Managementfunktionen für eine Menge von Dienstnutzern anzubieten. Dynamisch instanziierbare Managementfunktionen ermöglichen es, das Managementsystem an wechselnde Gegebenheiten anzupassen. Durch diese Konfigurierbarkeit des Managementdienstes können mehrere Nutzer Managementaufgaben durchführen lassen. Dabei müssen die verschiedenen Benutzer kooperieren, um unterschiedliche Ziele in Einklang zu bringen.

Die Komplexität der Managementaufgabe sowie ihre Aufteilung in verschiedene funktionelle Bereiche (siehe z.B. [HeAb93]) erfordern geeignete Abstraktionen. Ein Mittel hierzu ist die hierarchische Gliederung der Managementfunktionen. Komponentenorientierte Managementfunktionen überwachen und steuern einzelne Komponenten eines verteilten Dienstes. Anwendungsorientierte Funktionen kontrollieren das Anwendungssystem als Ganzes. Zur Durchführung ihrer Aufgaben können sie die untergeordneten Komponentenfunktionen nutzen. Bild 11 illustriert diesen Ansatz.

Der MELODY-Managementdienst besitzt ein objektorientiertes Informationsmodell (7.1.1), welches in einheitlicher Weise Managementaspekte der verteilten Anwendungen und Managementfunktionen des verteilten Managementsystems beschreibt. Das Systemmodell (7.1.2) des MELODY-Managementdienstes ist an das CORBA-Modell für verteilte objektorientierte Systeme angelehnt. Es besteht aus dem Kernsystem (7.1.3), allgemein vorhandenen Objektdiensten (7.1.4), gemeinsamen Managementfunktionen (7.1.5) und verschiedenen Managementanwendungen (7.1.6).

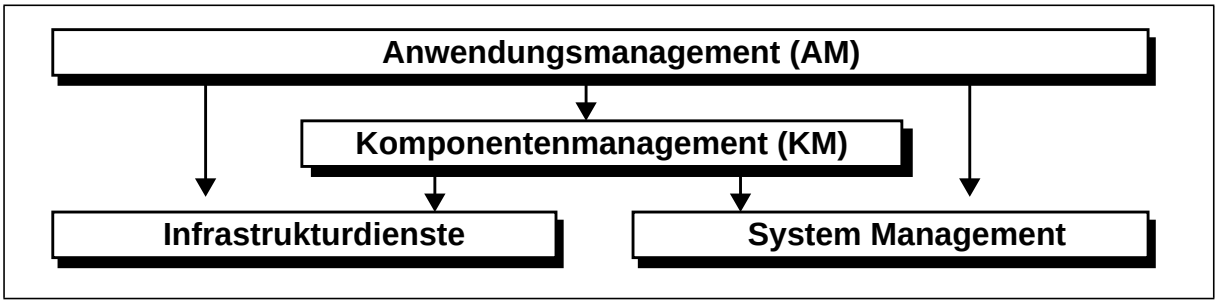


Abbildung 11. Funktionelle Managementhierarchien

7.1.1 Das Informationsmodell des MELODY Managementdienstes

Das Informationsmodell legt die Modellierung der Managementinformation und der Managementanwendungen in einheitlicher Weise fest.

Management Objekte (MOs)
Das *Managementobjekt (MO)* ist die zentrale Komponente des Informationsmodells. Alle für die Verwaltung relevanten Aspekte werden durch MOs repräsentiert. MOs gehören zu *Klassen*, die die vorhandenen Attribute, Ereignisse und Aktionen festlegen. Es gibt eine Vererbungshierarchie zwischen den Klassen, die auch Mehrfachvererbung beinhaltet. Objekte sind selbstbeschreibend, d.h. ein Manager kann die vorhandenen Attribute mitsamt ihrem Typ, die Ereignisse mitsamt der zusätzlich übermittelten Information sowie die Aktionen mitsamt den unbedingt erforderlichen Parametern dynamisch abfragen.

Ressourcen-MOs und Management-MOs
MOs werden für zwei Zwecke herangezogen. *Ressourcen MOs (R-MOs)* repräsentieren die Ressourcen der Anwendungskomponenten. *Management MOs (M-MOs)* enthalten Funktionen, die Aufgaben des Managementsystems durchführen. R-MO sind daher Teil der Anwendung, während M-MO Teil der separaten Managementanwendungen sind. Das einheitliche Informationsmodell bietet den Vorteil, auf beide Arten von MOs in gleicher Weise zugreifen zu können und die gleichen Informationsstrukturen zu benutzen.

Containment-Baum
MOs sind in einem Containment-Baum angeordnet. Dieser Baum verwirklicht eine Enthaltenseinsrelation der MOs, d.h. im Baum tiefer angesiedelte MOs repräsentieren einzelne Teile des übergeordneten MOs. Bei R-MOs repräsentieren untergeordnete MOs Teile der realen Ressource. Dementsprechend stellt die Wurzel des Containmentbaums letztendlich die gesamte Anwendungskomponente dar, während untergeordnete MOs Teile - beispielsweise ein Subsystem - der Anwendungskomponente repräsentieren. Jede Anwendungskomponente enthält genau einen solchen Baum (vergleiche [Kova93b]).

Der Containment-Baum wird auch zur Namensgebung verwendet, indem jedes MO relativ zu seinem Vaterobjekt eindeutig benannt wird. Aus der Konkatenation der relativen Namen ergibt sich dann eine eindeutige Benennung eines MOs im Containmentbaum. Die Baumstruktur kann mit Hilfe entsprechender Managementoperationen abgelaufen werden. Typischerweise wird innerhalb einer Anwendung ein Objektbaum erzeugt und dann dem Managementsystem bekannt gegeben. Alle weiteren Manipulationen am Containmentbaum werden in der Folgezeit dem Managementsystem automatisch mitgeteilt.

Management-Domänen
Management-Domänen unterteilen die Menge der vorhandenen MO-Bäume nach administrativen, operativen und strukturellen Kriterien. Das Wurzelobjekt eines MO-Baums wird in einer

Domäne eingetragen und erhält einen innerhalb dieser Domäne eindeutigen Domänennamen. Über diesen kann das Wurzelobjekt direkt angesprochen werden.

Domänen werden zur Strukturierung wiederum in einer Hierarchie angeordnet. Diese Hierarchie wird ebenfalls zur Namensgebung verwendet. Damit ist eine global eindeutige Referenzierung eines Managementobjektes möglich. Beispielsweise enthält die Domäne

.ipvr.applications.compile_server

alle Server eines Instituts (hier IPVR), die einen Compile-Auftrag annehmen können. Server werden in der Reihenfolge ihres Startens durchnumeriert. So identifiziert der Name

.ipvr.applications.compile_server.comp_server_1

eindeutig das MO-Wurzelobjekt eines Compile-Servers.

Zur Implementierung wurde die Domänenverwaltung auf das DCE CDS abgebildet. Mit jedem Eintrag, der ein Wurzelobjekt eines MO-Baums identifiziert, wird ein CDS-Attribut verwaltet, welches den Namen des Rechners enthält, auf dem die Anwendung zu finden ist.

Management-Views
Jeder Manager definiert eine oder mehrere Views auf die Menge der vorhandenen MOs. Eine View wird aufgebaut, indem die benötigten MOs zur View hinzugefügt werden. Dabei wird eine Schattenkopie des Originalobjektes im lokalen Cache angelegt. Bestimmte, im Zuge von Managementoperationen vorgenommene Zustandsänderungen (beispielsweise das Anlegen von Schattenkopien, das Setzen eines Aktualitätsprädikats oder das Registrieren für Ereignisse) sind mit einer View verknüpft. Mit dem Löschen einer View werden die entsprechenden Zustände im Managementdienst rückgängig gemacht. Managementoperationen werden immer über eine View aufgerufen und liefern ihre Ergebnisse auch an eine View ab. Die in Tabelle 9 beschriebenen Operationen beziehen sich auf eine View.

Operation	Funktion
attach/detach	Fügt ein MO zur aktuellen View hinzu.
inspect	Holt den Wert eines MO Attributes.
manipulate	Setzt den Wert eines MO Attributes.
execute_action	Aktiviert die Methode eines Zielobjektes.
list_<element>	Listet variable Teile eines MOs. Mögliche Teile sind: Attribute, Kind-MOs, definierte Aktionen und Ereignisse.
get_<property>	Holt besondere Eigenschaften eines Objekts. Mögliche Eigenschaften sind: der Typ des MOs oder der Rechnerknoten des MOs.
register	Meldet einen Manager für ein bestimmtes Ereignis an.

Tabelle 9: Managementoperationen

Alle Managementoperationen sind thread-sicher implementiert und können sowohl synchron als auch asynchron aufgerufen werden. Für jede View kann individuell ein Timeout-Wert gesetzt werden, nach dem eine laufende Operation beendet und eine Fehlermeldung zurückgeliefert wird. Ebenfalls mit der View verbunden ist die Registrierung für Ereignisse. In diesem

Zusammenhang können auch Filter für bestimmte Managementereignisse beim Managementsystem registriert werden. Ereignisnachrichten für registrierte Ereignisse werden vom Managementsystem an diejenige View weitergeleitet, die sich für das Ereignis angemeldet hat.

7.1.2 Das Systemmodell

Das MELODY-Managementsystem ist unterteilt in das *Kernsystem*, in *Managementobjektdienste*, in *gemeinsame Managementfunktionen* und in *Managementanwendungen*. Diese Aufteilung ist an das OMG-Modell für verteilte objekt-orientierte Systeme angelehnt. Das Kernsystem bietet grundlegende Möglichkeiten zur Verwaltung und Kommunikation von MOs an. Es hat damit die Funktionen eines *Object Request Brokers*. *Managementobjektdienste* sind allgemeine Dienste, die die Verwaltung von Objekten unterstützen. Die *Namensverwaltung* erlaubt die Einteilung der MOs in verschiedene Domänen und die Benennung der MOs innerhalb ihrer Domäne. Damit ermöglicht er die Lokalisierung eines MOs. Aufbauend auf dem Kernsystem erlaubt das *Aktualitätsmanagement* ([Helb92]) die Realisierung eines intelligenten Caches auf den Knoten des Systems. Mit Hilfe von Aktualitätsprädikaten kann dieser Cache an die Anforderungen des Dienstnutzers angepaßt werden ([Kova93a]). Die Ereignisverwaltung verwaltet die von MOs ausgesendeten Ereignisnachrichten und leitet sie an interessierte Stellen weiter.

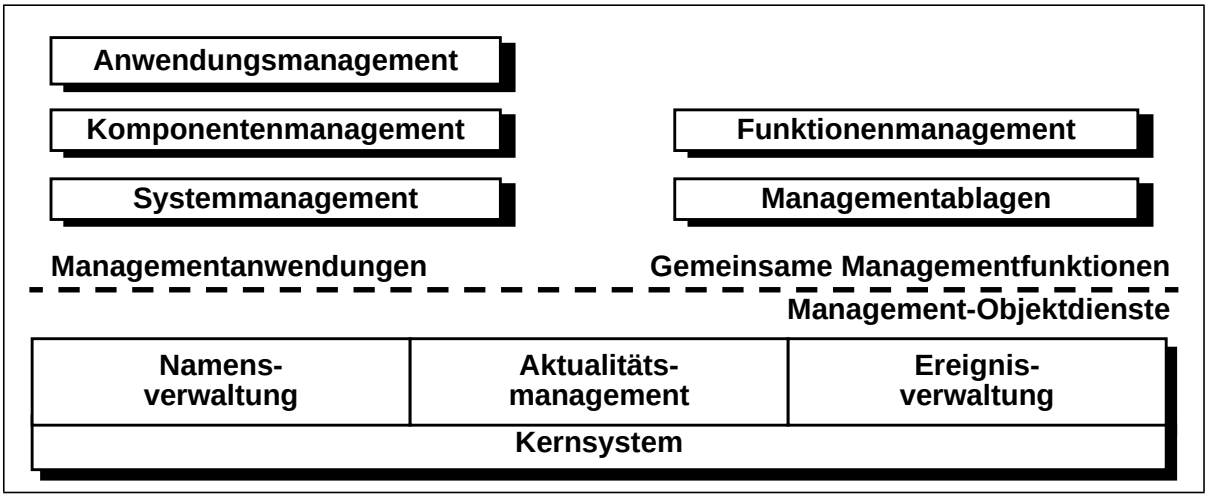


Abbildung 12. Systemmodell des MELODY Managementsystems

Während die eben genannten Managementobjektdienste für jedes MO genutzt werden können und auf jedem Knoten vorhanden sind, stellen *gemeinsame Managementfunktionen* Funktionen bereit, die im Netzwerk verteilt sind und für unterschiedliche Managementanwendungen gemeinsame Aufgaben erfüllen. Im Augenblick werden nur das *Funktionenmanagement* und die *Managementablage* untersucht, denkbar ist jedoch z. B. auch eine gemeinsame Managementfunktion für die Abrechnung von Dienstleistungen.

7.1.3 Das Kernsystem

Der Kern des MELODY Managementsystems stellt das Kommunikationssystem zur Verfügung und erlaubt die dynamische Einbindung von Anwendungskomponenten und Managementprozessen. Der Kern wird durch den MELODY Management Agenten (MMA) gebildet und ist auf jedem Rechner installiert, der mit Hilfe des Managementsystems überwacht werden

soll (siehe Bild 13). Die MMAs kommunizieren über Rechnergrenzen, indem sie mit Hilfe eines einfachen, UDP-basierten Protokolls (dem MELODY Management Protokoll, MMP) Managementrequests austauschen. Das MMP dient nur zur Übermittlung von Requests. Der angebotene Dienst garantiert keine gesicherte Übertragung der Requests. Dies muß mit anderen Mitteln - u.a. auf der Ebene der M-MO - erreicht werden.

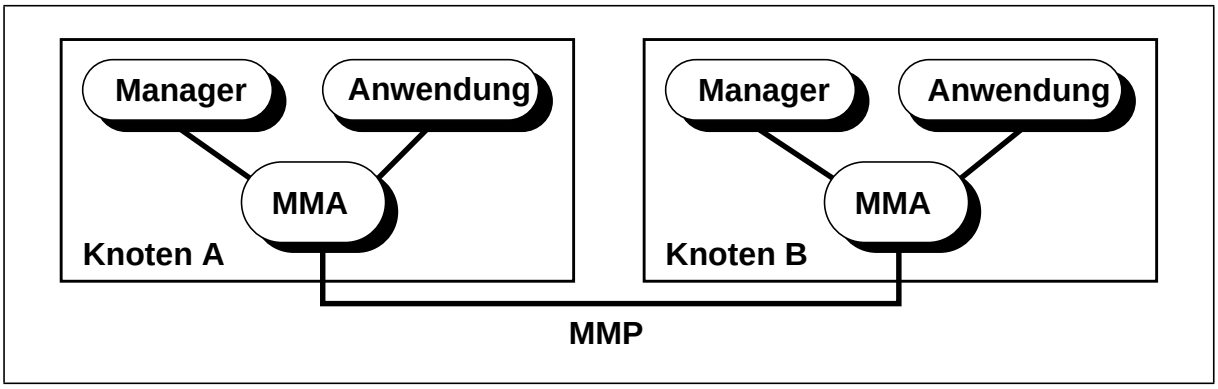


Abbildung 13. Kommunikationsmodell des Managementdienstes

Anwendungskomponenten und Managementkomponenten können beliebig auf Knoten des Netzwerkes ablaufen. Die eingebauten Kommunikationsmechanismen im MMA erlauben die Lokalisation und das entfernte Ansprechen einzelner MOs.

Anbindung an das Managementsystem

Die Anbindung der Anwendungskomponenten und der Managementprozesse an den MMA erfolgt mit Hilfe unterschiedlicher lokaler Kommunikationsmechanismen, die vom Nutzer individuell ausgewählt werden können. Es gibt die Möglichkeit, UNIX-Pipes (CM-PIPE), lokale TCP/IP-Verbindungen (CM-TCP) oder Shared Memory (CM-SHM) zu verwenden. Da zwischen Manager und Anwendung keine direkte Verbindung besteht, meldet das Kernsystem eine Anwendungskomponente und die von ihr verwalteten MOs automatisch beim Managementsystem an. Anschließend werden Änderungen an den Managementdaten bei Bedarf automatisch an das Kernsystem weitergeleitet. Das Kernsystem übernimmt die Speicherverwaltung für die MOs und ermöglicht es M-MOs, auf die Managementinformation der Komponenten zuzugreifen. Neben lesenden Zugriffen sind auch Schreiboperationen möglich. Ferner können Aktionen eines MOs aufgerufen und Ereignisse von einem MO empfangen werden.

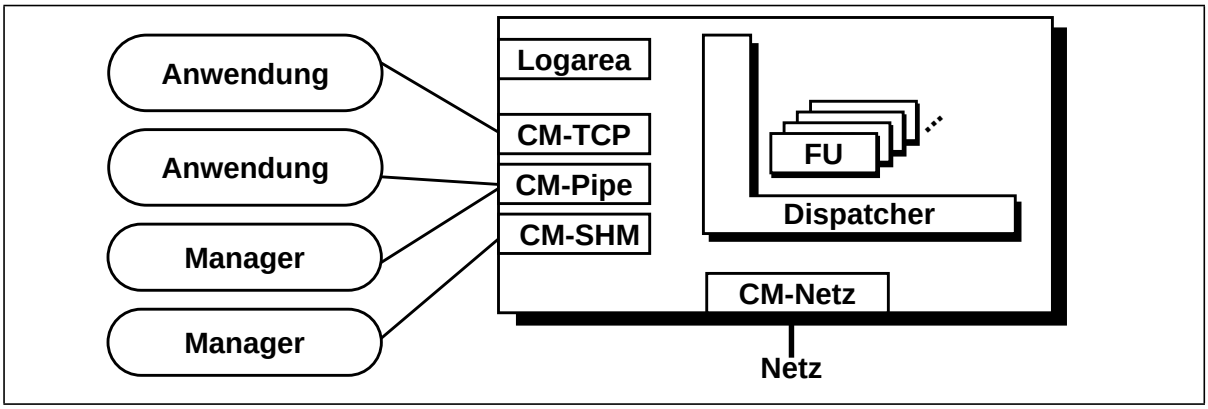


Abbildung 14. Architektur des MELODY Management Agenten

Bild 14 beschreibt die Architektur eines MMA: Die drei lokalen Kommunikationsmechanismen (CM-TCP, CM-Pipe und CM-SHM) regeln die Verbindung zu den lokal laufenden Komponenten oder Managern. Die Logarea verwaltet die Anmeldungen von laufenden Komponenten. Das Objekt CM-Netz überwacht das Netz und empfängt und sendet Management-Anforderungen. Der Dispatcher leitet derartige Nachrichten an die richtigen Funktionseinheiten (FUs, Functions Units) im MMA weiter. Die FUs haben jeweils eine spezielle, eindeutig definierte Funktion. Beispielsweise verwaltet die Ereignis-FU alle Management- und die damit zusammenhängenden Ereignisse sowie die Ereignis-Filter oder die Verteilerlisten von Ereignissen. Andere FUs realisieren den lokalen Cache, ermitteln den Knoten des zu einem Managementrequest gehörenden Zielobjekts, u.a. (vgl. [Jahk93]).

Um die Zeitdauer eines Netzzugriffs zu nutzen, wurde die Kommunikation zwischen Managern, MMAs und Anwendungssystem strikt asynchron ausgerichtet. Auf jeder Seite existiert eine Warteschlange, in die neu eingetroffene Nachrichten eingetragen werden. Die Abarbeitung dieser Warteschlange wird dann von den einzelnen Komponenten gesteuert und kann u.U. zeitversetzt erfolgen.

7.1.4 Management-Objektdienste

Objektdienste stellen allgemeine Funktionen bereit, die von jedem Managementobjekt genutzt werden können. Zu den realisierten Objektdiensten gehören die Namensverwaltung, das Aktualitätsmanagement und die Ereignisverwaltung.

Namensverwaltung
Der Objektdienst Namensverwaltung ermöglicht die global eindeutige Benennung der MOs mit Hilfe des Domänenbaums. Die Namensverwaltung stellt Dienste zur Verfügung, den MO-Namen einzutragen, zu einem gegebenen MO-Namen die Adresse des zugehörigen Rechners zu bestimmen und sich den Baum der vorhandenen MOs aufzulisten. In der aktuellen Implementierung wird der Directory-Dienst des DCE für die Namensverwaltung genutzt.

Aktualitätsmanagement
Das Aktualitätsmanagment stellt einen intelligenten Cache für MOs bzw. für ihre Attribute dar ([Helb92]). Eine Managementanwendung kann gezielt angeben, welche Attribute eines MOs für sie interessant sind und im Cache gehalten werden sollen. Von den zugehörenden Objekten wird dann eine lokale Kopie (ein Schattenobjekt) angelegt. Ferner kann mit Hilfe von Aktualitätsprädikaten ([Kova93a]) eine Anforderung an die benötigte Aktualität der Attribute eines Schattenobjekts gestellt werden. Der Cache sorgt dann dafür, daß eine geeignete Invalidierungs- und Update-Strategie (vgl. Kapitel 4) ausgewählt wird und die vorgegebenen Aktualitätsbedingungen eingehalten werden. Dazu arbeitet er mit dem MMA auf dem Knoten des Originalobjekts zusammen. Durch den regelmäßigen Austausch von Informationen über die Zugriffs- und Änderungsfrequenzen kann sich das Aktualitätsmanagement auf sich ändernde Bedingungen einstellen.

Ereignismanagement
Das Ereignismanagement ermöglicht es Managementanwendungen, sich für Ereignismeldungen auf einem entfernten Knoten zu registrieren Die Registrierung enthält einen Schwellwert für die Priorität eines Ereignisses und einen Ereignisfilter. Mit Hilfe der Priorität läßt sich einstellen, welche Ereignisse weitergegeben werden sollen. Der Ereignisfilter reduziert die Anzahl der weitergeleiteten Ereignisse weiter. Ereignisfilter können MO-Werte und Parameter der Ereignismeldung enthalten.

7.1.5 Gemeinsame Managementfunktionen

Gemeinsame Managementfunktionen sind im Netz installierte M-MOs, die von unterschiedlichen Managementanwendungen genutzt werden. Es gibt das *Funktionenmanagement* und die *Managementablagen*.

Das Funktionenmanagement
Aufgabe des Funktionenmanagement ist es, dynamisch Managementfunktionen im Netz zu installieren, zu überwachen und ggf. zu beenden. Dazu stellen sogenannte *Objektserver* eine Ausführungsumgebung für M-MOs zur Verfügung. Mit Hilfe des Funktionenmanagements können dort dynamisch Managementfunktionen instanziiert werden. Weiterhin können die installierten M-MOs überwacht und verwaltet werden. So kann beispielsweise eine Managementanwendung ein M-MO installieren, welches die Verfügbarkeit einer Komponente eines verteilten Systems überwacht. Das installierte M-MO ruft dazu in periodischen Abständen eine Testfunktion des Servers auf und berechnet daraus die durchschnittliche Erreichbarkeit des Servers.

Managementablagen
Managementablagen dienen der Speicherung des aktuellen Zustands eines MOs. Sie sind im Netz verstreut und stellen Funktionen zur Speicherung eines Objektzustands und zum Zugriff auf die gespeicherten Objekte dar. Dabei können neben der aktuellen Version auch ältere Versionen gespeichert werden, um z.B. eine zeitliche Analyse des Objektverhaltens durchzuführen.

7.1.6 Managementanwendungen

Bei Managementanwendungen wird aus konzeptionellen Gründen zwischen dem Komponentenmanagement und dem Anwendungsmanagement unterschieden (siehe Abschnitt 7.1.2), wobei das Anwendungsmanagement auf dem Komponentenmanagement aufbaut. Sowohl Komponentenmanagement als auch Anwendungsmanagement sind aus M-MOs aufgebaut. M-MOs sind entweder Teil eines Anwendungsprozesses oder werden auf Objektservern installiert.

Das Komponentenmanagement
Die Managementfunktionen des Komponentenmanagement (KM) managen Einzelaspekte einer einzelnen Anwendungskomponente. Beispielsweise kann eine solche Managementfunktion die Auslastung eines Servers über eine bestimmte Zeit abfragen und daraus eine durchschnittliche Auslastung berechnen. Andere Komponentenmanagementfunktionen können die Verfügbarkeit einer Komponente überwachen, sie bei Bedarf herunterfahren oder neu starten.

Das Anwendungsmanagement
Das Anwendungsmanagement wird mit Hilfe der Objektdienste und der gemeinsamen Dienste des MELODY Managementsystems realisiert. Die Funktionen des Anwendungsmanagement (AM) beschäftigen sich mit übergreifenden Aspekten einer verteilten Anwendung. Beispielsweise kann eine solche AM-Funktion die Abrechnungsdaten aller Komponenten sammeln oder für alle Komponenten einen neuen Tarif festlegen und diesen verbreiten. Typischerweise greift das Anwendungsmanagement auf die Funktionen des Komponentenmanagement zurück, um seine Aufgaben zu erfüllen. Das Zusammenspiel ist hierarchisch organisiert, d.h. die Funktionen der höheren Ebene rufen die Funktionen der unteren Ebene auf. Häufig ist jedoch jede Ebene in unterschiedliche Verwaltungsbereiche geteilt, zwischen denen kooperiert werden muß, um die Gesamtfunktion zu erbringen. Desweiteren kann es unterschiedliche

Zuständigkeitsbereiche (beispielsweise getrennt nach System- und Anwendungsmanagement) geben. Dies erfordert dann weitere Absprachen und Kooperationen zwischen unterschiedlichen Teilen des Systems. Bild 15 zeigt das Zusammenspiel zwischen dem Anwendungsmanagement und dem Funktionenmanagement.

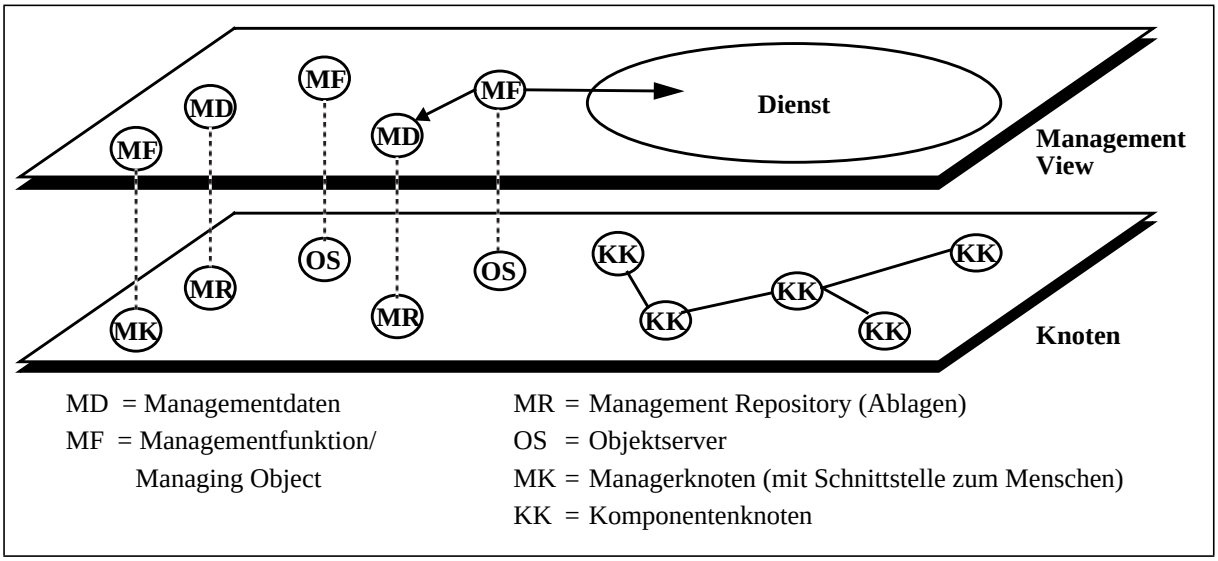


Abbildung 15. Architektur des Anwendungsmanagements

7.2 Integration des Trading-Dienstes und des Managementdienstes

Trading-Dienst und Managementdienst arbeiten nach dem Konzept der gleichberechtigten Integration (vgl. Abschnitt 3.2) zusammen, um gemeinsam den Marktinfrasturkturdienst zu erbringen. Im folgenden werden die einzelnen Zusammenhänge nochmals aufgezählt und erläutert.

Dynamische Attribute
Dynamische Attribute eines Dienstes können über die Attribute eines MOs definiert werden. Dem Trading-System wird diese Abbildung beim Export des Dienstangebots mitgeteilt. Dieses nutzt dann im Verlauf der Dienstvermittlung die Möglichkeiten des Managementsystems, um auf den aktuellen Wert des Attributs zuzugreifen. Dabei kann die Möglichkeit des intelligenten Caches und der Aktualitätsprädikate ausgenutzt werden, um die Anfragehäufigkeit zu reduzieren. Der Dienstanbieter gibt dabei eine erste sinnvolle Aktualitätsanforderung an, Systemverwaltung oder Nutzer können die benötigte Aktualität dynamisch anpassen. Damit werden die in 6.1.2 geforderten Unschärfebedingungen realisiert.

Das Managementsystem bietet eine synchrone und eine asynchrone Arbeitsweise an und ist thread-sicher implementiert. Durch die Nutzung der asynchronen Anfragemöglichkeit sowie durch die Möglichkeit, verschiedene Threads einzusetzen, unterstützt es das Trading-System beim parallelen Zugriff auf im System verstreute Informationen.

Dynamisch installierte Managementfunktionen
Mit Hilfe von M-MOs können bestimmte Aspekte von Diensten überwacht und berücksichtigt werden. M-MOs können entweder vom Dienstverwalter oder vom Trading-Dienst angelegt werden. Damit ist es dem Trading-Dienst möglich, den zu vermittelnden Dienst zu überwachen und auch indirekt über den Dienst gewonnene Erkenntnisse bei der Dienstauswahl zu berücksichtigen. Beim Export eines Dienstes wird angegeben, welche Attribute eines M-MOs

für den Dienst relevant sind. Der Trader überprüft beim Einfügen des Dienstes in die eigene Datenbank die Existenz des M-MOs und installiert gegebenenfalls benötigte M-MOs im Netz. Im Unterschied zu dynamischen Attributen, bei denen der Zugriff direkt vom Dienst zur Verfügung gestellt wird, kann mit Hilfe der Managementfunktionen weitere Information über den Dienst gewonnen und bei der Vermittlung berücksichtigt werden.

Eigenschaften der Kommunikationsverbindung
Bei der Dienstvermittlung kann die Qualität der Kommunikationsverbindung zwischen dem Nutzer und dem Diensterbringer von Bedeutung sein. Dazu können dynamisch erzeugte M-MOs verwendet werden, die die Qualität (z.B. Fehlerrate, Durchsatz, reservierte Bandbreite, o.ä.) zur Laufzeit ermitteln und an den Trader melden.

Auswahlregeln
Die im Trader verwendeten Auswahlregeln bestimmen die bei der Dienstauswahl zu verfolgende Strategie. Daraus folgt, daß die Systemverwaltung zur Umsetzung ihrer Managementziele die Auswahlregeln geeignet modifizieren und dementsprechend das Verhalten des Traders steuern kann. Um beispielsweise in naher Zukunft einen Server herunterfahren zu können, kann ein Systemverwalter beschließen, diesen ab sofort nicht mehr vermitteln zu lassen. Mit Hilfe einer impliziten Auswahlregel kann daraufhin bei der Dienstvermittlung bewirkt werden, daß dieser Server in Zukunft nicht mehr vermittelt wird.

Zustandsüberwachung
Das Managementsystem überwacht den Zustand der angebotenen Dienste und meldet aufgetretene Zustandsänderungen mit Hilfe von Ereignismeldungen. Das Trading-System ermittelt ausgefallene Dienste und schließt diese von der Vermittlung aus. In Zusammenarbeit mit dem Managementsystem wird die erneute Verfügbarkeit eines Dienstes überwacht und ggf. dieser wieder bei der Vermittlung berücksichtigt.

7.3 Zusammenfassung

Das MELODY Managementsystem ist ein verteilter Dienst, der die Verwaltung eines (ebenfalls) verteilten Dienstes ermöglicht. Das Informationsmodell beruht auf einer objektorientierten Modellierung der Managementressourcen und der Managementfunktionen. Das Systemmodell ist an das CORBA-Modell für verteilte objektorientierte Systeme angelehnt. Es bietet allgemein verfügbare Managementdienste zur Benennung von Objekten, zur Verwaltung der Aktualität und zur Behandlung von Ereignismeldungen. Gemeinsame Managementfunktionen stellen eine Ausführungsumgebung für dynamisch instanziierbare Managementfunktionen und für Managementablagen bereit. Die eigentlichen Managementaufgaben werden vom Komponenten- und Anwendungsmanagement durchgeführt. Das Managementsystem ist flexibel und dynamisch um neue Funktionen erweiterbar.

Mit Hilfe dieser umfangreichen Möglichkeiten ist ein umfassendes Management einer verteilten Anwendung möglich. Dabei ist es jedoch unerläßlich, daß die Managementfunktionalität in die Komponenten der Anwendung integriert wird. Ein derartiger Prozeß wird als Instrumentieren der Anwendung bezeichnet. Die Instrumentierung sowie die vorgesehenen Managementfunktionen sollten bereits bei der Entwicklung eines Dienstes mit eingeplant werden. Es stehen vordefinierte Managementanwendungen zur Verfügung, die jedoch für spezielle Aspekte eines verteilten Dienstes an die entsprechende Aufgabenstellung angepaßt werden müssen.

8. Implementierung

In diesem Kapitel wird die aktuelle Implementierung des MELODY-Systems beschrieben. Am Anfang wird ein Gesamtüberblick über die Architektur des Systems gegeben. Im Anschluß daran werden die Implementierungen des Trader-Dienstes und des Managementdienstes im Detail besprochen.

8.1 Architektur des Trading-Dienstes

Der Trading-Dienst ist nach dem Klient-Server-Modell entworfen. Dabei erbringt der Serverprozeß die Funktionalität des Traders. Er wird Trader Service Agent (TSA) genannt. Nutzer des Trader-Dienstes wenden sich über einen Benutzeragenten (Trader User Agent - TUA) an den Trader. TUA und TSA sind als C++ - Objekte implementiert. Die Kommunikation zwischen ihnen erfolgt mit Hilfe von Objektadaptern auf der Basis des Distributed Computing Environment (DCE, [Schi93]). Methodenaufrufe des TUA-Objektes werden auf den TUA-Objektadapter abgebildet. Dieser ist für die Lokalisation des TSA mit Hilfe des DCE-Namensdienstes und für die Abbildung der Trader-Dienstnutzungen auf die DCE-RPCs verantwortlich. Auf der TSA-Seite bildet der Objektadapter die DCE-RPC Aufrufe direkt auf Methodenaufrufe des TSA-Objektes ab. Durch diese Separierung der Funktionalität von der verwendeten Kommunikationsschnittstelle (in der Form des Objektadapters) ist ein leichter Übergang auf einen anderen Kommunikationsmechanismus möglich. Dies wurde beim Übergang von einem SUN-RPC-basierten Kommunikationsmechanismus zu einem DCE-basierten Mechanismus demonstriert. Bild 16 erläutert diese Architektur.

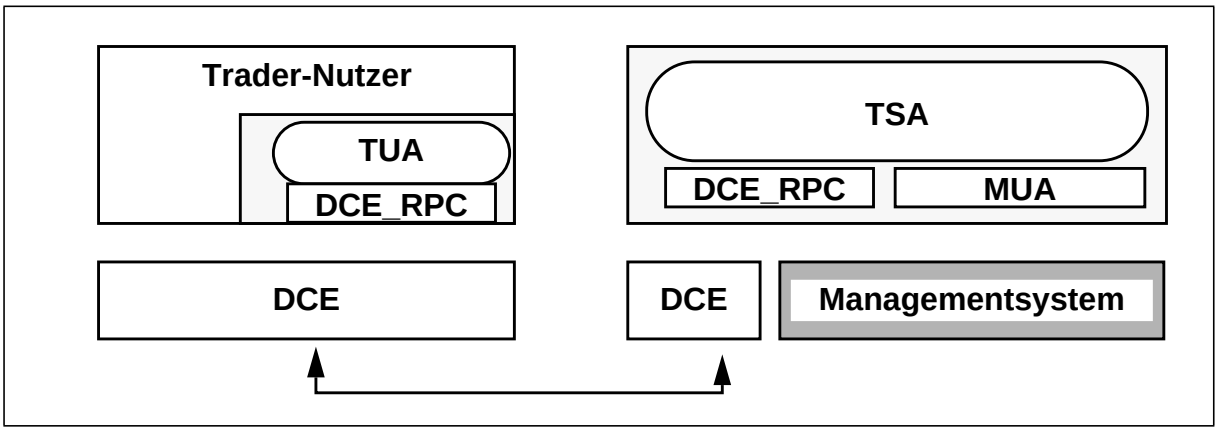


Abbildung 16. Systemarchitektur des Traders

Das TSA-Objekt implementiert alle für das Trading notwendigen Module (z.B. Kontextverwaltung und Typmanagement). Die Zusammenarbeit mit dem Managementdienst erfolgt mit Hilfe des *Management User Agent (MUA)*, mit dem die Dienste des Managementsystems angefordert werden.

8.1.1 Das TUA-Objekt

Das TUA-Objekt sorgt für den Zugang der Trader-Nutzer zur Trading-Dienstleistung. Es ermittelt anhand einer Umgebungsvariable TRADER_DOM die aktuelle Trader-Domäne. Mit Hilfe der Namensverwaltung des DCE kann der TUA den zu einer Trader-Domäne gehörenden TSA lokalisieren. Nach der Instanziierung des TUA-Objektes und dem Aufruf der vorbereitenden Methode *prepareConnection* kann der Trader der aktuellen Trader-Domäne mit Hilfe der Objektmethoden des TUA angesprochen werden.

Für den Fall, daß mit anderen Trader-Domänen kommuniziert werden soll, kann bei der Instanziierung eines TUA-Objektes der Name einer anderen Trader-Domäne angegeben werden. Innerhalb eines Prozesses sind mehrere unterschiedliche TUAs möglich, um mit verschiedenen Trader-Domänen zu kommunizieren. Auf diese Weise wird z.B. auch die Kooperation zwischen verschiedenen Tradern realisiert.

Das TUA-Objekt stellt die Trader-Operationen aller funktionalen Bereiche - z.B. für die Angebotsverwaltung oder für das Importmanagement - zur Verfügung (siehe Abschnitt 6.2.1). Seiner Rolle entsprechend wird ein Dienstanbieter die Methoden *insertService*, *modifyService* und *deleteService* zum Exportieren, Ändern und Zurückziehen eines Dienstes verwenden (Angebotsverwaltung). Ein Dienstnutzer wird die Operationen *listService*, *searchService* und *select-Service* nutzen, um den Kontextraum des Traders zu durchsuchen, sich nach gewissen Auswahlkriterien eine Reihe von Diensterbringern aufzulisten und um sich vom Trader einen Dienst auswählen zu lassen (Importverwaltung).

Jede Operation bekommt ein Objekt übergeben, welches die Parameter der Operation enthält. Als Rückgabe werden ein Fehlercode, ein Antwortobjekt und im Fehlerfall eine Fehlermeldung zurückgeliefert. Jedes Parameterobjekt des Traders enthält einen Kontrollblock, mit dem bestimmte Strategieparameter einer Trader-Operation, beispielsweise die maximale zeitliche Dauer oder der Umfang der Antwort definiert werden.

8.1.2 Das TSA-Objekt

Das TSA-Objekt implementiert die Trader-Funktionalität. An seiner Schnittstelle stellt es alle Operationen für die unterschiedlichen Funktionsbereiche bereit. Das TSA-Objekt enthält die für die Trader-Operationen benötigten Funktionsmodule, d.h. verschiedene Teilobjekte (z.B. für die Zugriffskontrolle). Einzelne Trader-Operationen werden unter Kontrolle des TSA-Objektes im Zusammenspiel mit den TSA-Teilobjekten erbracht.

Es gibt je ein Objekt für den Typraum (*typeSpace)* und den Kontextraum (*contextSpace).* Das *Operationsmodul* führt die Trader-Operationen durch, während das *Managementmodul* für Trader-interne Vorgänge - beispielsweise die Thread-Verwaltung - zuständig ist. Das Reservierungsmodul führt Garantieverhandlungen mit dem Diensterbringer durch und kann dabei Ressourcen belegen. Das *Kooperationsmodul* ist für die Etablierung von Kooperationen und in diesem Zusammenhang auch für die Überwachung des Trading-Dienstes insgesamt zuständig. Bild 17 zeigt die Architektur des TSA-Objekts.

Managementmodul und Operationsmodul
Das Managementmodul ermöglicht die intern im Trader erfolgenden Abläufe. Es bereitet die Trader-Operationen vor, erzeugt die notwendigen Threads und führt weitere verwaltende Tätigkeiten durch. Das Operationsmodul verwirklicht die Trader-Operationen. Dazu greift es auf den Kontextraum und den Typraum zu.

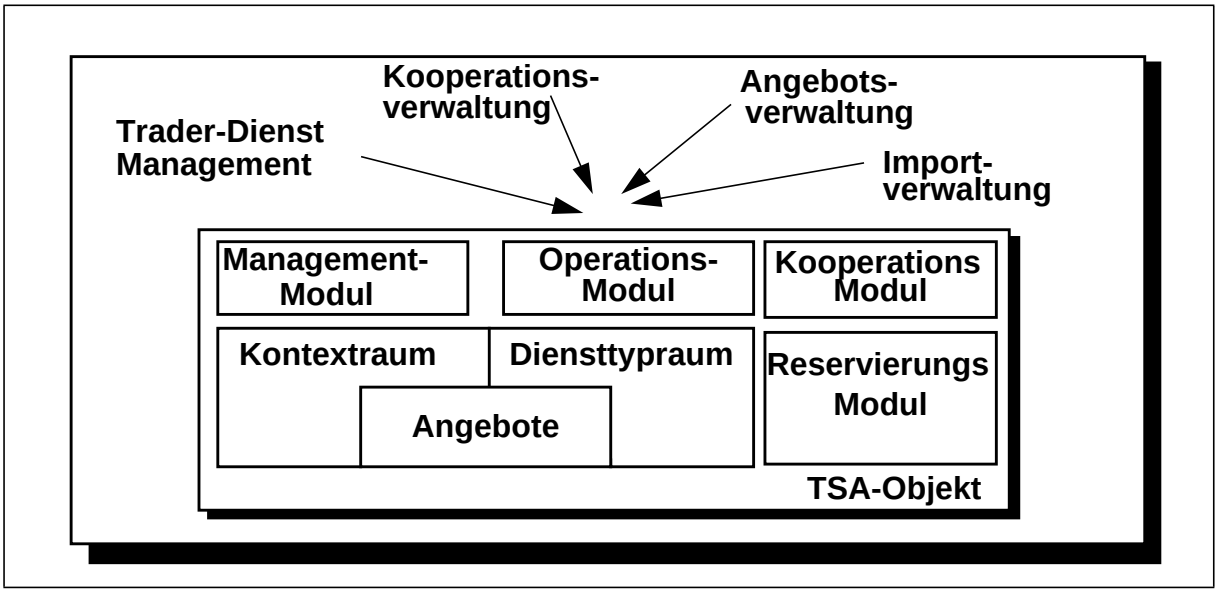


Abbildung 17. Struktur des TSA-Objektes

Der Kontextraum
Der Kontextraum strukturiert die Dienstangebote. Er kann von den Anwendern und vom Systemverwalter beliebig in der Form eines hierarchischen Baums strukturiert werden. In einem Kontext werden die dazu gehörenden und dementsprechend darin gespeicherten Dienste verwaltet. Zum schnellen Auffinden eines Kontexts wird der Kontextraum durch eine Hashtabelle über die (vollen) Namen der Kontexte indiziert ([Hamh92]).

Typraum
Der Typraum verwaltet die Diensttypen und die zwischen den Diensttypen bestehenden Subtyp-Relationen. Der Typraum ist als azyklischer Graph realisiert. Jeder Typ besitzt einen eindeutigen Namen. Zur schnellen Suche im Typraum werden die Typen in einer Hashtabelle verwaltet. Mit den Diensttypen integriert ist die Verwaltung der Auswahlregeln, die im Trader gespeichert werden können (siehe Abschnitt 8.1.5). Weiterhin enthält ein Diensttyp eigene Diensttypattribute. Diese beschreiben beispielsweise, welche Attribute in einer Dienstbeschreibung dieses Typs zwingend erforderlich oder optional sind.

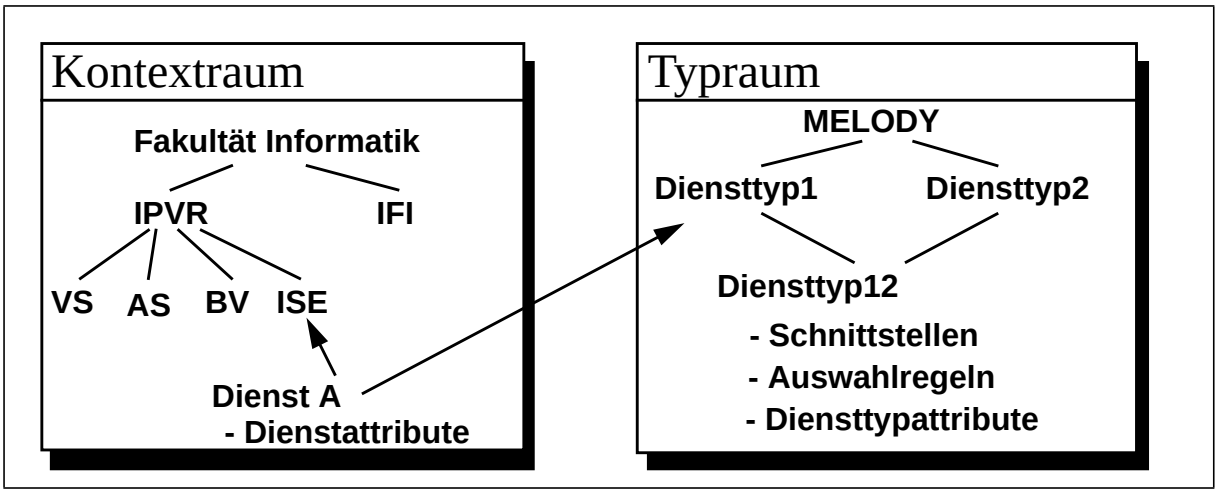


Abbildung 18. Informationsstrukturen im Trader

Angebote
Angebote werden in den jeweiligen Kontexten verwaltet. Sie bestehen neben dem Angebotsnamen aus einer Liste von Attributobjekten unterschiedlichsten Typs, auf die mit einer einheitlichen Schnittstelle zugegriffen wird. In Abhängigkeit vom Typ des Attributes wird der Wert des Attributes vom Managementdienst abgefragt oder aus der eigenen Datenbank ausgelesen.

8.1.3 Die Abarbeitung einer Import-Anfrage

Eine Import-Anfrage wird vom Operationsmodul ausgeführt. Dieses bestimmt die für den gesuchten Diensttyp zutreffenden Angebote und wertet bei einem SELECT im Anschluß daran für jedes Angebot den SRL-Ausdruck aus. Beim Zugriff auf dynamische Attribute werden verschiedene Optimierungen durchgeführt, um die Zugriffszeiten zu reduzieren (siehe [KoWi94]). Dazu wird wie folgt vorgegangen:

Gestaffelte Auswertung
Beim Prinzip der gestaffelten Auswertung wird der SRL-Ausdruck einer Anfrage in einen Operatorbaum überführt. In diesem Baum werden Äste, die keine dynamischen Attribute besitzen, zuerst ausgewertet. Im Anschluß daran erfolgt die Evaluierung dynamischer Attribute.

Asynchrone Anfragen
Anfragen an Attribute können asynchron abgesetzt und zu einem späteren Zeitpunkt aufgesammelt werden. Beim Aufsammeln wird der Operatorbaum unmittelbar weiter ausgewertet. Durch das Ausnutzen der verkürzten Auswertung boolscher Operatoren können Teile des Operatorbaumes direkt eliminiert werden, was die Auswertung weiterhin beschleunigt.

Parallele Anfragen
Verschiedene Dienstangebote können parallel in verschiedenen Threads evaluiert werden. Das Managementmodul stellt dafür eine Anzahl Threads zur Verfügung. Nach der Bewertung der Angebote wird die Rangfolge der Dienste bestimmt und gegebenenfalls ein bester Dienst ausgewählt.

Intelligenter Cache
Der Managementdienst erlaubt die Definition von Unschärfebedingungen an die Aktualität eines Attributwertes. Hierfür stellt er einen intelligenten Cache zur Verfügung, der vom Trading-Dienst durch die Angabe von Aktualitätsprädikaten an die Aktualitätsanforderungen der Dienstnutzer und -erbringer angepaßt wird. Die oberen drei Optimierungen beschleunigen die Auswertung des Traders, ohne die Qualität der Trading-Entscheidung zu vermindern. Durch den eingesetzten Cache kann nun u.U. eine veraltete Sicht auf das System entstehen, wodurch die Qualität des Trading-Dienstes eingeschränkt wird. Diese Einschränkung bleibt jedoch im Rahmen der vorgegebenen Aktualitätsbedingungen. Die im Cache verwendeten Invalidierungs- und Update-Strategien vermeiden durch dynamische Anpassung hohe Netzlasten und verringern die Antwortzeiten des Trading-Dienstes (siehe [KoWi94]).

8.1.4 Implementierung dynamischer Attribute

Der MELODY Trader-Dienst unterstützt zwei Arten von dynamischen Attributen (vgl. 6.2.3). Die erste Klasse umfaßt Attribute, die aus dem MELODY Managementdienst gewonnen werden. Die zweite Klasse beinhaltet Attribute, auf die mit Hilfe von *Objekteigenschaften (Object Properties [OMG95])* des CORBA-Objektmodels zugegriffen wird. Um auf beide Attributarten und auf statische Attribute einheitlich zugreifen zu können, wurde eine abstrakte Zugriffs-

schnittstelle für Dienstattribute definiert. Dies geschieht durch die Verwendung einer gemeinsamen virtuellen Basisklasse in C++. Für jede Attributklasse wurde diese Basisklasse derart spezialisiert, daß Zugriffe des Traders auf Zugriffe auf die Informationsquelle (d.h. Managementsystem oder CORBA Objektdienst) abgebildet werden. Jede abgeleitete Klasse verwaltet intern Information, die den Zugriff auf die Informationsquelle erlaubt. Der allgemeine Aufbau der Attributklassen wird in Abbildung 19 illustriert. Die verwendete Zugriffsschnittstelle wird in Tabelle 10 erläutert.

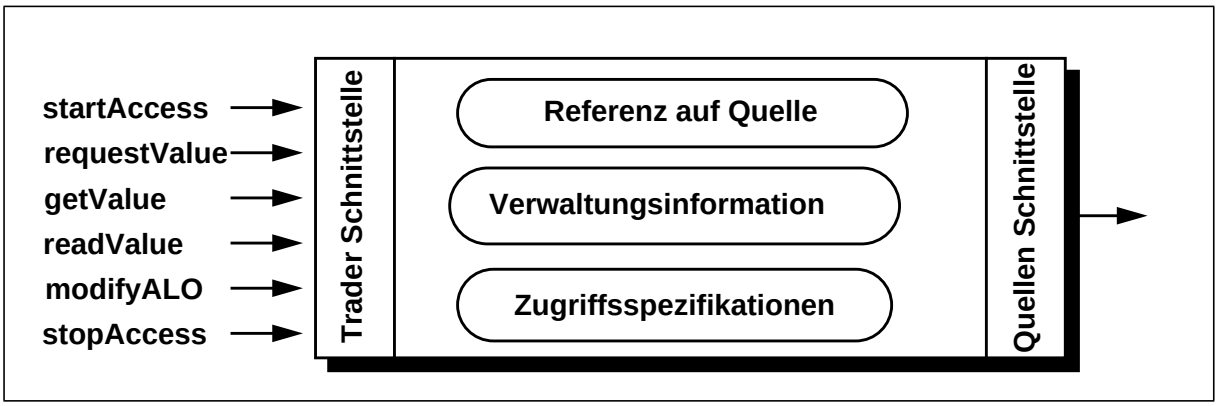


Abbildung 19. Zugriffsschnittstelle auf Dienstattribute

Ein Attributobjekt wird allgemein auch als *Access Location Object (ALO)* bezeichnet. Es verwaltet eine Referenz auf die Quelle des Attributwertes, Verwaltungsinformation für den Zugriff und - falls benötigt - zusätzliche Spezifikationen für den Zugriff. Im einfachsten Fall enthält die Referenz auf die Quelle alle benötigte Information, so daß weder Verwaltungsinformationen noch Zugriffsspezifikationen benötigt werden. Im Fall von Managementattributen enthält die Verwaltungsinformation beispielsweise Angaben über die definierten Aktualitätsprädikate.

Eine Zugriffsspezifikation ist ein Ausdruck, der dynamisch zum Zugriffszeitpunkt evaluiert wird, um ein geeignetes Attribut zu bestimmen. Dabei werden insbesondere freie Variablen des Ausdrucks im Rahmen des aktuellen Suchkontexts gebunden. Um beispielsweise die Netzbandbreite zwischen dem Dienstnutzer und dem Diensterbringer ermitteln zu können, wird für beide Beteiligten der Name des zugehörigen Rechners benötigt. Die Bandbreite könnte durch den folgenden Ausdruck definiert sein:

.connection.$SERVICE_NODE.$USER_NODE.bandwidth

Dieser Ausdruck enthält zwei Variablen, die den Knoten des Diensterbringers und des Dienstbenutzers spezifizieren. Diese Information kann aus den Eigenschaften des Dienstangebotes bzw. des Dienstnutzers ermittelt werden, ist jedoch erst zum Zeitpunkt der Auswertung der Zugriffsspezifikation vollständig bekannt. Beide Variablen werden also erst zum Zeitpunkt des Attributszugriffs aus dem gültigen Auswertungskontext gebunden und spezifizieren dann eine gültige Variable des Managementsystems.

Tabelle 10 erläutert die Aufgaben der einzelnen Funktionen der Zugriffsschnittstelle. Die jeweiligen Attributklassen sind nun so spezialisiert, daß die Zugriffsfunktionen des ALO-Objektes auf die entsprechenden Operationen des verwendeten Kommunikationssystems abgebildet werden. Der Auswertungsalgorithmus verwendet in Abhängigkeit von der spezifizierten Auswertungsstrategie eine synchrone oder asynchrone Zugriffsweise. Ebenfalls durch die spe-

zifizierte Auswertungsstrategie wird bestimmt, ob unterschiedliche Angebote sequentiell in einem einzigen Thread oder parallel in jeweils eigenen Threads ausgewertet werden.

Funktion	Erläuterung
startAccess	Dienst zum Anlegen des Attributobjektes. Für einige Attributtypen werden dadurch Initialisierungsoperationen aufgerufen, beispielsweise das Anmelden beim Managementsystem oder die Installation einer notwendigen Managementfunktion im Netzwerk.
stop Access	Wird beim Löschen des Attributobjektes aufgerufen und macht z.B. die bei der Anmeldung durchgeführten Operationen rückgängig.
requestValue	Synchroner Aufruf, um den Wert eines Attributs zu lesen.
getValue	Asynchroner Aufruf, um den Wert eines Attributs zu lesen.
readValue	Liefert den gelesenen Wert oder gibt Auskunft über den Zustand der asynchronen Operation.
modifyALO	Verändert die Verwaltungsinformation, z.B. die Referenz auf die Quellen, die spezifizierten Verwaltungs- oder Zugriffsinformationen.

Tabelle 10: Funktionen der Zugriffsschnittstelle

8.1.5 Das Regelmanagement

Mit den Diensttypen können typspezifische Auswahlregeln gespeichert werden. Auswahlregeln bestehen aus einem Namen (Label) und dem Regelinhalt, einem SRL-Ausdruck. Mit Hilfe des Namens können Regeln im SRL-Ausdruck einer Anfrage referenziert werden. Der endgültige SRL-Ausdruck entsteht dann durch die Kombination der Regel mit dem angegebenen SRL-Ausdruck. Abbildung 20 illustriert diesen Ansatz. Ein Klient kann bei der Auswahl

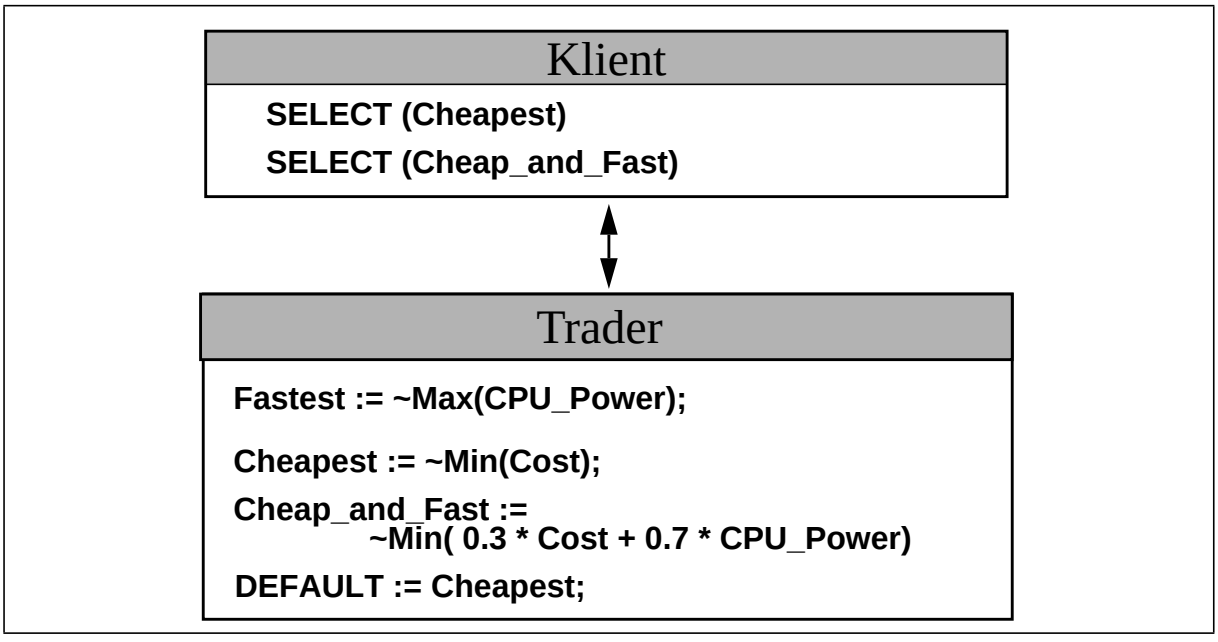


Abbildung 20. Verwaltung von Auswahlregeln im Trader

eines Dienstes beispielsweise den billigsten (Cheapest) oder eine Kombination aus billig und schnell (Cheap_and_Fast) anfordern. Der Trader hat unter den entsprechenden Namen Auswahlregeln gespeichert, die er bei der Suche nach geeigneten Dienstangeboten ausnutzt.

Es gibt drei Arten von Regeln:

- DEFAULT-Regel

  Die DEFAULT-Regel gibt das Auswahlkriterium an, nach dem Dienste bei einer Dienstauswahl bewertet werden, wenn durch den Trader-Nutzer keine Auswahlregel vorgegeben worden ist. DEFAULT-Regeln werden also verwendet, wenn ein Dienst ausgewählt werden soll, jedoch kein SRL-Ausdruck angegeben worden ist.

- Explizite Regeln

  Explizite Regeln können mit ihrem Namen als Teil einer Trader-Anfrage referenziert werden. Der Name wird dann im SRL-Ausdruck syntaktisch durch den Regelinhalt ersetzt. Nach der vollständigen Ersetzung aller expliziten Regeln kann der SRL-Ausdruck ausgewertet werden. Es ist darauf zu achten, daß keine unzulässigen Ausdrücke entstehen (vgl. [Wira94]).

- Implizite Regeln

  Implizite Regeln werden vom Trader automatisch zur Bewertung von Dienstangeboten hinzugenommen. Dieser Mechanismus erlaubt es, die vom Klienten angegebenen Auswahlbedingungen gemäß vorgegebener Strategien anzupassen.

Regeln ermöglichen es dem Systemadministrator, die genauen Kriterien für eine Dienstauswahl festzulegen. Klienten können mit Hilfe des Regelnamens angeben, welche Art von Optimierung gefordert wird, ohne das genaue Kriterium zur Optimierung angeben zu müssen. Dies erlaubt es, bei der Dienstauswahl Strategien zu berücksichtigen, ohne daß eine Vielzahl von Klienten über diese Strategien detailliert Bescheid wissen müssen.

8.1.6 Garantie von Dienstgüte

Der MELODY-Trading-Dienst ermöglicht es, für bestimmte Dienstparameter eine Garantie auszuhandeln. Diese Bestimmung der Garantie ist durch eine *Verhandlungsstrategie* und durch eine *Ressourcenbelegungsstrategie* festgelegt. Im Augenblick ist jeweils nur eine sehr einfache Strategie definiert.

Für die Verhandlung nutzt der Trader die Funktionen des Managementdienstes, um mit dem Dienst die Ressourcenbelegung auszuhandeln. Der Trader ruft eine Managementfunktion auf, der er ein Jobprofil übergibt. Dieses Jobprofil enthält Angaben über die Eigenschaften, die es zu garantieren gilt - beispielsweise die Position in einer Warteschlange. Der Dienst versucht nun, diese Anforderungen durch interne Reservierungen zu garantieren und übermittelt eine Erfolgs- oder Ablehnungsmeldung. Bei dieser Strategie werden keine Verhandlungen über mögliche Alternativen vorgenommen. Mit der Erfolgsmeldung ist ein *Reservierungsidentifier* verbunden, den der Dienstnutzer während seiner Dienstaufrufe verwenden kann, um auf die reservierten Ressourcen Bezug zu nehmen. Die Ressourcenbelegungsstrategie definiert das Verhalten eines Dienstanbieters für belegte Ressourcen. Die augenblickliche Ressourcenbelegungsstrategie reserviert Ressourcen nur für eine vorgegebene Zeitspanne. Werden sie innerhalb dieser Zeitspanne nicht durch den Nutzer angefordert, so kann der Diensterbringer sie wieder freigegeben. In der augenblicklichen Implementierung sind diese Strategien sowie die eingestellte Reservierungsdauer fix für alle Dienstnutzer.

8.1.7 Das Kooperationsmodul

Das Kooperationsmodul etabliert, überwacht und beendet die Kooperation zwischen verschiedenen Tradern. Falls gemäß der vorgegebenen Kooperationsstrategie ein Kooperationsbedarf entdeckt wird, der nicht anderen Trader-Zielen widerspricht, so wird mit anderen Tradern eine Kooperation ausgehandelt und aufgebaut. Die Liste der Trader aus anderen Domänen wird im eigenen Kontextbaum in Form von Dienstangeboten verwaltet. Dies hat den zusätzlichen Vorteil, vor dem Aufbau von Kooperationen bereits eine Vorselektion durchführen zu können, um nur mit geeigneten, d.h. kooperationswilligen Tradern mit den gesuchten Eigenschaften zu verhandeln. Ein Beispiel für eine Kooperationsstrategie wurde in Kapitel 5 und [Burg95] erläutert, wo die Dienstqualität des Trading-Dienstes überwacht und daraus Kooperationsbedingungen abgeleitet wurden.

8.1.8 Ausblick

In der augenblicklichen Implementierung ist jeder Trader einer Trading-Domäne ein einzelner Dienstprozeß. Zur Steigerung der Fehlertoleranz und zur effizienteren Bearbeitung paralleler Anfragen kann die Realisierung des Traders verteilt und repliziert erfolgen. Dieser Ansatz wurde im MELODY-Projekt in Zusammenarbeit mit dem debis Systemhaus verfolgt ([Walk94]), bisher jedoch nicht in das aktuelle Release des MELODY-Prototypen integriert.

Das Regelmanagement und die Bewertung von Dienstangeboten sollen flexibler gestaltet werden, so daß beispielsweise wesentlich komplexere Bedingungen bei der Dienstauswahl berücksichtigt werden können. Auch die Kombination von vorgegebenen Regeln und aktuellen Anforderungen des Dienstnutzers soll flexibler gestaltet werden. Hier ist insbesondere auch zwischen individueller und sozialer Optimierung abzuwägen.

Trader unterschiedlicher Domänen erfahren durch den Systemverwalter voneinander. Je nach der im Trader verfolgten Kooperationsstrategie ([Burg95]) können Trader miteinander kooperieren. An der Implementierung des in Kapitel 5 beschriebenen Kooperationsmodells mit Hilfe mehrerer kooperierender Trader wird im Moment gearbeitet.

Für die Zukunft ist eine Erweiterung der Anfragesprache geplant. Ein Ziel ist dabei die Berücksichtigung der Belange eines universellen Traders bei dem z.B. nach einer Menge von Kooperationspartnern gesucht werden kann. Ein zweites Ziel ist die Verbesserung des Optimierungsausdrucks, so daß ein Angebot nach unterschiedlichen Kriterien bewertet werden kann.

9. Zusammenfassung und Ausblick

9.1 Zusammenfassung

Das MELODY-Projekt verfolgt das Ziel, einen allgemeinen Dienst für die Infrastruktur eines offenen Marktes von Diensten zur Verfügung zu stellen. Dieser Bericht beschreibt die Konzepte und Realisierung eines solchen Dienstes durch die Kombination eines Trading- und eines Managementdienstes. Da der Trading-Dienst erst in den letzten Jahren an Bedeutung gewonnen hat, wurde auf seine Darstellung und auf die Beschreibung verschiedener Realisierungsmöglichkeiten ausführlich Wert gelegt. Ein weiterer Schwerpunkt im Zusammenhang mit dem Trading-Dienst war die Beschreibung der Kooperation verschiedener Trading-Domänen, um eine Verbesserung des Trading-Dienstes und eine domänenübergreifende Zusammenarbeit zu ermöglichen.

Im zweiten Teil der Arbeit wurde auf die Implementierung des Trading-Dienstes im Rahmen des MELODY-Projektes ausführlich eingegangen. Der Trading-Dienst arbeitet nach dem Prinzip der gleichberechtigten Integration mit dem MELODY Managementsystem zusammen, welches ebenfalls beschrieben wurde. Der Bericht wird durch eine tabellarische Übersicht verschiedener Aspekte einzelner Trader-Prototypen sowie durch einen ausführlichen Literaturteil beendet.

9.2 Ausblick

Mit Hilfe einer Marktinfrastruktur ist es jedermann möglich, auf schnelle und bequeme Weise sowohl Dienste in das verteilte Systeme einzubringen als auch welche zu nutzen. Die Vermittlung unbekannter Dienste oder die Anwerbung potentieller Nutzer hilft bei der Nutzung der Dienste, während das Management den Markt überwacht und steuert. Insgesamt erlaubt eine solche Marktinfrastruktur eine Konzentration auf die Entwicklung der Dienstfunktionalität, da für viele Aspekte des Managements bereits erprobte Lösungen vorhanden sind.

Eine weitere Verbesserung dieses Ansatzes kann durch die Kombination von existierenden Diensten zu neuen, mächtigeren Dienste geschaffen werden. Ein Trader kann diese bei der Dienstauswahl berücksichtigen. In [PKM93] werden die entstehenden Probleme kurz angerissen und das sehr einfache *Parallel Service Combination* Problem diskutiert. Auch in [Kell94] und [ISO93] wird das analoge Problem der komplexen Dienste kurz angerissen. Eine Lösung der Gesamtproblematik steht jedoch noch aus. Abhilfe könnten hier Verfahren aus dem Bereich der Wiederverwendbarkeit von Software schaffen.

Auch die Problematik der Kooperation in verteilten Systemen gewinnt zunehmend an Bedeutung. Durch den Übergang von einem geschlossenen und eher zentral verwalteten System zu offenen, autonomen Systemen ergibt sich ein erhöhter Bedarf an Abstimmung und Kooperation in verteilten Systemen. Die dafür notwendigen Abstimmungsschritte und Verhandlungen sind eher unzureichend erforscht. Hier ergeben sich auf zweierlei Weise interessante Anknüp-

fungspunkte zum Feld der sogenannten agenten-orientierten Systeme, deren Bedeutung im Zunehmen begriffen ist. Einerseits können die zur Kooperation zwischen Agenten entworfenen Verfahren auf Trader- und Managementkomponenten angewendet werden, wie dies bereits teilweise in Kapitel 5 demonstriert wurde. Andererseits ist zu prüfen, inwieweit die im vorliegenden Bericht beschriebenen Konzepte und Ansätze einer Marktinfrastruktur auch für intelligente, mobile Agenten eine Basis zur Nutzung von Diensten bieten können.

Mobile Nutzer werden in zunehmendem Maße dynamisch auf die Dienste im Netz zugreifen. Die Marktinfrastruktur muß ihnen einen schnellen und möglichst lokalen Zugangspunkt zu den Diensten bieten. Das vorgestellte Trading-Modell trägt dieser Entwicklung Rechnung, indem zwischen dem Dienstanbieter und seinem Zugangspunkt unterschieden wird. Hier eröffnet sich jedoch noch ein weites Spektrum an neuartigen Aufgabenstellungen. Beispielsweise können sowohl Repräsentanten des mobilen Nutzers als auch Dienstinstanzen dynamisch im System angelegt werden oder durch Migration auf neue Knoten wandern. Bei der Dienstvermittlung muß dies berücksichtigt werden, indem beispielsweise der günstigste Ort zur Ausführung der Komponente bestimmt wird.

Bei der Dienstvermittlung ist bisher von einem Bedarf des Dienstnutzers an einem Dienst ausgegangen worden. Aus der Sicht des Dienstanbieters besteht aber auch ein Bedarf an Dienstnutzern, die dieser aktiv einwerben möchte. Ein aktives Trading-System kann dem durch entsprechende Kommunikation mit den Nutzerinstanzen Rechnung tragen. Hier ist es denkbar, daß ein Trader neben der Dienstleistung Dienstvermittlung auch weitere, eigenständige Ziele verfolgt. Beispielsweise kann ein Trader durch eine hohe Kooperationsbereitschaft intensiv um Kunden aus anderen Trader-Domänen bemüht sein und so die Nutzung der eigenen Dienstangebote erhöhen. Die Auswirkung verschiedener Trader-Ziele sowie die Realisierung durch geeignete Traderstrategien ist heutzutage kaum betrachtet.

Die in den Ergebnissen der Dienstvermittlung angesprochenen Nutzungskontrakte sind heutzutage nur sehr rudimentär. Sie entsprechen im Regelfall nur technischen Absprachen zwischen Dienstnutzern und -anbietern. Für die Zukunft läßt sich jedoch erwarten, daß für die Dienstnutzung durch rechtliche Vorgaben oder geschäftliche Anforderungen weitere Regelungen getroffen werden müssen. Diese sind Bestandteil des Dienstkontrakts und müssen bei der Vermittlung berücksichtigt werden.

Insgesamt gesehen stellt die Entwicklung des Trading-Dienstes eine logische Fortsetzung der Namensverwaltung dar. Mit Sicherheit ist abzusehen, daß die heutigen Konzepte den Anforderungen eines offenen Markt von Diensten in vielen Bereichen nur ungenügend entsprechen. Mit der Realisierung des Marktes werden auch neue Anforderungen an die Marktinfrastruktur, sowohl im Bereich Dienstvermittlung als auch im Bereich Dienstmanagement, gestellt.

Appendix A: Klassifikation von Trader-Eigenschaften

Flache Typen	Typen mit Versionen	Typhierarchien	Typübersetzung
SUN RPC [Inc88]	NCS [ZDL+90]	ANSA [ANSA91]	Cygnus ([Chan90], [ChRa94])
	DCE [Schi93]	Emerald [BHJH87]	Type Coercion [MaBl92]
		Objektorientierte Systeme (z.B. [Schi90])	

Tabelle A.1: Interface-Typkompatibilität

Festgelegte Attributmenge	Beliebige Attributmenge	Dynamische Attribute
Debis-Trader [Walk94]	ANSA	MELODY
	MELODY	BERCIM / Y [PTT91]

Tabelle A.2: Attribute

Statische Konfiguration	Export-Operation	Kommunikation mit anderen Tradern	Aktive Dienstsuche
Debis-Trader [Walk94]	ANSA	Föderative Trader	[ScHa91] Web-Wurm
	MELODY	MELODY (Burg95)	

Tabelle A.3: Dienst-Akquirierung

Unstrukturierter Dienstraum	Zellenorientierter Dienstraum	Hierarchischer Dienstraum	Membership Rules
	Debis-Trader [Walk94]	ANSA	[BeRa93]
		MELODY	

Tabelle A.4: Dienstraum

Random	Conservative Choice	First Choice	Cyclic Choice	Historie	Best Match
MELODY (ohne Regeln)			Aachen ([Küpp94])	Haas94	ANSA
					MELODY
TRADE-Trader [MML94b]					

Tabelle A.5: Auswahlstrategien

Boolscher Ausdruck	Gestaffelte boolsche Beschreibung	Strategien (Policies)	Regeln
[BeRa93]	Aachen	ISO ODP Standard	Haas94
MELODY			

Tabelle A.6: Dienstauswahlbeschreibung

Individuelle Optimierung	Systemspezifische Optimierung	Regeln
MELODY	DEFAULT und Implizite Regeln ([KoWi94])	Haas94

Tabelle A.7: Individuelle oder systemübergreifende Optimierung

Föderation	Kooperationsstrategien
ANSA	[Burg95]
[BeRa91]	

Tabelle A.8: Kooperationsunterstützung

Zeitliche Einschränkung der Suche	Längenrestriktionen	Fehlertoleranz	Views
MELODY	MELODY		[Eber93]
ODP-Trader			

Tabelle A.9: Strategien, die die Qualität des Trading-Dienstes beeinflussen

Erwähnung als Ziel	Trading von heterogenen Diensten
[BeBe94]	MELODY (DCE und CORBA-Dienste)

Tabelle A.10: Trading von heterogenen Entitäten

Kontrollierte gemeinsame Nutzung ohne externe Interferenz	Aufteilung administrativer Verantwort-lichkeiten	Erweiterung von Kundenkreis und Markt	Orts-/ Verteilungs-transparenz	Erfüllung aller Anfragen und kein Match
[BeRa91] , [NiGo94]	[BeRa91]	[BeRa91], [ISO94]	[BeRa91], [NiGo94]	[NiGo94], [VBB95]

Tabelle A.11: Motivation zur Kooperation

	Exportierender Trader	Importierender Trader
Kooperationsbeginn	[BeRa91]	[BeRa91] , [NiGo94]
Kooperationsende	[BeRa91] , [NiGo94]	[BeRa91]

Tabelle A.12: Verteilung der Initiative bei einer Kooperation

Instanzenmenge + Beziehungen	Beliebige Abgrenzungskriterien (organisatorisch, physikalisch, dienstspezifisch, Verwaltungsaspekte wie z. B. Sicherheit, Zugangsregeln)	Lokales Verteiltes System und Naming Domain, homogen	Autonome Verwaltungs bereiche, heterogen
[BML91]	[MoSl91a], [SlMo89] , [SLMO89b], Slom90	[GO91], [NiGo94]	[BeRa91]

Tabelle A.13: Semantik von Domänen

Disjunkt	Überlappend
[BeRa91] , [NiGo94], [ISO94]	[ISO94]

Tabelle A.14: Verantwortungsbereiche von Traderinstanzen

1 Objekt	Beliebig viele Objekte (Liste von Diensttypen oder Directoryteile)
[NiGo94]	[BeRa91], [ISO94], [VBB95]

Tabelle A.15: Anzahl exportierter/importierter Objekte je Interaktion

Ausgezeichnetes Attribut	Standardisierter Diensttyp	Andere Beschreibung
[NiGo94]	[BeRa91], [ISO94], [VBB95]	[GSS94], [MML94a], [PMGG95]

Tabelle A.16: Objektbezeichnung

EXPORT	Request/ Search	Exchange, modify contract	Federation Operations	WITH-DRAW	List, ENQUIRY
[BeRa91] , [NiGo94]	[BeRa91] , [NiGo94]	[BeRa91]	[BeRa91]	[BeRa91] , [NiGo94]	[BeRa91] , [NiGo94]

Tabelle A.17: Mögliche Operationen während einer Trader-Kooperation

EXPORT-Bedingungen (Gültigkeitsdauer, Kosten, Rückzugbedingungen)	Umfang des erlaubten Zugriffes, Dienststrukturen, Abbildungsfunktionen, Chaining Flag, Liste erlaubter Standard-Trader-Operationen bzw. zugelassener Benutzer
[NiGo94], [ISO94], [VBB95]	[BeRa91]

Tabelle A.18: Zusätzliche Parameter bei Trader-Kooperationen

Literatur:

[ABC90] Rafael Alonso, Daniel Barbara, and Luis L. Cova. Using Stashing to Increase Node Autonomy in Distributed File Systems. In *Proceedings of the Ninth Symposium on Reliable Distributed Systems*, pages 12–21, Huntsville, Alabama, October 1990. IEEE.

[ANSA91] ANSA. ANSAware 3.0 Implementation Manual. Manual RM.097.01, Architecture Projects Management Limited, February 1991.

[BeBe94] Ashley Beitz and Mirion Bearman. An ODP Trading Service for DCE. In *Presentation at the First International Workshop on Services in Distributed and Networked Environments (SDNE'94)*, Prague, Czechia, June 1994. IEEE.

[BeRa91] M. Bearman and K. Raymond. *Federating Traders: An ODP Adventure*. International IFIP Workshop on Open Distributed Processing, October 1991.

[BeRa93] Mirion Bearman and Kerry Raymond. Contexts, Views and Rules: An Integrated Approach to Trader Contexts. In *Proceedings of the IFIP TC6/WG6.1 International Conference on Open Distributed Processing*, pages 181–19, Berlin, Germany, 13-16 September 1993. North-Holland.

[Beut92] Ulrich Beutenmüller. Modellierung von Ähnlichkeit zwischen Diensten. Diplomarbeit 912, Universität Stuttgart, Stuttgart, August 1992.

[BHJH87] Andrew Black, Norman Hutchinson, Eric Jul, and Larry Carter Henry Levy. Distribution and Abstract Types in Emerald. *IEEE Transactions on Software Engineering*, 13(1), January 1987.

[BML91] G. v. Bochmann, P. Mondain-Monval, and L. Lecomte. Formal Description of Network Management Issues. In I. krishnan and W. Zimmer, editors, *Integrated Network Management*, volume II, pages 77 – 91. IFIP, Elsevier Science Publishers B. V. (North-Holland), 1991.

[Brod94] Michael L. Brodie. Putting Objects To Work On A Massive Scale. In *Vortrag auf der 5-Jahres-Feier des IPVRs*. IPVR, Universität Stuttgart, 1994.

[Burg90] C. Burger. *Leistungsoptimierung im heterogenen Rechnerverbund durch dynamische Zuordnung von Benutzerauftraegen*. PhD thesis, Universitaet Karlsruhe, Fakultaet fuer Informatik, feb 1990.

[Burg95] C. Burger. Cooperation policies for traders. In K. Raymond, editor, *IFIP International conference on open distributed proce ssing ICODP'95*. IFIP, feb 1995.

[BuSe94] C. Burger and F. Sembach. Ein Überblick über Verfahren yur rechnergestützten Kooperation. Technical Report 7, Universität Stuttgart, Fakultät für Informatik, July 1994.

[Chan90] Rong Nickle Chang. A Network Service Acquisition Mechanism for the Client/Service Model. Master's thesis, University of Michigan, 1990.

[ChRa94] Rong N. Chang and Chinya V. Ravishankar. A service Acquisition Mechanism for Server-Based Heterogeneous Distributed Systems. *IEEE Transaktions on Parallel and Distributed Systems*, 5(2):154–169, 1994.

[dPLJMa95] Luiz Augusto de Paula Lima Jr. and Edmundo Roberto Mauro Madeira. A Model for a Federated Trader. In Raymond [Raym95].

[DuLe91] E. H. Durfee and V. R. Lesser. Partial Global Planning. *IEEE Transactions on Systems, Man, and Cybernetics*, 21(6):1167 – 1183, November/December 1991.

[Eber93] Georg Eberhardt. Trading-Aufgaben im Rahmen einer unternehmensweiten Druckdienstes. Diplomarbeit 1013, Universität Stuttgart, Stuttgart, Juli 1993.

[GSS94] M.R. Genesereth, M.P. Singh, and M.A. Syed. A Distributed and Anonymous Knowledge Sharing Approach to Software Interoperation. 1994.

[Haas94] Barbara Haase. Einsatz eines ATM-Netzwerks zur Unterstuetzung von Multimedia in Ingenieuranwendungen. Diplomarbeit, Universität Stuttgart, November 1994.

[Hamh92] Michael Hamhaber. Realisierung der Basisfunktionen eines Traders. Studienarbeit 1165, Universität Stuttgart, Mai 1992.

[HeAb93] Heinz-Gerd Hegering and Sebastian Abeck. *Integriertes Netz- und Systemmanagement*. Addison-Wesley, Bonn, 1993.

[Helb92] Tobias K. Helbig. Strategien zur Einhaltung von Aktualitätsanforderungen. Diplomarbeit Nr. 944, Universität Stuttgart, Institut für Parallele und Verteilte Höchstleistungsrechner, Stuttgart, Dezember 1992.

[HePo92] O. Hermann and C. Popien. Modelling Heterogeneouse CIM-Interface with ODP. In *Direkt von den Autoren erhalten*, 1992.

[IBR93] J. Indulska, M. Bearman, and K. Raymond. A TYPE Management System for an ODP Trader. In *Proceedings of the IFIP TC6/WG6.1 International Conference on Open Distributed Processing*, Berlin, Germany, September 1993. North-Holland.

[Inc87] Sun Microsystems Inc. XDR: External Data Representation standard. RFC 1014, IAB, June 1987.

[Inc88] Sun Microsystems Inc. RPC: Remote Procedure Call Protocol specification: Version 2. RFC 1057, IAB, June 1988.

[ISO92] ISO. Working Document on Topic 9.1 - ODP Trader. *Arbeitspapier des ISO/IEC JTC1/SC21/WG7: N7047*, May 1992.

[ISO93] ISO. Working Document on Topic 9.1 - ODP Trader. *Arbeitspapier der ISO/IEC JTC1/SC21/WG7: N807*, July 1993.

[ISO94] ISO. ISO/IEC 13235 : Draft ODP Trading Function. *Arbeitspapier des ISO/IEC JTC1/SC21: N9122*, Mai 1994.

[ISO95] ISO. ISO/IEC 13235 : Draft ODP Trading Function. *Arbeitspapier des ISO/IEC JTC1/SC21: N???, Draft International Standard*, June 1995.

[Jahk93] Thilo Jahke. Ein objektorientiertes Managementmodell für verteilte Anwendungen. Diplomarbeit Nr. 984, Universität Stuttgart, Institut für Parallele und Verteilte Höchstleistungsrechner, Stuttgart, Dezember 1993.

[Kell93] Ludwig Keller. Vom Name-Server zum Trader - Ein Überblick über Trading in verteilten Systemen -. *PIK - Praxis der Informationsverarbeitung und Kommunikation*, 16(3), 1993.

[Kell94] Ludwig Keller. Trading of Complex Services in Distributed Systems. In *Proceedings of the Fifth IFIP/IEEE International Workshop on Distributed Systems: Ope-*

rations & Management, Toulouse, 1994.

[Kirs94] T. Kirsche. *Kooperation und Koordination in verteilten Systemen*, chapter 57 -75, pages 57 – 75. BI-Wissenschaftsverlag, 1994.

[Kova92] Ernö Kovacs. Effizienter Zugriff auf dynamische Information im Melody-Trader. In *Unveröffentliches Papier*, 1992.

[Kova93a] Ernö Kovacs. Automatic Selection of an Update Strategy for Management Data. In *Proceedings of the IEEE First International Workshop on Systems Management (IWSM'93)*, Los Angeles, April 1993.

[Kova93b] Ernö Kovacs. Das MELODY-Managementsystem für verteilte Anwendungen. In Prof. Dr. K. Geihs Prof. Dr. O. Drobnik, editor, *Tagungsband zum Arbeitstreffen der GI/ITG Entwicklung und Management verteilter Anwendungssysteme*. GI/ITG, Oktober 1993.

[KoWi94] Ernö Kovacs and Stefan Wirag. Trading and Distributed Application Management: An Integrated Approach. In *Proceedings of the Fifth IFIP/IEEE International Workshop on Distributed Systems: Operations & Management*, Toulouse, 1994.

[Kule95] Ottokar Kulendik. Trading von CORBA-Objekten. Master's thesis, Universitaet Stuttgart, August 1995.

[Küpp94] A. Küpper. Eine Strategie zur Verbesserung der Dienstvermittlung unter ANSAware. In *Tagungsband zum 1. Aachener Arbeitstreffen 'Neue Konzepte für die Offene Verteilte Verarbeitung', ISBN 3-86073-143-2*, Aachen, September 1994.

[LeBe95] Ok-Ki Lee and Steve Benford. An Explorative Model for Federated Trading in Distributed Computing Environments. In Raymond [Raym95].

[MaBl92] Alastair J. Macartney and Gordon S. Blair. Flexible trading in distributed systems. *Computer Networks and ISDN Systems*, (15):145–157, 1992.

[Mars93] Lindsay F. Marshall. Representing Management Policy Using Contract Objects. In *IEEE First International Workshop On Systems Management*, Los Angeles, April 1993. IEEE.

[MML94a] M. Merz, K. Müller, and W. Lamersdorf. Service Trading and Mediation in Distributed Computing Systems. In *The 14th International Conference on Distributed Computing Systems*, 1994.

[MML94b] K. Müller, M. Merz, and W. Lamersdorf. Der TRADE-Trader: Ein Basisdienst offener, verteilter Systeme. In *Tagungsband zum 1. Aachener Arbeitstreffen: "Neue Konzepte für die Offene Verteilte Verarbeitung"*, Aachen, September 1994. Augustinus Verlag.

[MML95] K. Müller-Jones, M. Merz, and W. Lamersdorf. The TRADEr: Integrating Trading into DCE. In *IFIP International conference on open distributed processing (ICODP'95)*, Feb 1995.

[MoSl91a] J. D. Moffett and M. S. Sloman. Delegation of Authority. In I. Krishnan and W. Zimmer, editors, *Integrated Network Management*, volume II, pages 595 – 606. IFIP, Elsevier Science Publishers B.V. (North-Holland), 1991.

[MoSl91b] Jonathan D. Moffett and Morris S. Sloman. The Representation of Policies as System Objects. In *Conference on Organizational Computer Systems (COCS'91)*. Imperial College London, November 1991.

[NiGo93] Ying Ni and Andrzej Goscinski. Resource and Service Trading in a Heterogeneous Large Distributed System. In *Proceedings of the IEEE Worshop on Advances in Parallel and Distributed Systems*, Princeton, New Jersey, October 1993.

[NiGo94] Y. Ni and A. Goscinkski. Trader cooperation to enable object sharing amoung users of homogeneous distributed systems. *computer communications*, 17(3):219 – 229, mar 1994.

[OMG91] OMG. The Common Object Request Broker: Architecture And Specification. Technical Report 91.12.1, Object Management Group, December 1991.

[OMG95] OMG. Object Property Service. Technischer bericht, International Business Machines Corporation, SunSoft Incorporated, Taligent Incorporated, Juni 1995.

[Pete88] Larry L. Peterson. The Profile Naming Service. *ACM Transactions on Computer Systems*, 6(4):341–364, 1988.

[PKM93] Claudia Popien, Axel Kuepper, and Bernd Meyer. Trading Enhancement by Service Combination in ODP. In *Position Paper for the International Conferenc on Open Distributed Processing*, 1993.

[PMGG95] A. Puder, S. Markwitz, F. Gudermann, and K. Geihs. AI-based Trading in Open Distributed Environments. In *IFIP International conference on open distributed processing - ICODP'95*. IFIP, Feb 1995.

[PoMe94] Claudia Popien and Bernd Meyer. A Service Request Description Language. In *Proceedings of FORTE'94*, Bern, Switzerland, October 1994.

[PTT91] R. Popescu-Zeletin, V. Tschammer, and M. Tschichholz. 'Y' distributed application platform. *Computer Communications*, 14(6):366–374, July/August 1991.

[PYF+92] M. Palaniappan, N. Yankelovich, G. Fitzmaurice, A. Loomis, B. Haan, J. Coombs, and N. Meyrowitz. The Envoy Framework: An Open Architecture for Agents. *ACM Transactions on Information Systems*, 10(3):233 – 264, July 1992.

[Raym95] Kerry Raymond. International Conference on Open Distributed Systems. IFIP, Chapman and Hall, feb 1995.

[RoZl94] J. S. Rosenschein and G. Zlotkin. *Rules of Encounter: Designing Conventions for Automated Negotiation among Computers*. MIT Press, 1994.

[ScHa91] Michael F. Schwartz and Darren R. Hardy. Supporting Resource Discovery Among Public Internet Archives Using a Spectrum of Information Quality. In *11th International Conference on Distributed Computing Systems*, Arlington, Texas, May 1991. IEEE.

[Schi90] Alexander Schill. *Migrationssteuerung und Konfigurationsverwaltung für verteilte objektorientierte Anwendungen*. Informatik-Fachberichte 241. Springer, Berlin [u.a.], 1990. 174 S.

[Schi93] Alexander Schill. *DCE - Das OSF Distributed Computing Environment*. Springer-Verlag, 1993.

[SlMo89a] M. Sloman and J. Moffett. Managing Distributed Systems. Technical report, Imperial College London, Department of Computing, sep 1989.

[SlMo89b] M. S. Sloman and J. D. Moffett. Domain Management for Distributed Systems. In B. Meandzija and J. Westcott, editors, *Integrated Network Management*, volume I,

pages 505 – 516. IFIP, Elsevier Science Publishers B. V. (North-Holland), 1989.

[Slom90] M. Sloman. Management for open distributed processing. In *Future Trends '90, Second IEEE Workshop on Future Trends of Distributed Computing Systems*, pages 533 – 539. IEEE Computer Society Technical Committee on Distributed Processing, IEEE Computer Society Press, sep 1990.

[SMTK93] M. Sloman, J. Magee, K. Twidle, and J. Kramer. An Architecture For Managing Distributed Systems. In *Proceedings of the 4th Workshop on Future Trends in Distributed Systems*, pages 40–46. IEEE, 1993.

[TWH90] V. Tschammer, A. Wolisz, and J. Hall. Support for cooperation and coherence in an open service environment. In *Workshop on Future Trends of Distributed Computing Systems*, volume 2, pages 222 – 228. IEEE, sep/oct 1990.

[vB90] Reinhard Gotzhein and Gregor von Bochmann. Deriving Protocol Specifications from Service Specifications Including Parameters. *ACM Transactions on Computer Systems*, 8(4), November 1990.

[VBB95] Andreas Vogel, Mirion Bearman, and Ashley Beitz. Enabling Interworking of Traders. In *Submitted for the International Conference on Open Distributed Processing (ICODP'95)*, Queensland Australia, 1995.

[Walk94] Dirk Walkenhorst. Entwicklung eines verteilten Traders unter besonderer Berücksichtigung der Fehlertoleranz und Effizienz. Diplomarbeit, Universitä Stuttgart, Stuttgart, 1994.

[Wies94] R. Wies. Policy Definition and Classification: Aspects, Criteria, and Examples. In *Proc. 5th IFIP/IEEE International Workshop on Distributed Systems: Operations & Management*, oct 1994.

[Wira94] Stefan Wirag. Einsatz dynamischer Parameter bei der Dienstauswahl. Diplomarbeit 1088, Universität Stuttgart, Diplomarbeit Nr. 1088, Februar 1994.

[Wong93] S. T. C. Wong. COSMO: A Communication Scheme for Cooperative Knowledge-based Systems. *IEEE Transactions on Systems, Man, and Cybernetics*, 23(3):809 – 824, May/June 1993.

[WoTs90] A. Wolisz and V. Tschammer. Service provider selection in an open services environment. In *Workshop on Future Trends of Distributed Computing Systems*, volume 2, pages 229 – 235. IEEE, sep/oct 1990.

[WoTs91] Adam Wolisz and Volker Tschammer. Some Performance Aspects of Trading Service Design. In IEEE, editor, *IEEE Conference*, pages 919–927, 1991.

[ZDL+90] L. Zahn, T.H. Dineen, P.J. Leach, E.A. Martin, N.W. Mishkin, J.N. Pato, and G.L. Wyant. *Network Computing Architecture*. Prentice Hall, Englewood Cliffs,N.J., 1990.